Proving Associative-Commutative Termination Using RPO-compatible Orderings*

Deepak Kapur[1] and G. Sivakumar[2]

[1] Department of Computer Science, University of New Mexico,
kapur@cs.unm.edu
[2] Computer Science Department, Indian Institute of Technology, Bombay, India
siva@cse.iitb.ernet.in

Abstract. Developing path orderings for associative-commutative (AC) rewrite systems has been quite a challenge at least for a decade. Compatibility with the recursive path ordering (RPO) schemes is desirable, and this property helps in orienting the commonly encountered distributivity axiom as desired. For applications in theorem proving and constraint solving, a total ordering on ground terms involving AC operators is often required. It is shown how the main solutions proposed so far ([7],[13]) with the desired properties can be viewed as arising from a common framework. A general scheme that works for non-ground (general) terms also is proposed. The proposed definition allows flexibility (using different *abstractions*) in the way the candidates of a term with respect to an associative-commutative function symbol are compared, thus leading to at least two distinct orderings on terms (from the same precedence relation on function symbols).

1 Introduction

Rewrite systems provide a useful model of computation based on the simple inference rule of "replacing equals by equals." Rewrite techniques have proved successful in many areas including equational programming, theorem proving, specification and verification, and proof by induction.

Rewriting techniques particularly are effective in reducing the search space for finding proofs because of the ability to orient equations into one-directional rewrite rules. Rewrite rules are used for "simplifying" expressions by repeatedly replacing instances of left-hand sides by the corresponding right-hand sides. For example, the rules below express addition and multiplication over natural num-

* This paper is a revised version of an earlier draft entitled *A recursive path ordering for proving associative-commutative termination* [8] by the authors which was published as a technical report of the Department of Computer Science, State University of New York, Albany, NY 12222, May 1998. This research has been partially supported by the National Science Foundation Grant nos. CCR-9712366, CCR-9712396, and CDA-9503064.

bers.

$$\boxed{\begin{array}{rcl} 0+x & \rightarrow & x \\ s(x)+y & \rightarrow & s(x+y) \\ 0*x & \rightarrow & 0 \\ s(x)*y & \rightarrow & y+(x*y) \end{array}}$$

A sample derivation chain is $s(0)*s(0) \rightarrow s(0)+(0*s(0)) \rightarrow s(0)+0 \rightarrow s(0+0) \rightarrow s(0)$.

Termination of such derivations is crucial for using rewriting in proofs and computations, as well as for mechanizing proofs by induction. One approach to prove termination is to design *well-founded orderings* on terms which include the *rewrite* relation. Several syntactic *path orderings* based on extending a precedence relation $\succ$ on function symbols to terms have been developed [4].

The Recursive Path Orderings (RPO) [4] are the most commonly used orderings in rewrite-based theorem provers . When $\succ$ is a total precedence relation on function symbols, RPO is total (up to equivalence) on ground terms (terms without variables). That is, given two distinct ground terms, either they are equivalent under the ordering, or one of them is bigger. Total orderings on ground terms have been found useful in theorem proving and constraint solving.

Many interesting and useful theories use operators (such as $+,*,\vee,\wedge,\oplus$) which are *associative* and *commutative*. We refer to such operators as AC-operators and terms using these operators as AC-terms in the rest of the paper. Developing well-founded orderings on AC-terms which are useful for proving termination of AC-rewrite systems has been quite a challenge. A number of attempts have been reported in [1, 15, 3]; see also [11, 2] for polynomial orderings as well as [5] for other approaches.

Two properties can be used to distinguish between most of the related previous work and to motivate the work done in this paper. The first property is *RPO-compatibility*. That is, whether an AC-ordering behaves exactly like RPO on non-AC terms as well as allows (like RPO) the orientation of the distributivity axiom $(x*(y+z) \rightarrow (x*y)+(x*z)$ as desired by making $* \succ +$, even though both $+$ and $*$ are AC-operators. The second property is the *ground totality*. That is, whether the proposed ordering is total on ground terms when the precedence relation on function symbols is total.

In 1990, Kapur, Sivakumar and Zhang proposed a general ordering scheme based on recursive path ordering with status (rpos) without any restrictions on the precedence relation between function symbols [9]. Their ordering scheme however has a weakness: it is not total on equivalence classes of AC ground terms even when the precedence relation on function symbols is total. A total ordering on AC-ground terms was first proposed by Narendran and Rusinowitch [12] based on polynomial interpretations (hence, it is not RPO-compatible). That ordering does not orient distributivity appropriately.

Rubio and Nieuwenhuis introduced a total ordering on ground terms for a total precedence relation on function symbols [14] which uses both interpretations similar to [1], and the idea of *elevation* used in [9]. They showed how their ordering can be lifted to non-ground terms. The main weakness of their ordering is that it does not orient distributivity properly. In particular, even if $* > +$, $x * (y + z) < (x * y) + (x * z)$, so that ordering is not RPO-compatible either.

In [7], we proposed a path ordering based on rpos that is total on AC ground terms, and orients the distributivity law properly by making $a * (b + c) > (a * b) + (a * c)$ if $* > +$. However, we could not give a natural extension of the definition to non-ground terms, and had to resort to approximations and constraint solving for extending the ordering to general terms. Using ideas similar to [7], Rubio proposed an ordering using a bottom-up construction [13]. He defined an interpretation associated with a term (with respect to an AC symbol) as a multiset of sequences. This ordering has all the desired properties on ground terms, but does not lift to non-ground terms. In [13], Rubio stated "the ordering we have defined on ground terms ... cannot be used directly on terms with variables." He instead gave two approximations to extend his interpretations to non-ground terms. While the first approximation is not as complex, it is unable to handle comparisons of simple terms such as $f(i(i(x)), x)$ with $f(i(x), i(x))$ with $f \succ i$. The second approximation is quite complex.

In this paper, we show how the main ideas used in the earlier approaches can be cast in a uniform framework. The proposed scheme simplifies the definitions in our earlier paper [7] even though the basic concepts of candidates and constructions for generating candidates in the two papers are related. More importantly, it is possible to extend the definition directly to non-ground (general) terms. This scheme also allows us to identify the main parameter— the *abstraction* used to compare contributions from function symbols ignored during *elevation*— that allows variants (two are given in this paper) of the basic ordering.

The rest of this paper is organized as follows. In Section 2, we give the relevant definitions and background for proving termination using RPO. In Section 3, we explain the major issues in designing a RPO-compatible ordering for AC-terms using illustrative examples to bring out the key ideas that have been proposed. In Section 4, we define an ordering scheme for comparing AC-terms. It is parameterized by an *abstraction* function, giving at least two distinct orderings for the same precedence relation on function symbols. Sections 5 and 6 discuss proofs, focusing on irreflexivity, transitivity, subterm, replacement and stability properties. It is also shown that the ordering is total on nonequivalent ground terms. We conclude in Section 7 with discussion of related work and suggestions for future work.

2 Rewrite systems and Simplification Orderings

Let $T(F, X)$ be a set of terms constructed from a (finite) set F of function symbols and a (countable) set X of variables. We normally use the letters a through h for function symbols; s, t, and u through w for arbitrary terms; x, y, and z for variables. Each function symbol $f \in F$ has an *arity* $n \geq 0$; *constants* are function symbols of arity zero. Variable-free terms are called *ground* .

A term t in $T(F, X)$ may be viewed as a finite ordered tree. Internal nodes are labeled with function symbols (from F) of arity greater than 0. The out-degree of an internal node is the same as the arity of the label. Leaves are labeled with either variables (from X) or constants. We use $root(t)$ to denote the symbol labeled at the root of the tree corresponding to t. A subterm of t is called *proper* if it is distinct from t. By $t\mid_\pi$, we denote the *subterm* of t rooted at *position* π. Let u be a term and π a position in u. We use $u[\cdot]_\pi$ to denote the *context* for position π in u. Loosely speaking, the context is the tree obtained by deleting the subterm at position π leaving a "hole" in the term. We use $u[t]_\pi$ to denote a term that has t plugged in as a subterm at the "hole" in the context $u[\cdot]_\pi$.

A substitution σ is a mapping from variables to terms such that $x\sigma \neq x$ for a finite number of variables. The depth of a substitution σ is the maximum of the depths of the terms used in this mapping. A substitution can be extended to be a mapping from terms to terms. We use $t\sigma$ to denote the term obtained by applying a substitution σ to a term t. For example, the ground substitution $\sigma = \{x \mapsto k(a), y \mapsto d\}$ when applied to the term $t = f(x, y)$ gives the ground term $t\sigma = f(k(a), d)$.

A rewrite *rule* over a set $T(F, X)$ of terms is an ordered pair (l, r) of terms such that the variables in r also appear in l, and is written $l \to r$. A *rewrite system* (or *term rewriting system*) R is a set of such rules. Rules can be used to replace instances of l by corresponding instances of r.

One approach to proving termination of rewrite systems is to use simplification orderings [4]. A simplification ordering has the following properties.

1. **Subterm Property**: $u[t] \succ t$ for any term t and a non-empty context $u[\cdot]$.
2. **Monotonicity**: $s \succ t$ implies that $u[s]_\pi \succ u[t]_\pi$, for all contexts $u[\cdot]$, terms s and t, and positions π.

We have omitted the deletion property which is needed only if there are function symbols with varying arity. Any simplification ordering $\succ$ on terms is well-founded, that is, there is no infinite descending chain of terms $t_1 \succ t_2 \succ t_3 \cdots$.

