Decomposition Ordering
at a Tool to Prove the Termination of Rewriting Systems

Pierre LESCANNE

Centre de Recherche en Informatique de Nancy, CO 140, F54037 Nancy, France
and
Laboratory for Computer Science, Massachusetts Institute of Technology,
545 Technology Square, Cambridge, Massachusetts 01239, USA

Abstract: Decomposition ordering is a well-founded monotonic ordering on terms. Because it has the subterm and the deletion properties, decomposition ordering is useful to prove termination of rewriting systems. An algorithm comparing two terms is given.[1]

1. Introduction

In this paper, a term is a pair formed by an operator and a list of terms. In other words, no restriction exists on the arity of operators. Here are some definitions about ordering on terms. More information can be found in [1] and [4].

- An ordering on terms is *monotonic* if for all operators f

  $s<t$ implies $f(\ldots,s,\ldots)<f(\ldots,t,\ldots)$

- It has the *subterm property* if

  $s<f(\ldots,s,\ldots)$

- It has the *deletion property* if

  $f(\ldots\ \ldots)<f(\ldots,s,\ldots)$

- A monotonic ordering which has the subterm property and the deletion property is a *simplification ordering.*
- An ordering is *well-founded* if there exists no infinite strictly decreasing sequence

  $s_0>s_1>\ldots>s_n>\ldots$

Dershowitz [1] proved that if $\{R_i\rightarrow L_i\}$ is a rewriting system for which exists a simplification ordering $<$ such that for all i and for all ground substitutions σ, $\sigma(L_i)<\sigma(R_i)$, then $\overset{*}{\rightarrow}$ is well-founded, which means no infinite sequence $t_0\rightarrow t_1\rightarrow\ldots\rightarrow t_n\rightarrow\ldots$ exists. The system is said to have the *termination property.* However, Huet and Lankford [5] showed that the termination problem for rewriting systems is undecidable; therefore no uniform method based, for instance, on orderings can exist. But proving termination of particular rewriting system is an essential problem in many areas where rewriting systems are used. In simplification systems this insures that the process will terminate. In the KnuthBendix method, proving termination of the system is necessary, to deduce global confluence from local confluence [7]. For example the system $\{A\rightarrow B,\ A\rightarrow C,\ B\rightarrow A,\ B\rightarrow D\}$ is locally confluent but not globally confluent. If the Knuth-Bendix algorithm is used to transform a set of rewrite rules into a canonical one, Huet [3] showed that uniform termination ordering is necessary for all rules which appear in the process. Several authors propose simplification orderings based on syntactical properties of terms: ordering based on weight by Knuth and Bendix [7], recursive path of subterms ordering by Plaisted [10], recursive path ordering by Dershowitz [1], non-ascending property by Pettorossi [9], generalized recursive path ordering by Kamin and Levy [6]. In this paper a new ordering called the *decomposition ordering* is defined. Its definition is relatively simple, it allows efficient implementation and it has interesting generalizations which are currently being worked upon. As do the other cited orderings, it derives from an ordering on operators. It is based on a decomposition of terms in three parts, on which comparisons are recursively performed. When the basic ordering is total, decomposition ordering is equivalent to the orderings proposed by Plaisted and Dershowitz. But their respective implementations are quite different. A comparative analysis of these methods is proposed in [6] for monadic terms. Here, only decomposition ordering on terms without variables, on a totally ordered set of operators, is given. Such an ordering can be used in simplification systems.

2. Definition of decomposition ordering

Let T(F) be the set of terms on the set F of operators. F is assumed to be totally ordered by $<$. The decomposition ordering is denoted $\prec$.

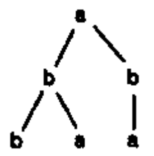
$s=f(s_1,\ldots,s_m)$ is a *left-weighted* term if $s_1,\ldots,s_m$ are left-weighted and $s_1\preceq\ldots\preceq s_m$, where $s\preceq t$ means for left-weighted that $s\prec t$ or s is equal to t.