A simplification ordering that also has the following property

- **Stability**: for all terms s and t, $s \succ t$ implies that for all substitutions σ, $s\sigma \succ t\sigma$.

can be used for proving termination of rewrite systems. If for every rule $l \to r$ in R, $l \succ r$ in a simplification ordering $\succ$ which is stable under substitutions, then $\to$ is terminating.

2.1 Recursive path ordering

Let $\succ$ be a well-founded precedence relation on a set of function symbols F. For simplicity and without loss of generality, we assume in the rest of the paper that $\succ$ is *total*, i.e., any two distinct function symbols f, g are comparable using $\succ$ (i.e. they are equivalent or one is bigger). Also without loss of generality, we assume that there is at least one constant symbol in F, and we denote the smallest constant in F by a. The recursive path ordering extends $\succ$ on function symbols to a well-founded ordering on terms [4]. For convenience we do not use left-to-right or right-to-left status for any operator as in LRPO although all results in this paper easily extends to this also.

Definition 1. Two terms t and s are *equivalent* ($t \sim s$) if they are either the same variable, or both are non-variables i.e. $t = f(t_1, \ldots, t_n)$, $s = g(s_1, \ldots, s_m)$ and $f = g$, $n = m$, and there is a permutation p of $(1, \ldots, n)$ such that $t_i \sim s_{p(i)}$.

Definition 2 (RPO[4]). $s = f(s_1, \ldots, s_n) \succ g(t_1, \ldots, t_m) = t$ iff one of the following holds.

1. $f \succ g$, and $s \succ t_j$ for all j $(1 \leq j \leq m)$.
2. $f = g$, and $\{s_1, \ldots, s_n\} \succ^{mul} \{t_1, \ldots, t_m\}$.
3. $f \not\geq g$, and for some i $(1 \leq i \leq n)$, either $s_i \sim t$ or $s_i \succ t$.

The ordering $\succ$ is a simplification ordering [4]. Also, when the precedence relation on F is total, two distinct ground terms are either equivalent or comparable under $\succ$.

2.2 AC-rewriting and its termination

With operators that are associative and commutative, the definition of rewriting needs to be modified to include the consequences of these two properties which are not explicitly added as rules (since $x * y = y * x$ cannot be oriented). Let F_{AC} denote the set of such operators.

Consider a rule $a * b \to c$, where $* \in F_{AC}$. The term $t = (b * c) * a$ cannot be rewritten directly if we use the definition of $\to$ given earlier, as no subterm of t is of the form $a * b$. But t is equivalent to the term $s = (a * b) * c$ (using the AC properties of $*$) and $s \to c * c$. AC-rewriting ($\to_{AC}$) is defined to cover such cases as follows.

Definition 3 (AC-rewriting). $u[t] \to_{AC} u[s']$ if $t \leftrightarrow_{AC} \cdots \leftrightarrow_{AC} s$ (re-arranging arguments of AC-operators) and $s \to s'$ using $l \to r$ in R.

With this definition of rewriting, we need to ensure that any simplification ordering also has the following property to ensure its usefulness in proving termination of AC-rewriting.

Definition 4 (AC-Compatibility). A simplification ordering $\succ$ is AC-compatible if, for any terms t, t_1, s, and s_1, if $t \sim^* t_1$, $s \sim^* s_1$, and $t \succ s$, then $t_1 \succ s_1$.

2.3 Flattening

A key idea that figures in many approaches to AC-rewriting is to treat AC-operators not as strictly binary functions (arity = 2) but to let them be vary-adic and use *flattening* (of AC-operators) to convert terms like $(a*b)*c$ to $*(a, b, c)$. We use $\overline{t}$ to denote the flattened form of a term t.

Definition 5. The flattened form $\overline{t}$ of a term t is defined below.

$$\overline{t} = \begin{cases} x & \text{if } t = x \text{, a variable} \\ f(\overline{t_1}, \ldots, \overline{t_n}) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin F_{AC} \\ f(T_1 \cup \cdots \cup T_n) & \text{if } t = f(t_1, \ldots, t_n), f \in F_{AC} \text{ and} \\ & \quad T_i = \begin{cases} \{s_1, \ldots, s_m\} & \text{if } \overline{t_i} = f(s_1, \ldots, s_m) \\ \{\overline{t_i}\} & \text{otherwise} \end{cases} \end{cases}$$

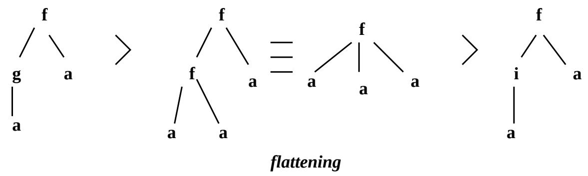
The important question that arises is, whether the termination of $\rightarrow_{AC}$ can be proved using $\succ$ and treating AC-operators just like other function symbols (except that they have variable arity), and using flattened terms always. That is, we may attempt to define an ordering $\succ$ on AC-terms using $s \succ t$ iff $\overline{s} \succ \overline{t}$.

3 Issues in extending RPO to AC-terms

It is well known that defining RPO on flattened terms does not give a simplification ordering on AC terms. We show through a series of examples, first with ground terms and then with terms containing variables, the difficulties encountered in adapting RPO to terms with AC-operators. We then introduce the notion of *candidates* for a term, taking into consideration how different subterms with an outermost symbol smaller than an AC-operator are elevated; this concept of elevation was first used in [9]. We motivate possible *abstractions* that can be used when comparing candidates. This leads to several versions of AC-RPO including an ordering similar to [7] and an ordering in [13].

In all examples henceforth, we use $g \succ f \succ j \succ i \succ c \succ b \succ a$ with $f \in F_{AC}$ as the precedence relation.

Using RPO, $g(a) \succ f(a, a) \succ i(a)$. By *monotonicity*, the following chain must hold. The second inequality does not hold under RPO since $\{i(a), a\} \succ \{a, a, a\}$.



This shows that when comparing two terms having the same AC-operator as root symbol, the arguments cannot be compared simply as multisets (or sequences) as in RPO. A more nuanced comparison of arguments is needed to prove $f(a, a, a) \succ f(i(a), a)$.

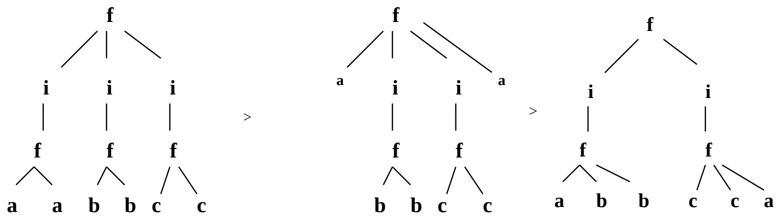
The number of arguments (referred to as *fcount* later) below the AC-operator has a role to play to account for the effect of flattening. A similar example is $f(a, a, a) \succ f(c, c)$. A useful idea that helps is to compare *small* terms (no symbol bigger than f) first by number of such terms and then only the actual terms themselves, is to abstract such terms by a (the smallest constant) first.

It may be useful to observe that when operators bigger than f are involved, the situation is somewhat different. That is, since $g(a) \succ f(c, \ldots, c)$ (any number of c's since $g \succ f$), we have that $f(g(a), a) \succ f(c, \ldots, c)$. So, the number of arguments becomes a factor only when the *big* arguments (those with root symbol bigger than f) do not play any role in the strict comparison.

To summarize, to maintain the monotonicity property, in comparing two ground terms, an argument of f with a symbol bigger than f can take care of many smaller arguments of f (which do not have any occurrence of a symbol bigger than f), but otherwise, the number of arguments to f becomes relevant.

A second interesting observation is that since $f(b, c) \succ i(f(a, c))$ by RPO, we should make $f(b, c, b, c) \succ f(i(f(a, c)), i(f(a, c)))$. That is, the role of the smaller operator i in between nested $f's$, that can be **elevated**, is less important than that of the actual arguments b, c below the AC-operator, even though i is bigger than b and c.

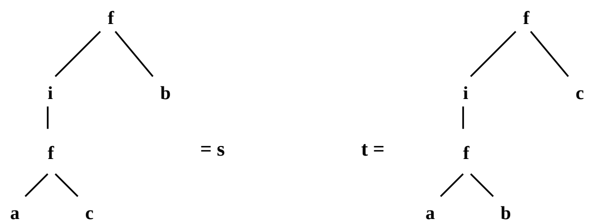
The following example shows a third dimension–how the arguments of an AC-operator are partitioned. The first comparison is because we have simply



replaced $i(f(a, a))$ by its subterm $f(a, a)$ and flattened. The second follows once we note that $f(a, i(f(c, c))) \succ i(f(a, c, c))$ by RPO.

In the examples above, the requirements of any *simplification* ordering such as the subterm and monotonicity properties, forced the outcome of the comparison. Whatever method of comparing AC-terms we provide must preserve the order in these examples.

In the following example we have persevered the arguments below the AC-



operator and kept the partition sizes the same. Then, different ways of defining comparison of AC-terms could orient these differently. Using [7], $t \succ s$, whereas using [13] as well as the orderings proposed in this paper, $s \succ t$.

3.1 Candidates

The main approaches for developing RPO-compatible orderings for AC-terms ([9],[7],[13]) have tackled the problems identified above in slightly different ways. All of them change RPO only in the case when we compare $s = f(S)$ with $t = f(T)$ with $f \in F_{AC}$. They give different ways to compare S and T.

A unifying framework can be developed using the common threads. Some useful definitions are given first.

Definition 6 (Big terms). A term t is called *big* with respect to an AC-operator f, if $t = g(t_1, \ldots, t_m)$ with $g \succ f$.

Definition 7 (Small Terms). A term t is called *small* with respect to an AC-operator f, if t is a ground term with all function symbols and constants in t smaller than f.

If a non-variable term is neither Big nor Small, then it is said to be *elevatable*.

Definition 8 (Elevatable position). A position λ in $t = i(t_1, \ldots, t_n)$ with $f \succ i$ is said to be elevatable with respect to $f \in F_{AC}$, if for all proper prefixes λ' of λ, we have $f \succ top(t|_{\lambda'})$ and $top(t|_{\lambda}) \geq f$, or $t|_{\lambda}= x$ a variable.

Definition 9 (Elevatable subterm). A subterm t' at position λ in the term $t = i(t_1, \ldots, t_n)$ is said to be elevatable with respect to f if λ is an elevatable position in t with respect to f.

For example, in the term $j(g(x), i(f(x, a)))$, we have two elevatable subterms $g(x)$ and $f(x, a)$ with respect to f. In the term $t = j(x, j(g(a), i(y)))$, we can elevate x, $g(a)$ or y; in $f(a, j(f(b, c), g(a)))$, $f(b, c)$ as well as $g(a)$ can be elevated.

Definition 10 (Elevation). We can derive s from t by *elevation* (denoted $t \Rightarrow s$) if $t = f(\ldots, i(t_1, \ldots, t_n), \ldots)$ with $f \succ i$, and $s = \overline{f(\ldots, t', \ldots)}$ for some term t' elevatable from $i(t_1, \ldots, t_n)$ with respect to f. We say that t elevates to s.

For example, $f(a, j(f(b, c), g(a)))$ elevates to either $f(a, g(a))$ or $f(a, b, c)$ depending on which subterm is used in the elevation.

We say that a term $t = f(T)$ is in elevation normal form (*enf*) iff $T = B \cup V \cup S$ where B is a multiset of big term, V is a multiset of variables and S is a multiset of small terms. If we further require that each small term $s \in S$ is only a the smallest constant, then we call such a term as being in *argument normal form*. We can convert a term $s = f(S)$ to argument normal form $s' = anf(s)$, by first doing all possible elevations and then replacing all maximal small terms by a.

Comparing terms in argument normal form by simply comparing the arguments as multisets is safe to do. This has been used as a subcase in many previous approaches to designing AC-RPO.

With this comparison of terms, a sufficient condition for $s = f(S) \succ f(T) = t$ can be stated.

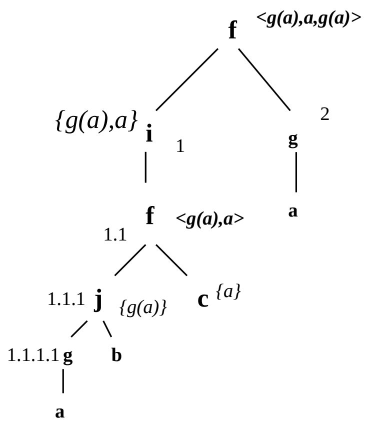
Definition 11 (ACRPO-1). $s = f(S) \succ f(T) = t$ if for every $t' \in anf(t)$, there is a $s' \in anf(s)$ such that $s' \succ t'$.

Note that this definition is sufficient to handle some of the problem cases described earlier. For example, we can show $f(a,a,a) \succ f(i(a),a)$.

ACRPO-1, though sound, is not enough to ensure that we have a simplification ordering since it cannot show, for example, that
$f(i(f(a,a)), i(f(a,a)), i(f(a,a))) \succ f(i(f(a,a,a)), i(f(a,a,a)))$.

It is essential to give some way to compare terms whose biggest anf-forms are identical as in the case above. It becomes necessary, therefore, to build up another view (called *Context* in [7]) while deriving the enf-form (called *Arguments* in [7]). The role of *context* is to bring back the actual values of the Small terms (replaced by a), the smaller operators (such as i) lost during the elevation, and the role they play in partitioning the arguments.

Consider the example shown in the figure. The view as *Arguments* is $\{g(a), a, g(a)\}$



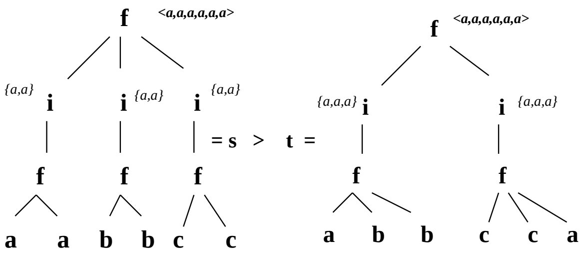
(the anf-form) with the elevations happening at position 1 and later at the subterm $j(g(a),b)$ at position 1.1.1. We can define the context of this view as the multiset of pairs $\{\langle\{g(a)\}, j(g(a),b)\rangle, \langle\{g(a),a\}, i(f(j(g(a),b),c))\rangle, \langle\{a\}, c\rangle\}$ which is the collection of the subterms in which elevations occurred and by the contribution of this elevated subterm to the view as Arguments, and also the Small Terms in the elevation normal form that were replaced by a.

This leads to the idea of *candidates*. A candidate for a term $s = f(S)$ (with $f \in F_{AC}$) is obtained by taking from each $s_i \in S$, its contribution to two different "views" of S– **Arguments** and **Contexts**. When comparing two terms $f(S_1)$ and $f(S_2)$, the views as Arguments are compared first, and only if they are equivalent, the contexts are compared. The interesting issue is how contexts should be compared.[1]

[1] Definitions of [7] have been simplified here. The context in a candidate as defined in [7] is a multiset of pairs of terms corresponding to the order in which elevatable

3.2 Context Comparison

We go back to the example shown earlier for partitioning to illustrate some im-



portant issues in context comparison. The view as Arguments is the same for both terms: $(\{a, a, a, a, a, a\})$. So, we have to consider the role of the contexts C_s and C_t where dropping common pairs,
$C_s - C_t = \{\langle\{a, a\}, i(f(a, a))\rangle\langle\{a, a\}, i(f(b, b))\rangle\langle\{a, a\}, i(f(c, c))\rangle\}$ and
$C_t - C_s = \{\langle\{a, a, a\}, i(f(a, b, b))\rangle\langle\{a, a, a\}, i(f(a, c, c))\rangle\}$ and we must have $C_s \succ C_t$.

Since the Arguments are overall the same on both sides, if an elevated subterm has contributed less, that means that it appears in a bigger context (more contribution from the other arguments), and it should be given more weight. This suggests that the contribution of the elevated subterms to the Arguments play an *inverse* role in the comparison.

Pairs in contexts are thus compared by first using union of contributions from other subterms to this view (Arguments) first, and then going to the elevated subterms themselves.

That is, if $f(S)$ has a candidate $\langle A_s, C_s\rangle$ and $f(T)$ has a candidate $\langle A_t, C_t\rangle$ and we need to compare C_s with C_t (since $A_s = A_t$), then for every pair $\langle A, tt\rangle \in C_t - C_s$, we look for a pair $\langle B, ss\rangle \in C_s - C_t$ such that either $A_s - B$, the context of ss, takes care of $A_t - A$, the context of tt, or the two contexts are equivalent and $ss \succ tt$.

In the example above, we compare
$C'_s = \{\langle\{a, a, a, a\}, i(f(a, a))\rangle\langle\{a, a, a, a\}, i(f(b, b))\rangle\langle\{a, a, a, a\}, i(f(c, c))\rangle\}$ and
$C'_t = \{\langle\{a, a, a\}, i(f(a, b, b))\rangle\langle\{a, a, a\}, i(f(a, c, c))\rangle\}$ and as desired we get $C_s \succ C_t$.

For comparing the contexts themselves ($A_s - B$ with $A_t - A$), variations are possible. One is to compare them just as multiset of terms (the *term abstraction*) using the same ordering. This leads to a definition similar, but not exactly the same, as the one in [7]. Another approach is to simply compare their lengths (number of terms in $A_s - B$ with the number in $A_t - A$) since this is a measure of how the views of Arguments are partitioned.

---

subterms are elevated; the first component in the pair is the rest of the term obtained after deleting the elevated argument, and the second component is the elevated argument itself.

An example that distinguishes the two orderings is given: $s = f(i(g(b)), g(c))$, $t = f(i(g(c)), g(b))$. The Arguments for both s and t are the same: $\{g(b), g(c)\}$. In s, $g(b)$ is elevatable, whereas in t, $g(c)$ is elevatable. The context of the candidate of s is $\{\langle\{g(b)\}, i(g(b))\rangle\}$; the context of the candidate of t is $\{\langle\{g(c)\}, i(g(c))\rangle\}$; If we use the *fcount* abstraction, then $t \succ s$. Using the *term* abstraction, we get $s \succ t$.

4 Definition of AC-RPO ($\succ_{ac}$)

As indicated in the previous section, a *candidate* for a term t is a pair $\langle A, C\rangle$ where A is the view as arguments, and C is the context for this view. The multiset of candidates of any term t with respect to an AC-operator f is defined formally below as $cands(t, f)$.

Definition 12 (Candidates).