1 This work was supported by National Science Foundation and Centre National de la Rechmene Scicntifique.

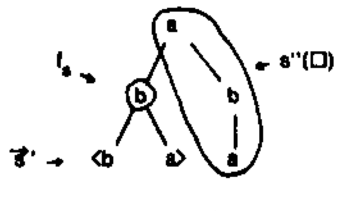
In order to compare two terms u and v, sort the terms from leaf to roots, yielding s and t. If s = t, then u and v are *decomposition equivalent*, which means m = n and there exists a permutation σ on [1..m] such that u_i is decomposition equivalent to $v_{\sigma(i)}$. Otherwise compare s and t using the decomposition ordering method. In this and the following section, assume that all terms are left-weighted.

The *decomposition* of a left-weighted term s is the unique triple $[f_s; \vec{s}'; s''(\square)]$ such that $s = s''(f_s(s_1',\dots,s_p'))$ where $s''(v)$ is the terms given by substituting term v in place of $\square$, where f_s is the first occurrence of the maximal operator in s on its most left branch, $\vec{s}' = \langle s_1',\dots,s_p'\rangle$ is a sequence of terms and $s''(\square)$ is a term with a unique constant subterm denoted by the symbol $\square$; the set of such terms will be written $T(F;\square)$. It will be assumed that the term $\square$ is less than any other term.

For instance, the following term:



decomposes into $[b;\langle b,a\rangle;a(\square,b(a))]$, as described by the following figure.



Note that $s''(\square)$ is not left-weighted. Except for the first immediate subterm, the term is left-weighted; this is recursively true for each first subterm of this term. Such terms will be called almost-left-weighted.

Definition 1: A term s of $T(F;\square)$ is *almost-left-weighted* iff

$s = \square$

or $s = f(s_1,\dots,s_m)$

and s_i is left-weighted for all $i \geq 2$

and s_1 is almost-left-weighted

and for all i,j $1 < i < j \leq m$ implies $s_i \preceq t_j$

Note that a term is almost-left-weighted if it is left-weighted. The decomposition ordering can be given for almost-left-weighted terms of $T(F;\square)$ as follows:

Definition 2: The *decomposition ordering* on almost-left-weighted terms s and t is defined as:

$s \prec t$ iff

$s = \square$ and $t \neq \square$

or $f_s < f_t$

or $f_s = f_t$ and $\langle s_1',\dots,s_p'\rangle \prec_{lex} \langle t_1',\dots,t_q'\rangle$

or $f_s = f_t$ and $\langle s_1',\dots,s_p'\rangle = \langle t_1',\dots,t_q'\rangle$
and $s''(\square) \prec t''(\square)$

where $\prec_{lex}$ is the lexicographic ordering deduced from $\prec$

3. An implementation of decomposition ordering

Using this definition, it is possible to associate with each term, a term on a heterogeneous algebra with five phyla: **operator** (the operators of F), **triple** (representing the decomposition of left-weighted terms), **list** (representing the list of the decompositions of the almost-left-weighted terms), **almost_triple** (representing the decompositions of almost-left-weighted terms), **almost_list** (representing the list of decompositions, the first element in the list being the decomposition of an almost-left-weighted term). Five heterogeneous operators are given:

t : (**operator**, **list**, **almost_triple**)
→ **triple**

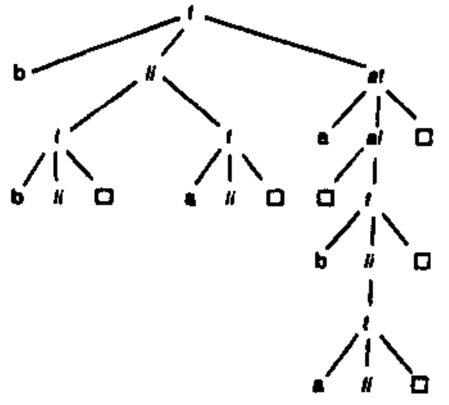
// : (**triple***) → **list**

$\square$: ( ) → **almost_triple**

al : (**operator**, **almost_list**, **almost_triple**)
→ **almost_triple**

al : (**almost_triple**, **triple***)
→ **almost_list**

It must be noticed that the two operators *//* and *al* are associative, which means their profile is not a tuple on phyla but any sequence having a given form: for *//* any sequence on **triple**, and for *al* any sequence beginning by **almost_triple** followed by any sequence on **triple**. The decomposition of the term of the previous example is represented by:



The mapping associating a tree with its decomposition term is one-to-one. To compare two decomposition terms, traverse them in prefix order until the first difference appears. This first difference can occur in one of three cases.