1.	$cands(x, f) \Rightarrow$	$\{\langle\{x\}, \phi\rangle\}$	
2.	$cands(t = g(t_1, \ldots, t_n), f) \Rightarrow$	$\{\langle\{t\}, \phi\rangle\}$	if $g \succ f$.
3.	$cands(t, f) \Rightarrow$	$\{\langle\{a\}, \{\langle\{a\}, t\rangle\}\rangle\}$ if t is a Small term wrt f.	
4.	$cands(f(t_1, \ldots, t_n), f) \Rightarrow$	$\{\biguplus_i c_{t_i} \in cands(t_i, f)\}$	
5.	$cands(t = i(t_1, \ldots, t_n), f) \Rightarrow$	$\bigcup\{\langle A', C' \cup \{\langle A', t\rangle\}\rangle \mid \langle A', C'\rangle \in cands(t', f)\}$ if $f \succ i$ and t' is elevatable from t.	

The first three rules are the base cases. A term t which is a variable, or whose top symbol is bigger than f (i.e. a Big term), contributes itself to Arguments and nothing to the Context. A Small term contributes a to arguments (the smallest constant) and $\langle\{a\}, t\rangle$ to context.

Rule 4 defines the candidates of $f(t_1, \ldots, t_n)$ to be union of candidates obtained by the component-wise union of one candidate from each of the arguments $t_1, \ldots, t_n$. Note that if t_j has n_j candidates, then the number of candidates of $f(t_1, \ldots, t_n)$ is the **product** of the n_j-s.

Finally, Rule 5 defines the interesting case when $i < f$ is the top symbol of $t = i(t_1, \ldots, t_n)$ and t has some elevatable subterm t'. A candidate $c_t = \langle A_t, C_t\rangle$ for t is obtained from each candidate $c_{t'} = \langle A'C'\rangle$ of t' by not changing the view in Arguments (i.e. $A_t = A'$), and adding to the view as Context the pair $\langle A', t\rangle$ (i.e. $C_t = C' \cup \langle A', t\rangle$). In this case, if each elevatable t' has n_j candidates, the number of candidates is the **sum** of the n_j-s.

A few illustrative examples of candidates is given below. In all these examples $g \succ f \succ j \succ i \succ c \succ b \succ a$ is the precedence relation, as before.

Example 1. $s_1 = f(i(i(i(a))), b)$ has only one candidate:
$\langle\{a, a\}, \{\langle\{a\}, b\rangle, \langle\{a\}, i(i(i(a)))\rangle\}\rangle$. Notice that both small terms have contributed only a to the Argument view.

Example 2. $s_2 = f(i(g(b)), c)$ has only one candidate:
$\langle\{g(b), a\}, \{\langle\{g(b)\}, i(g(b))\rangle\langle\{a\}, c\rangle\}\rangle$

Example 3. $s_3 = f(i(f(b,c)),c,c)$ also has only one candidate:
$\langle\{a,a,a,a\},\{\langle\{a\},b\rangle,\langle\{a\},c\rangle,\langle\{a\},c\rangle,\langle\{a\},c\rangle,\langle\{a,a\},i(f(b,c))\rangle\}\rangle$. Notice that $i(f(b,c))$ contributes $\{a,a\}$ to Arguments.

Example 4. $s_4 = f(j(g(x),g(c)),g(z))$ has two candidates:
$\langle\{g(x),g(z)\},\{\langle\{g(x)\},j(g(x),g(c))\rangle\}\rangle$ and $\langle\{g(c),g(z)\},\{\langle\{g(c)\},j(g(x),g(c))\rangle\}\rangle$ depending on which argument of $j(g(x),g(c))$ is elevated.

Example 5. $s_5 = f(i(c),j(f(b,b),y))$ also has two candidates:
$\langle\{a,a,a\},\{\langle\{a\},i(c)\rangle,\langle\{a\},b\rangle,\langle\{a\},b\rangle,\langle\{a,a\},j(f(b,b),y)\rangle\}\rangle$ if $f(b,b)$ is elevated from $j(f(b,b),y)$. Or,
$\langle\{a,y\},\{\langle\{a\},i(c)\rangle,\langle\{y\},j(f(b,b),y)\rangle\}\rangle$ if y is elevated from $j(f(b,b),y)$ which then gets $\{y\}$ as its contribution to the Arguments.

Example 6. $s_6 = f(i(f(x,i(f(b,y)))),c)$ has only one candidate:
$\langle\{x,a,y,a\},\{\langle\{a\},b\rangle,\langle\{y,a\},i(f(b,y))\rangle,\langle\{x,y,a\},i(f(x,i(f(b,y))))\rangle\langle\{a\},c\rangle\}\rangle$
Note that the repeated elevation from $i(f(x,i(f(b,y))))$ adds one pair to the context each time with its contribution to the Arguments.

4.1 Properties of Candidates

Definition 13 (f-blocked). A term t is said to be f-blocked if it is a variable, a Big term, or the smallest constant a.

Property 14. Let $\langle A_t, C_t\rangle \in cands(t,f)$. Then,

1. Every term in A_t is an f-blocked term. That is A_t contains only big terms, variables, or a (smallest constant).
2. $A_t = \{t\}$ or A_t is a multiset of proper subterms of t. So, if $\succ$ is any ordering with the subterm property, then $\{t\} \geq A_t$.
3. In every pair $\langle A, tt\rangle \in C_t$,
   (a) A contains only f-blocked terms.
   (b) tt is never a variable. It is either Small (ground) term or has an elevatable subterm with respect to f.
   (c) $A \subseteq A_t$. If $top(t) = f$, then $A \subset A_t$ (strict subset) and tt is a proper subterm of t.

Property 15 (Equivalent Candidates). If $s \sim t$ and $f \in F_{AC}$, then for every candidate $\langle A_t, C_t\rangle \in cands(t,f)$, there is a candidate $\langle A_s, C_s\rangle \in cands(s,f)$ (1-1 correspondence) which is component-wise equivalent. That is, $cands(s,f) = cands(t,f)$ ($=$ is modulo $\sim$).

Proof. Since $s \sim t$, we can map the positions used in t to derive any $\langle A_t, C_t\rangle$ to an appropriate set of positions (arguments can be permuted) in s and use it to derive the corresponding $\langle A_s, C_s\rangle$.

Property 16 (Distinct Candidate Sets). If $s \not\sim t$ and $f \in F_{AC}$, then $cands(s,f) \neq cands(t,f)$.

4.2 Comparing Terms

Definition 17 (AC-RP0). A non-variable term $s \succ_{ac} x$ for any variable x in s. Otherwise, $s = f(s_1, \ldots, s_n) \succ_{ac} g(t_1, \ldots, t_m) = t$, if one of the following holds.

1. $f \succ g$, and $s \succ_{ac} t_j$ for all j $(1 \leq j \leq m)$.
2. $f = g \notin F_{AC}$, and $\{s_1, \ldots, s_n\} \succ_{ac}^{mul} \{t_1, \ldots, t_m\}$.
3. $f = g \in F_{AC}$, and $cands(s, f) \succ_c^{mul} cands(t, f)$.
   (That is, for every candidate $c_t \in cand(t, f) - cands(s, f)$, there is a candidate $c_s \in cands(s, f) - cands(t, f)$ such that $c_s \succ_c c_t$.)
4. $f \not\succeq g$ and for some i $(1 \leq i \leq n)$, either $s_i \sim t$ or $s_i \succ_{ac} t$.

Note that if x is a variable, $x \not\succ_{ac} t$ for any t. Also, $x \succeq_{ac} a$, given that a is the smallest constant.

The comparison $\succ_c$ of candidates used above is defined as follows. Let $c_s = \langle A_s, C_s \rangle \in cands(s, f)$ and $c_t = \langle A_t, C_t \rangle \in cands(t, f)$. $c_s \succ_c c_t$ if $A_s \succeq_{ac}{}^{mul} A_t$ and and one of the following holds.

1. $A_s \succ_{ac}^{mul} A_t$. Or,
2. $A_s \succeq_{ac}{}^{mul} A_t$, and $C_s \succ C_t$ as defined below:
   $(C_s - C_t) \neq \phi$ and for every pair $\langle A, tt \rangle \in C_t - C_s$, there is a pair $\langle B, ss \rangle \in C_s - C_t$ such that
   (a) $abs(A_s - B, f) > abs(A_t - A, f)$, or
   (b) $abs(A_s - B, f) \geq abs(A_t - A, f)$ and $ss \succ_{ac} tt$

The function $abs(M, f)$ where M is a multiset of f-blocked terms is a suitable *abstraction* of the contribution of M to the Argument view. This abstraction can be defined in different ways yielding different orderings provided that the orderings on abstractions satisfy properties such as *irreflexivity*, *transitivity*, *monotonicity*, *stability*, and *totality* on ground cases. These properties will be mentioned as needed in the proofs.

3.3.1 Abstractions for Contexts Different *abs*'s are motivated and illustrated using the comparison of $s = f(i(f(a,a)), i(f(b,b)), i(f(c,c)))$ with $t = f(i(f(a,b,b)), i(f(a,c,c)))$. Recall from previous section that any RPO-compatible simplification ordering must make $s \succ_{ac} t$.

There is only one candidate for each side—$c_s = \langle A_s, C_s \rangle$ and $c_t = \langle A_t, C_t \rangle$ where $A_s = A_t = \{a, a, a, a, a, a\}$. But,

$$C_s - C_t = \{\langle \{a,a\}, i(f(a,a)) \rangle, \langle \{a,a\}, i(f(b,b)) \rangle, \langle \{a,a\}, i(f(c,c)) \rangle\},$$

whereas

$$C_t - C_s = \{\langle \{a,a,a\}, i(f(a,b,b)) \rangle, \langle \{a,a,a\}, i(f(a,c,c)) \rangle\}.$$

For defining candidate comparison, we compare the contexts of the elevated subterms by comparing the measures of the multiset differences $A_s - B$ and $A_t - A$ above.

3.3.1.1 fcount One way to define a measure of the contribution of a subterm is by using $fcount$ on f-blocked terms as defined below.

Definition 18.

$$fcount(t,f) = \begin{cases} x \text{ if } t = x, \text{ a variable,} \\ 1 \text{ if } top(t) = g \succ f, \\ 1 \text{ if } t \text{ is a Small term w.r.t. } f. \end{cases}$$

For a multiset A of f-blocked terms,

$$fcount(A,f) = \sum_{t_i \in A} fcount(t_i, f).$$

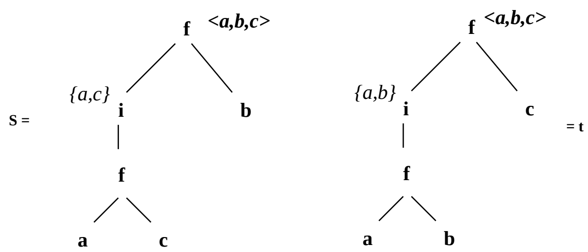
For example, $fcount(\{g(x), x, a, y, x\}, f) = 2 * x + y + 2$; $fcount(\phi, f) = 0$.

The measure for any non-variable f-simple term is 1. A variable's measure depends on what is substituted for it. If $\sigma(x)$ has no elevatable f-subterm, then $\sigma(x)$ contributes 1. Otherwise, it may contribute more due to elevation and flattening. So, the variables in the measure only take values > 0 (i.e. $x, y \geq 1$ above). Given two measures, which are linear polynomials p and q, we compare them by checking if $p - q > 0$ for all non-zero positive integers for the variables. For example, $(3x + 4) > (2x + 3)$, and $(x + 1) \geq 2$, but $(x + 1) \not> 2$.

3.3.1.2 Term Another abstraction can be defined by considering f-blocked terms themselves instead of abstracting out how many f arguments they contribute. So $abs(M, f)$ is defined as $f(M)$ which is compared using the same underlying ordering since terms serving as indices are smaller in size. This leads to a different ordering since $fcount$ does not distinguish between the contributions of f-blocked terms such as $g(a)$ and $g(c)$ (both have $fcount$ of 1).

An example which distinguishes this abstraction from $fcount$ is to compare $s = f(i(f(g(a), g(b))), g(c))$ with $t = f(i(f(g(a), g(c))), g(b))$. Using $fcount$, $t \succ_{ac} s$, but using the term abstraction, $s \succ_{ac} t$.

In contrast to the above abstraction functions, in [7], the context in a candidate is a multiset of pairs, but the first component in a pair is the term obtained after deleting the subterm being elevated. For the example below, $t \succ s$, since



$c \succ b$, whereas using $fcount$ or $term$ abstractions, $s \succ t$ since $i(f(a, c)) \succ i(f(a, b))$.

4.3 Examples

We will use the $fcount$ abstraction in all the examples below. We assume $f \in F_{AC}$ and $g > f > i > c > b > a$.

Example 7. $s = f(i(b), b) \succ_{ac} i(f(b, b)) = t$ because $f \succ i$, and $f(i(b), b) \succ_{ac} f(b, b)$ due to the Context component using $\langle\{a\}, i(b)\rangle \succ \langle\{a\}, b\rangle$.

Example 8. $s = f(i(b), b, i(b), b) \succ_{ac} f(i(f(b, b)), i(f(b, b))) = t$
Both sides have only one candidate each with Arguments component $A_s = A_t = \{a, a, a, a\}$, but $C_s - C_t = \{\langle\{a\}, i(b)\rangle, \langle\{a\}, i(b)\rangle\}$ whereas
$C_t - C_s = \{\langle\{a\}, b\rangle, \langle\{a, a\}, i(f(b, b))\rangle, \langle\{a\}, b\rangle, \langle\{a, a\}, i(f(b, b))\rangle\}$.
The pair $\langle\{a\}, i(b)\rangle \succ \langle\{a, a\}, i(f(b, b))\rangle$ because $abs(A_s - \{a\}) = 3 > 2 = abs(A_t - \{a, a\})$. Hence $C_s > C_t$.

Example 9. $s = f(i(f(a, c)), b) \succ_{ac} f(i(f(a, b)), c) = t$. Arguments are the same, but $C_s - C_t = \{\langle\{a, a\}, i(f(a, c))\rangle\}$ and $C_t - C_s = \{\langle\{a, a\}, i(f(a, b))\rangle\}$. Since $i(f(a, c)) \succ_{ac} i(f(a, b))$, we have $C_s \succ C_t$.

Example 10. $s = f(x, x, x) \succ_{ac} f(a, b) = t$. Since $\{x, x, x\} \succ_{ac} \{a, a\}$.

Example 11. $s = f(a, i(x)) \succ_{ac} f(a, b) = t$. Using $x \geq a$, $A_s = \{a, x\} \geq \{a, a\} = A_t$. We also have $\langle\{x\}, i(x)\rangle \succ \langle\{a\}, b\rangle$ since $abs(A_s - \{x\}) = 1 = abs(A_t - \{a\})$ and $i(x) \succ_{ac} b$.

It is quite easy to verify that with $* \succ +$, $x * (y + z) \succ_{ac} (x * z) + (y * z)$. Further, $s = x * y * (u + v) \succ_{ac} x * ((y * u) + (y * v))$ with $* \succ +$.

5 Proofs

We prove the properties of $\succ_{ac}$ in the following order: irreflexivity, subterm property, transitivity, replacement, totality on ground terms. In the next section, we prove that $\succ_{ac}$ is stable under substitutions. Most proofs are done using induction on the size of the term, or the sum of sizes of terms being compared. Except for the replacement and stability properties, the proofs are similar to proofs for RPO [4].

Property 19 (Irreflexivity). For every term t, we have $t \not\succ_{ac} t$.

Proof. By induction on $|t|$. If $t = f(t_1, \ldots, t_n)$, then if $f \notin F_{AC}$, by induction since $\{t_1, \ldots, t_n\} \not\succ \{t_1, \ldots, t_n\}$, we have the desired result. If $f \in F_{AC}$, we have similarly $cands(t, f) \not\succ_c^{mul} cands(t, f)$.

Property 20 (Subterm). If $s \succeq_{ac} t$, then $s \succ_{ac} t_j$ for any proper subterm t_j of t and also $h(\ldots s \ldots) \succ_{ac} t$ for any proper superterm of s.

Proof. As in [4], we can prove these two simultaneously by induction on $\langle |t|, |s| \rangle$ and case analysis.

Property 21 (Transitivity). Given $s = f(s_1, \ldots, s_n), t = g(t_1, \ldots, t_m)$ and $u = h(u_1, \ldots, u_k)$, if $s \succ_{ac} t$ and $t \succ_{ac} u$, then $s \succ_{ac} u$.

Proof. By induction on $|s| + |t| + |u|$. All cases except $f = g = h \in F_{AC}$ are very similar to [4]. In this case also (i.e., $f = g = h \in F_{AC}$), since all terms in any component of $cands(s, f), cands(t, f), cands(u, f)$ are strictly smaller in size than s, t, u, respectively, and since the comparison of abstraction function $(abs(M, f))$ is also transitive, we have the desired result by induction.

To show the replacement property, we first prove a useful lemma.

Lemma 1 (Candidates Property). *$s \succ_{ac} t$ and $f \in F_{AC}$ implies $cands(s, f) \succ_c^{mul} cands(t, f)$.*

Proof. By induction on $\langle |t|, |s| \rangle$. The base case when t is a constant or a variable is simple. Otherwise, let $s = g(s_1, \ldots, s_m)$ and $t = h(t_1, \ldots, t_n)$. We now do case analysis on the relation between g and f.