1) The operators are different. Then the leaser operator belongs to the lesser term.

2) In the traversal, one list finished before the other one. Then the shorter list belongs to the lesser term.

3) A subterm with phylum almost J riple corresponds to $\square$ in one decomposition term and corresponds to a term with a root *at* in the other one. Then $\square$ bolongs to the lesser term.

The structure presented is voluntarily redundant for pedagogical reasons. It could be encoded by a linear list containing only relevant information.

4. Conclusion and Perspectives

Decomposition ordering gives an easy way to order a pair of terms. It boils down to comparing two decomposition terms using a simple algorithm studied by Flajolet and Steyaert in their analysis of pattern matching [2]. They consider the Catalan statistics where the average performance of the algorithm io determined over the set of all possible shapes of decomposition terms of given sizes. They show that, before termination, this algorithm performs on the average a constant number of tests for that statistics. In [8] we use a statistics where all terms of T(F) are equiprobabie. By elementary methods, we prove that on monadic terms, the algorithm stops after approximately 1 + 1/v ♦ 1/nv + c testa when one term has length 1, the other has length n and, the set F has v elements and where e is $O(n/v^2)$ as n / →) . It would be interesting to adapt Flajolet and Steyaert's method to this more natural statistics. On the other hand we are currently designing an efficient algorithm to build the decomposition term from term on T(F). Generalizations of decomposition ordering to terms with variables on partially ordered set of operators are possible, but the definitions are more complex [11]. For instance, the concept of "left-weighted" term does not exist anymore and must be extended.

5. References

[1] N. Dershowitz: *Ordering tor Term Rewriting Systems,* Proc. 20th Symposium on Foundations of Computer Science, (1079), 123-131.

[2] P. Flajolet, J.-M. Steyaert: *On the analysis of tree-matching algorithms* 7th ICALP, Lecture Notes in Computer Science 85, pp 208-219.

[3] Q. Muet: *A complete proof of correctness of the Knuth-Bendix Completion algorithm,* Rapport INRIA 25, INRIA Rocquencourt, France.

[4] G. Huet, DC. Op pen: *Equations and Rewrite Rules: a Survey,* in *Formal Languages perspectives and Open Problems,* Ed. Book R.f Academic Press, (1080).

[5] G. Huet, D.S. Lankford: *On the Uniform Halting Problem for Term Rewriting Systems,* Rapport Laboria 283, IRIA, Mars 1078, INRIA Rocquencourt, France.

[6] S. Kamin, J. J. Levy: *Attempts for generalizing the Recursive Path Ordering* (unpublished manuscript 1080).

[7] D.E. Knuth, P. Bendix: *Simple Word Problems in Universal Algebra,* in *Computational Problems in Abstract Algebra,* Ed. Leech J., Pergamon Press, (1070), 263-207.

[8] P. Lescanne: *Two Implementations of Recursive Path Ordering on Monadic Terms,* Centre de Recherche en Informatique de Nancy, Nancy, France, CRIN 81 -R-011.

[0] A. Pettorossi: *Comparing and putting together Recursive Path Orderings, Simplification Ordorings and Non-Ascending Property for termination proofs of Term Rewriting Systems* 8th Int. Coll. Automata Languages and Programming, Haifa Israel, 1881.

[10] D. Plaisted: *A Recursively Defined Ordering for Proving Tormination of Term Rewriting Systems,* Dept of Computer Science Report 78043, U. of Illinois at Urbana-Champaign, Sept. 1078.

[11] F. Reinig: *Decomposition Ordering: Theory and Application,* Thesis in preparation, University of Nancy, France.