1. $g \succ f$: In this case s has only one candidate $c_s = \langle \{s\}, \phi \rangle$. In any candidate $c_t = \langle A_t, C_t \rangle \in cands(t, f)$, we have by Property 14, $\{t\} \geq A_t$. By transitivity, we have $\{s\} \succ_{ac}^{mul} A_t$. Thus $c_s \succ_c c_t$ for any c_t which means $cands(s, f) \succ_c^{mul} cands(t, f)$.
2. $g = f$: Consider the cases for h.
   (a) $h \succ f$: Since $s \succ_{ac} t$ and $h \succ f = g$, for some i, either $s_i \succ_{ac} t$ in which case, by induction, $cands(s_i, f) \succ_c^{mul} cands(t, f)$, or $s_i \sim t$ in which case by Property 15, $cands(s_i, f) = cands(t, f)$.
   Since $top(s) = f$, for every $\langle A_{s_i}, C_{s_i} \rangle \in cands(s_i, f)$, there is an $\langle A_s, C_s \rangle \in cands(s, f)$ with $A_{s_i} \subset A_s$ and $C_{s_i} \subset C_s$, by the definition of candidate construction. Hence, $cands(s, f) \succ_c^{mul} cands(t, f)$.
   (b) $h = f$: In this case since $f = h = g$ and $s \succ_{ac} t$, by the definition of $\succ_{ac}$, $cands(s, f) \succ_c^{mul} cands(t, f)$.
   (c) $f \succ h$: If $t = h(t_1, \ldots, t_n)$ is a Small Term, then the only candidate c_t of t is $\langle \{a\}, \{\langle \{a\}, t \rangle\} \rangle$. Since $top(s) = f$, in any candidate $c_s = \langle A_s, C_s \rangle \in cands(s, f)$ we must have $|A_s| \geq 2$ and hence $A_s \succ_{ac}^{mul} \{a\}$. So, $c_s \succ_c c_t$.
   Otherwise if t is not a Small Term, t must have at least one proper elevatable subterm t'. Since $s \succ_{ac} t$, $s \succ_{ac} t'$ for each subterm t' of t using the subterm and transitivity properties. By induction, $cands(s, f) \succ_c^{mul} cands(t', f)$ for every elevatable subterm t' of t, that is, for every $c_{t'} = \langle A', C' \rangle \in cands(t', f)$ there is a $c_s = \langle A_s, C_s \rangle$ with $c_s \succ_c c_{t'}$.
   Also, by the definition of $cands(t, f)$, since $f \succ top(t)$, every candidate c_t of t can be written as $c_t = \langle A', C' \cup \{\langle A', t \rangle\} \rangle$ where $\langle A', C' \rangle \in cands(t', f)$, meaning there is one extra pair $\langle A', t \rangle$ in C_t than in $C_{t'}$. We must show that c_s can take care of this pair also.
   Consider the cases used to show $c_s \succ_c c_{t'}$. If $A_s \succ_{ac}^{mul} A'$ then we also have that $c_s \succ_c c_t$. Otherwise, we must have $A_s \succeq_{ac}^{mul} A'$ and $C_s \succ C'$. We must show that there is at least one $\langle B, ss \rangle \in C_s - C'$ that can take care of $\langle A', t \rangle$ also as follows. Since $top(s) = f$, by Property 14, $B \subset A_s$. Hence $abs((A_s - B) \neq \phi) > abs((A' - A') = \phi)$. This implies $C_s \succ C_t$ also. Hence $c_s \succ_c c_t$ also and $cands(s, f) \succ_c^{mul} cands(t, f)$.

3. $f \succ g$: Consider the cases for h.
   (a) $h \succ f$ or $h = f$: Since $top(t) \succ top(s)$ and $s \succ_{ac} t$, by the definition of $\succ_{ac}$, either there is some $s_i \succ_{ac} t$ (then by induction, $cands(s_i, f) \succ_c^{mul} cands(t, f)$) or $s_i \sim t$ (in which case $cands(s_i, f) = cands(t, f)$). Since $f \succ top(s)$, for every candidate $\langle A_{s_i}, C_{s_i} \rangle \in cands(s_i, f)$, we have $\langle A_{s_i}, C_{s_i} \cup \langle A_{s_i}, s \rangle \rangle \in cands(s, f)$. So, in the either case above, $cands(s, f) \succ_c^{mul} cands(t, f)$.
   (b) $f \succ h$: If $t = h(t_1, \ldots, t_n)$ is a Small Term, then the only candidate c_t of t is $\langle \{a\}, \{\langle \{a\}, t \rangle\} \rangle$. If s is also small then its candidate $c_s = \langle \{a\}, \{\langle \{a\}, s \rangle\} \rangle$ is clearly bigger than c_t. Otherwise, let s have an elevatable subterm s' and let $c_s = \langle A_s, C_s \rangle$ be the candidate obtained by elevating s' from s. If s' is Big or $top(s') = f$, then $A_s \succ_{ac}^{mul} \{a\}$ and hence $c_s \succ_c c_t$. Finally, if $s' = x$, a variable, then $A_s = \{x\} \succeq_{ac}{}^{mul} \{a\} = A_t$ and $\langle \{x\}, s \rangle \succ \langle \{a\}, t \rangle$ since $abs(\{x\} - \{x\}) = abs(\{a\} - \{a\})$ and $s \succ_{ac} t$.
   Otherwise if t is not a Small Term, t must have at least one proper elevatable subterm t'. Since $s \succ_{ac} t$, $s \succ_{ac} t'$ for each subterm t' of t using the subterm and transitivity properties. For every elevatable subterm t' of t, by induction, $cands(s, f) \succ_c^{mul} cands(t', f)$, i.e., for every $c_{t'} = \langle A', C' \rangle \in cands(t', f)$ there is a $c_s = \langle A_s, C_s \rangle$ with $c_s \succ_c c_{t'}$.
   By the definition of $cands(t, f)$, every candidate c_t of t can be written as $c_t = \langle A_t, C_t \rangle = \langle A', C' \cup \{\langle A', t \rangle\} \rangle$ where $c_{t'} = \langle A', C' \rangle \in cands(t', f)$ for some elevatable t'. As before, there is only one extra pair in the context. Since $cands(s, f) \succ_c^{mul} cands(t', f)$, there is some $c_s = \langle A_s, C_s \rangle \in cands(s, f)$ with $c_s \succ_c c_{t'}$.
   If $A_S \succ^{mul} A'$ then $c_s \succ_c c_t$. Otherwise, $A_s \geq A'$ and $C_s \succ C'$. Since $f \succ g = top(s)$, $\langle A_s, s \rangle$ is in C_s. Also, this tuple cannot be in C' since $s \succ_{ac} t$ and s, therefore, cannot be a subterm of t. Also $\langle A_s, s \rangle \succ \langle A_t, t \rangle$ since $s \succ_{ac} t$.
   Thus $cands(s, f) \succ_c^{mul} cands(t, f)$ in this case also.

Using the above lemma, it is easy show the replacement property.

Property 22 (Replacement). $s \succ_{ac} t$ and $f \in F$ implies $s' = f(\ldots, s, \ldots) \succ_{ac} f(\ldots, t, \ldots) = t'$.

Proof. If $f \notin F_{AC}$, then this follows directly from the multiset comparison.

If $f \in F_{AC}$, then by the previous lemma $cands(s, f) \succ_c^{mul} cands(t, f)$. Since every argument in s' and t' other than s and t is the same, and $cands(s', f)$ (similarly $cands(t', f)$) is simply a component-wise summation of contributions from each argument, we have that $cands(s', f) \succ_c^{mul} cands(t', f)$ and hence $s' \succ_{ac} t'$.

Monotonicity is a direct corollary of the replacement property.

Property 23 (Ground Totality). Let $s = f(S) = f(s_1, \ldots, s_n)$ and $t = g(T) = g(t_1, \ldots, t_m)$ be ground terms, and $\succ$ be any total precedence relation on F. Then s and t are comparable. That is, $s \sim t$, or $s \succ_{ac} t$, or $t \succ_{ac} s$.

Proof. By induction on $|s| + |t|$. The basis case is when s and t are constant symbols, and this follows from the totality of $\succ$. Further, when $f \neq g$ or f, g

are not in F_{AC}, the proof that s and t are comparable, is the same as in the proof of totality of RPO on ground terms.

When, $f = g \in F_{AC}$, then by Property 16, $cands(s,f) \neq_{ac} cands(t,f)$. Since terms in a candidate of s (t) are strictly smaller in size than s (t, respectively), and abs used to compare the first component in the pairs in contexts is total for ground terms, s and t are comparable in this case also.

From the ground totality, we have the following property used later in proving that $\succ_{ac}$ is *stable* under substitutions.

Property 24 (Maximum Candidate). Let s be a ground term. Then s has a maximum candidate, i.e. there is $c_m \in cands(s,f)$ such that $c_m \succ_c c_s$ for any $c_s \in cands(s,f)$ with $c_s \neq_{ac} c_m$.

6 Stability of $\succ_{ac}$

We first analyze the effect of application of a ground substitution σ to a term s and the relationship between candidates of s and $s\sigma$ using an example. We show how the biggest candidate from a ground substitution of each variable can be used to construct the biggest candidate for $\sigma(s)$. To prove that the ordering $\succ_{ac}$ is preserved under a ground substitution σ, we compare $\sigma(s)$ with $\sigma(t)$ by comparing their biggest candidates constructed using the biggest candidates of substitutions. Later, we sketch a proof of stability of $\succ_{ac}$ for an arbitrary substitution.

Consider the term $s = i(f(x,x)$ with $f \in F_{AC}$ and the ground substitution $\sigma = \{x \mapsto gt\}$ where $gt = j(g(a), f(b,c))$.

The term s has only one candidate $\langle\{x,x\},\{\langle\{x,x\}, i(f(x,x))\rangle\}\rangle$. The term gt has two candidates $c^1_{gt} = \langle\{g(a)\},\{\langle\{g(a)\}, j(g(a), f(b,c))\rangle\}\rangle$ and $c^2_{gt} = \langle\{a,a\},\{\langle\{a\},b\rangle,\langle\{a\},c\rangle,\langle\{a,a\}, j(g(a),f(b,c))\rangle\}\rangle$. Note that $c^1_{gt} \succ_c c^2_{gt}$.

$s\sigma = i(f(j(g(a),f(b,c)), j(g(a),f(b,c))))$ has the following four candidates (only 3 are distinct).

$c_1 = \langle\{g(a),g(a)\},\ \{\langle\{g(a)\}, j(g(a),f(b,c))\rangle, \langle\{g(a)\}, j(g(a),f(b,c))\rangle,$
$\qquad \langle\{g(a),g(a)\}, i(f(j(g(a),f(b,c)), j(g(a),f(b,c))))\rangle\}\rangle$
$c_2 = \langle\{g(a),a,a\},\{\langle\{g(a)\}, j(g(a),f(b,c))\rangle, \langle\{a\},b\rangle,\langle\{a\},c\rangle,\langle\{a,a\}, j(g(a),f(b,c))\rangle,$
$\qquad \langle\{g(a),a,a\}, i(f(j(g(a),f(b,c)), j(g(a),f(b,c))))\rangle\}\rangle$
$c_3 = \langle\{a,a,g(a)\},\{\langle\{g(a)\}, j(g(a),f(b,c))\rangle, \langle\{a\},b\rangle,\langle\{a\},c\rangle,\langle\{a,a\}, j(g(a),f(b,c))\rangle,$
$\qquad \langle\{a,a,g(a)\}, i(f(j(g(a),f(b,c)), j(g(a),f(b,c))))\rangle\}\rangle$
$c_4 = \langle\{a,a,a,a\},\ \{\langle\{a,a\}, j(g(a),f(b,c))\rangle, \langle\{a\},b\rangle,\langle\{a\},c\rangle,\langle\{a\},b\rangle,\langle\{a\},c\rangle,$
$\qquad \langle\{a,a\}, j(g(a),f(b,c))\rangle, \langle\{a,a,a,a\}, i(f(j(g(a),f(b,c)), j(g(a),f(b,c))))\rangle\}\rangle$

Of these four, we say that c_1 and c_4 are obtained by *uniform replacement* since they always use the same candidate of gt when handling different occurrences of x, while c_2 and c_3 have been obtained by non-uniform replacement. Note also that c_1 is the biggest candidate of $s\sigma$, and c_1 uses the biggest candidate c^1_{gt} of the term substituted for x. We formalize this notion below.

Definition 25 (Parent Candidate). Let s be any term, $f \in F_{AC}$, σ be any ground substitution, and $c_{s\sigma} = \langle A_{s\sigma}, C_{s\sigma}\rangle \in cands(s\sigma, f)$. Let $\mathcal{P}_{s\sigma}$ be the positions in $s\sigma$ used (where elevations are done) in the derivation of $c_{s\sigma}$. Let $\mathcal{P}_s$ be the restriction of $\mathcal{P}_{s\sigma}$ to positions in s (i.e. omit any positions not in s). We call $c_s = \langle A_s, C_s \rangle$ derived using $\mathcal{P}_s$ as the *parent candidate* of $c_{s\sigma}$ in s.

Definition 26 (Uniform Replacement). Let $f \in F_{AC}$ and A be any multiset of f-blocked terms, i.e., A has only variables, small terms, and terms with top symbol bigger than f. Let $\sigma = \{x_i \mapsto gt_i\}$ be a ground substitution. Let $c_{gt_i} = \langle A_{gt_i}, C_{gt_i} \rangle \in cands(gt_i, f)$. The multiset A_σ is obtained by uniform replacement by candidates (denoted $A_\sigma = urep(A, \{x_i \mapsto A_{gt_i}\})$) if

$$A_\sigma = \bigcup_{s \in A} \begin{cases} A_{gt_i} & \text{if } s = x_i \text{ for some } i \\ \{s\sigma\} & \text{otherwise} \end{cases}$$

Lemma 2 (Argument Substitution). *Let $A_\sigma = urep(A, \{x_i \mapsto A_{gt_i}\})$ where A, f, σ as in the definition above of uniform replacement. Then $(A)\sigma \geq A_\sigma$ where $(A)\sigma = \bigcup_{s \in A} s\sigma$.*

Proof. Follows from Property 14 that $\{gt_i\} \geq A_{gt_i}$.

Lemma 3 (Uniform Replacement Candidate). *Let s be a term, $f \in F_{AC}$, and $\sigma = \{x_i \mapsto gt_i\}$ be a ground substitution. Let $c_s = \langle A_s, C_s \rangle \in cands(s, f)$ and $c_{gt_i} = \langle A_{gt_i}, C_{gt_i} \rangle \in cands(gt_i, f)$. Let $A_{s\sigma} = urep(A_s, \{x_i \mapsto A_{gt_i}\})$ and*

$$C_{s\sigma} = \bigcup_{\langle B, ss \rangle \in C_s} \{\langle urep(B, \{x_i \mapsto A_{gt_i}\}), ss\sigma \rangle\}$$

Then $c_{s\sigma} = \langle A_{s\sigma}, C_{s\sigma} \rangle$ (denoted $c_{s\sigma} = urep(c_s, \{x_i \mapsto A_{gt_i}\})$) is a candidate of $s\sigma$.

Proof. Since every $ss \in A_s$ is at some elevatable position λ in s, in $s\sigma$ also, λ is an elevatable position. Because of the way candidates are defined, the above replacement of each variable by the candidate derived from its substitution in σ, the above construction gives a valid candidate for $s\sigma$.

In the example at the beginning of this section, candidates c_1, c_4 are obtained by uniform replacement.

Definition 27 (Maximum Replacement). Let $\sigma = \{x_i \mapsto gt_i\}$ and $c_{gt_i} = \langle A_{gt_i}, C_{gt_i} \rangle = max(cands(gt_i, f))$, the maximum candidate of gt_i. Then, $mrep(c_s, \sigma) = urep(c_s, \{x_i \mapsto A_{gt_i}\})$. Similarly, if A is any multiset of f-blocked terms, then $mrep(A, \sigma) = urep(A, \{x_i \mapsto A_{gt_i}\})$.

Lemma 4 (Biggest candidate). *Let $s\sigma$ be any ground instance of s using some ground substitution $\sigma = \{x_i \mapsto gt_i\}$. Let $c_{s\sigma} = \langle A_{s\sigma}, C_{s\sigma} \rangle = max(cands(s\sigma, f))$. Let the parent candidate of $c_{s\sigma}$ be c_s. Let $c_m = \langle A_m, C_m \rangle = mrep(c_s, \sigma)$. Then $c_{s\sigma} = c_m$.*

Proof. By contradiction. Assume $c_{s\sigma} \neq c_m$. This implies that there is at least one elevatable position λ in s with $s/\lambda = x_j$ a variable and the contribution of $s\sigma/\lambda = gt_j$ to $c_{s\sigma}$ and c_m are different.

Let $\langle A^m_{gt_j}, C^m_{gt_j}\rangle$ be the maximum candidate of gt_j. By definition of *mrep* this is used to contribute to c_m. Let some other (i.e. smaller) candidate $\langle A_{gt_j}, C_{gt_j}\rangle$ of gt_j be used to contribute to $c_{s\sigma}$. We can now construct a candidate $c'_{s\sigma}$ of $s\sigma$ with $c'_{s\sigma} \succ_c c_{s\sigma}$ by keeping all other things the same, but only at position λ using $\langle A^m_{gt_j}, C^m_{gt_j}\rangle$ instead of $\langle A_{gt_j}, C_{gt_j}\rangle$.

This contradicts that $c_{s\sigma} = max(cands(s\sigma, f))$.

Property 28 (Abstraction Property 1). Let $f \in F_{AC}$, A, B be multisets of f-blocked terms, and σ any ground substitution. Then $fcount(A, f) > fcount(B, f)$ implies $fcount(mrep(A, \sigma), f) > fcount(mrep(B, \sigma), f)$.

Proof. $fcount(A, f) > fcount(B, f)$ implies that $Vars(B) \subseteq Vars(A)$, since all non-variable terms have an $fcount$ of 1. Also, in *mrep*, any non-variable term t is replaced by $t\sigma$ which is also a non-variable with $fcount$ of 1. (Note that this property is also preserved by the second abstraction $abs(A) = f(A)$ since $s \succ_{ac} t$ implies $Vars(s) \subseteq Vars(t)$.)

Property 29 (Abstraction Property 2). Let $f \in F_{AC}$, A, B be multisets of f-blocked terms, and σ any ground substitution. Then $(fcount(A, f) \geq fcount(B, f))$ implies $fcount(mrep(A, \sigma), f) \geq fcount(mrep(B, \sigma), f)$.

Proof. Similar to the previous property.

Lemma 5 (Ground Stability Property). *Let s, t be any terms with $s \succ_{ac} t$. Let $\sigma = \{x_i \mapsto gt_i\}$ be any ground substitution such that $s\sigma$ and $t\sigma$ are ground. Then $s\sigma \succ_{ac} t\sigma$.*

Proof. We use induction on $\langle |t|, |s|\rangle$. The base case when t is a variable or a constant is easy. So, let $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_m)$. Consider the cases used to show $s \succ_{ac} t$. In all cases except $f = g \in F_{AC}$, a straightforward use of the inductive hypothesis leads to a proof.

Consider the case of $f = g \in F_{AC}$. Let $c_{t\sigma}$ be the biggest candidate of $t\sigma$. Let $c_t = \langle A_t, C_t\rangle$ be the parent candidate of $c_{t\sigma}$. Then, $c_{t\sigma} = \langle A_{t\sigma}, C_{t\sigma}\rangle = mrep(c_t, \sigma)$. Since $s \succ_{ac} t$, we must have a candidate $c_s = \langle A_s, C_s\rangle$ of s with $c_s \succ c_t$. Let $c_{s\sigma} = \langle A_{s\sigma}, C_{s\sigma}\rangle = mrep(c_s, \sigma)$ be the biggest candidate of $s\sigma$. We prove $c_{s\sigma} \succ_c c_{t\sigma}$ by considering the cases used to show $c_s \succ_c c_t$.

1. $A_s \succ_{ac} A_t$: Without loss of generality, we assume $A_s \cap A_t = \phi$ for common terms make the same contributions to $A_{s\sigma}$ and $A_{t\sigma}$ which are obtained by maximum replacement in A_s and A_t respectively.
   Consider $tt \in A_t - A_s$. Since $A_s \succ A_t$, we must have $ss \in A_s - A_t$ with $ss \succ_{ac} tt$. Also, ss is not a variable since $x \not\succ t$ for any t. Hence $ss\sigma \in A_{s\sigma}$. By induction, $ss\sigma \succ_{ac} tt\sigma$. If tt is not a variable then $tt\sigma$ is itself the contribution of tt to $A_{t\sigma} = mrep(A_t, \sigma)$. If tt is a variable x_i, then also $\{tt\sigma\} = \{gt_i\} \geq A_{gt_i}$. Hence ss makes a bigger contribution to $A_{s\sigma}$ than tt does to $A_{t\sigma}$. Thus $A_{s\sigma} \succ A_{t\sigma}$.

2. $(A_s \succeq_{ac} A_t)$ and $C_s \succ C_t$: As in the previous case, $A_{s\sigma} = mrep(A_s, \sigma) \geq mrep(A_t, \sigma) = A_{t\sigma}$ from $A_s \geq A_t$, since all terms in A_s and A_t are strictly smaller in size than s and t respectively. If $A_{s\sigma} \succ A_{t\sigma}$ we are done. So, let $A_{s\sigma} = A_{t\sigma}$ (since these are multisets of ground terms).

   We now show $C_{s\sigma} \succ C_{t\sigma}$. Note that $C_{s\sigma}$ $(C_{t\sigma})$ is constructed using maximum replacement in C_s (C_t). Let $\langle A, tt\rangle \in C_t - C_s$ and $\langle A, ss\rangle \in C_s - C_t$ be the pair such that $\langle abs(A_s - B), ss\rangle \succ \langle abs(A_t - A), tt\rangle$.

   By the definition of uniform replacement, $C_{s\sigma}$ includes $\langle mrep(B, \sigma), ss\sigma\rangle$ whereas $C_{t\sigma}$ includes $\langle mrep(A, \sigma), tt\sigma\rangle$. By Property 29, $fcount(mrep((A_s - B), \sigma), f) \geq fcount(mrep((A_t - A), \sigma), f)$ since $fcount((A_s - B), f) \geq fcount((A_t - A), f)$. If $fcount(mrep((A_s - B), \sigma), f) > fcount(mrep((A_t - A), \sigma), f)$, then we are done.

   If $fcount(mrep((A_s - B), \sigma), f) = fcount(mrep((A_t - A), \sigma), f)$, then $ss\sigma \succ_{ac} tt\sigma$ by induction since $ss \succ_{ac} tt$.

   So, we have $C_{s\sigma} \succ C_{t\sigma}$.

Since $c_{t\sigma} = mrep(t, \sigma)$ is the biggest candidate of $t\sigma$, and $c_{s\sigma} \succ_c c_{t\sigma}$ we have $s\sigma \succ_{ac} t\sigma$.

Theorem 30 (Stability Property). *Let s, t be any terms with $s \succ_{ac} t$, and $\sigma = \{x_i \mapsto t_i\}$ be any substitution. Then $s\sigma \succ_{ac} t\sigma$.*

Proof. (Sketch)

It is easy to see that any substitution σ can be written as a sequence (composition) of simple substitutions of the form $\sigma = \{x \mapsto y\}$ and $\sigma = \{x \mapsto h(z_1, \cdots, z_k)\}$, where h is a function symbol of arity $k \geq 0$. It thus suffices to prove that $\succ_{ac}$ is preserved under these two kinds of substitutions.

The stability proof for $\sigma = \{x \mapsto y\}$ follows easily by case analysis guided by the definition of $\succ_{ac}$. The proof for $\sigma = \{x \mapsto h(z_1, \cdots, z_k)\}$ is interesting especially for the case when $f \succ h$, where $f \in F_{ac}$, as different occurrences of x in s, t can lead to many different candidates in $s\sigma$ and $t\sigma$; other cases can be easily considered by case analysis.

We consider the case of $f \succ h$. For every candidate of s (and t) in which x appears uniquely in the Arguments component, k candidates are generated in $s\sigma$ (and $t\sigma$), one for each argument z_i. Given $s \succ_{ac} t$, for every (uncommon) candidate of t, there is a bigger candidate of s; then for every corresponding candidate $t\sigma$, it is possible to find a bigger candidate of $s\sigma$. Unlike in the proof of ground stability, the concept of biggest candidate cannot be used since terms with variables are being considered, and we cannot compare a candidate due to the argument z_1, for instance, with a candidate due to the argument z_2 of h. If x occurs many times in the Arguments component of candidates of s and t, even then for every corresponding candidate of $t\sigma$ in which different arguments of $h(z_1, \cdots, z_k)$ may be used for different occurrences of x, a bigger candidate of $s\sigma$ can be suitably constructed.

7 Conclusion

We have discussed an RPO-like scheme for defining a well-founded ordering on AC-equivalent terms. The scheme works for general terms (including non-ground as well as ground terms). If the precedence relation on function symbols is total, then the scheme defines a total well-founded ordering on AC-equivalent ground terms. Distributivity axioms can be oriented in the proper direction by making $x * (y + z) > (x * y) + (x * z)$ when both $*$ and $+$ are AC operators and $* > +$. Orderings defined using this scheme can be easily implemented.

The proposed scheme is general and it simplifies the definitions in our earlier paper [7], even though the basic concepts of candidates and constructions for generating candidates in the two papers are related. In fact, it was our attempt to simplify the definition in [7] and related insight leading to the proposed definition. Three different orderings for the same precedence relation on function symbols are discussed. They all first compare terms by first ignoring symbols smaller than AC operators. In case that is not sufficient to compare terms, subterms with smaller symbols which get elevated and their contexts are compared (in the reverse order). The orderings differ in how these contexts are compared.

To define candidates and for elevation, it is required that every AC function symbol should be comparable with other symbols in the precedence relation. In this sense, the precedence relation on function symbols need not be total, insofar as the requirement to have a total ordering on ground terms is relaxed.

A weakness of the proposed ordering scheme is that unlike in the case of RPO for non-AC ground terms, it is not possible to define a family of orderings by incrementally adding precedence on function symbols. It is unclear how to generalize the concept of a candidate if precedence relation between an AC symbol and other symbols is unknown.

References

1. Bachmair, L., and Plaisted, D.A. (1985): Termination orderings for associative-commutative rewriting systems. *J. Symbolic Computation,* 1, 329-349
2. Ben Cherifa, A., and Lescanne, P. (1987): Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming,* 9, 2, 137-160.
3. Delor, C., Puel, L. (1993): Extension of the associative path ordering to a chain of associative commutative symbols. Proc. of *5th Intl. Conf. on Rewrite Techniques and Applications (RTA-93)*, LNCS, Springer-Verlag, 389-404.
4. Dershowitz, N. (1987): Termination of rewriting. *J. Symbolic Computation,* 3, 69-116.
5. Gnaeding, I., and Lescanne, P. (1986): Proving termination of associative-commutative rewriting systems by rewriting. Proc. of *8th Intl. Conf. on Automated Deduction (CADE-8),* Oxford, LNCS 230 (ed. Siekmann), Springer Verlag, 52-60.
6. Kapur, D., and Sivakumar, G. (1995): Maximal extensions of simplification orderings. Proc. of *15th Conf. on Foundations of Software Technology and Theoretical Computer Science* (ed. Thiagarajan), Bangalore, India, Springer Verlag LNCS 1026, 225-239, Dec. 1995.

7. Kapur, D., and Sivakumar, G. (1997): A total ground path ordering for proving termination of AC-Rewrite systems. Proc. *Rewriting Techniques and Applications, 8th Intl. Conf., RTA-97*, Sitges, Spain, June 1997, Springer LNCS 1231 (ed. H. Comon), 142-156.
8. Kapur, D., and Sivakumar, G.: *A recurive path ordering for proving associative-commutative termination* Technical Report, Department of Computer Science, State University of New York, Albany, NY, May 1998.
9. Kapur, D., Sivakumar, G. and Zhang, H. (1995): A new ordering for proving termination of AC-rewrite systems. *J. Automated Reasoning,* 1995.
10. Kapur, D., and Zhang, H. (1995): An overview of Rewrite Rule Laboratory (RRL). *J. Computer and Mathematics with Applications,* 29, 2, 91-114.
11. Lankford, D.S. (1979): On proving term rewriting systems are Noetherian. Memo MTP-3, Lousiana State University.
12. Narendran, P., and Rusinowitch, M. (1991): Any ground associative commutative theory has a finite canonical system. In Book, R. (ed.) Proc. of *4th Intl. Conf. on Rewrite Techniques and Applications (RTA-91),* LNCS 488, 423-434.
13. Rubio, A. (1997): A total AC-compatible ordering with RPO scheme. Technical Report, Technical Univ. of Catalonia, Barcelona, Spain.
14. Rubio, A., Nieuwenhuis, R. (1993): A precedence-based total AC-compatible ordering. In Kirchner, C. (ed.) Proc. of *5th Intl. Conf. on Rewrite Techniques and Applications (RTA-93),* LNCS Springer-Verlag, 374-388.
15. Steinbach, J. (1989): Path and decomposition orderings for proving AC-termination. Seki-Report, SR-89-18, University of Kaiserslautern. See also "Improving associative path orderings," in: Proc. of *10th Intl. Conf. on Automated Deduction (CADE-10)*, Kaiserslautern, LNCS 449 (ed. Stickel), 411-425.