On the Recursive Decomposition Ordering with Lexicographical Status and other related Orderings[*]

Pierre LESCANNE
Centre de Recherche en Informatique de Nancy
INRIA-Lorraine
Campus Scientifique, BP 239,
54506 Vandœuvre-les-Nancy, France

Abstract

This paper studies three orderings, useful in theorem proving, especially for proving termination of term rewriting systems: the recursive decomposition ordering with status, the recursive path ordering with status and the closure ordering. It proves the transitivity of the recursive path ordering, the strict inclusion of the recursive path ordering in the recursive decomposition ordering, the totality of the recursive path ordering – therefore of the recursive decomposition ordering –, the strict inclusion of the recursive decomposition ordering in the closure ordering and the stability of the closure ordering by instanciation.

1 Introduction

In this paper, facts about simplification orderings known as the recursive decomposition ordering with status, the recursive path ordering with status and a less known one called the closure ordering are proven. Essentially one aspect of these orderings is studied, namely when the status is lexicographical. There are three reasons for this choice. First the lexicographical status is actually the only useful one according to our experience. We found indeed no example of operations that requires a multiset status and rejects a lexicographical status. Second, the lexicographical status provides an ordering which is total on ground terms when the precedence is total or quasi-total

[*]Part of this work was done while the author was visiting the Institute for New Generation Computer Technology, Tokyo, Japan

(the definition will be given in the paper). Third, no proof of the relation between the recursive path ordering with status and the recursive decomposition ordering with status exists yet, it was only sketched in one of a previous paper of the author [16], by a matter of fact, compared with previous works, proofs can be made easier, because simpler definitions of both the recursive path ordering and the recursive decomposition ordering are used here. The properties that are proven are the transitivity of the recursive path ordering, the strict inclusion of the recursive path ordering in the recursive decomposition ordering, the totality of the recursive path ordering – therefore of the recursive decomposition ordering –, the strict inclusion of the recursive decomposition ordering in the closure ordering and the stability of the closure ordering by instanciation. Some results of this paper can be considered as folk theorems [5], but since their proofs are not available or do not exist we found interesting to provide them. These orderings are really useful for proving termination of term rewriting systems. People who are not familiar with these applications are invited to read [2] or [10].

2 Some definitions

In this paper, we study strict quasi-orderings, i.e., relations that are transitive, $(s < t \;\&\; t < u) \implies s < u$, and irreflexive, $s \not< s$, and we called them orderings for short. In addition, we consider terms in $T(F)$ or in $T(F, X)$, where $F = \bigcup_{n\geq 0} F_n$ is a set of operators and X is a set of variables and we are using some conventions that are really helpful. If $f \in F_n$ is an operator of arity n, we indeed will write indifferently s or $f(s_1, \ldots, s_m)$ or $f\vec{s}$ and, symmetrically, t or $g(t_1, \ldots, t_n)$ or $g\vec{t}$ and u or $h(u_1, \ldots, u_p)$ or $h\vec{u}$, with the convention that $\vec{s}$ represents the sequence $s_1, \ldots, s_m$. Many proofs in this paper will be by induction on the *strict subterm ordering*, written $\lhd$ and the cartesian products of this ordering, written $\lhd \times \lhd$ and $\lhd \times \lhd \times \lhd$. $s \lhd t$ means that s is a strict subterm of t and $(s_1, s_2) \lhd \times \lhd (t_1, t_2)$ means that s_1 is a subterm of t_1 and s_2 is a subterm of t_2 and at least one of the relations is strict. $\lhd \times \lhd \times \lhd$ is defined the same way. These three orderings are noetherian. Let us recall the definition of the *symbol of* t *at occurrence* α written $t(\alpha)$, which is $f(t_1, \ldots, t_n)(\epsilon) = f$ and $f(t_1, .., t_n)(i\alpha) = t_i(\alpha)$. The *subterm* t/α *at the occurrence* α is defined by $t/\epsilon = t$ and $f(t_1, \ldots, t_n)/i\alpha = t_i/\alpha$. In this paper, we need also a new concept, namely the *list of terms below the occurrence* α, written $s \bigtriangledown \alpha$. It is given by

Definition 1 *The* list of terms *below* α *is defined as*

- $f\vec{s} \bigtriangledown \epsilon = \vec{s}$
- $f\vec{s} \bigtriangledown i\alpha = s_i \bigtriangledown \alpha$

In other words if $s/\alpha = f\vec{t}$ then $s \bigtriangledown \alpha = \vec{t}$. We are now defining the concepts of *elementary decomposition,* written *edec, path decomposition*, written *pdec* and *decomposition* of a term, written *dec.*

Definition 2 • *The* elementary decomposition *at the occurrence α is defined as*

$$edec(t, \alpha) = \langle t/\alpha, t \bigtriangledown \alpha \rangle.$$

- *The* precedence elementary decomposition *at the occurrence α is*

$$\langle t(\alpha), t \bigtriangledown \alpha \rangle.$$

- *The* path decomposition *of a term along the path π is defined as*

$$pdec(t, \pi) = \{edec(t, \alpha) | \alpha \le \pi\}.$$

- *The* decomposition *of a term is*

$$dec(t) = \{pdec(t, \pi) | \pi \text{ path in } t\}.$$

In this paper, as in many others, people speak about the recursive path ordering or the recursive decomposition ordering, whereas they should say the recursive path extension or the recursive decomposition extension of a precedence or of an ordering. We will follow the tradition and stick to this abuse of language. In this paper, the status is supposed *lexicographical from left to right* which means that one compares the sequences of terms from left to right. For instance, $\vec{s} <^{lex} \vec{t}$ if either $s_1 < t_1$ or $s_1 = t_1$ and $(s_2, \ldots, s_m) <^{lex} (t_2, \ldots, t_n)$ or $\vec{s}$ is a *strict prefix* of $\vec{t}$, which means $\vec{t} = (s_1, ..., s_m, t_{m+1}, ..., t_n)$. Notice this last part in the definition, which allows considering equivalent operators with different arities. We are going to see later the interest of such a possibility. The recursive decomposition ordering supposes that a quasi-ordering $<$ on terms, the *basic ordering*, is already defined. The equivalence with respect to this quasi-ordering is written $\simeq$. The basic ordering is often founded on a *precedence*, which is a partial ordering on the symbols. Especially, the precedence may specify that different operators are equivalent. Most of the time in current applications, one uses a specific basic ordering, founded on a precedence, called the *root ordering* and written $<_\rho$. The root ordering compares two terms by comparing just their roots according to a specific precedence. In other words, $f\vec{s} <_\rho g\vec{t}$ if and only if $f < g$ and $f\vec{s} \simeq_\rho g\vec{t}$ if and only if $f \simeq g$. If two symbols f and g are not comparable by the precedence we write $f\sharp g$. A key problem in proving termination of term rewriting systems is to provide an ordering which is stable by instanciation. Usually, one requires the basic ordering to fulfill the variable condition, namely $s > x \Rightarrow x \in Var(s)$. With ordering based on precedence the idea is to make the precedences to satisfy the *variable condition.*

Definition 3 *A precedence satisfies the* variable condition *if*

- $(x \in X \,\&\, f \in F) \;\rightarrow\; x\sharp f$
- $(x \in X \,\&\, y \in X \,\&\, x \neq y) \;\rightarrow\; x\sharp y$

Precedence elementary decompositions are convenient and simpler. The definition of the recursive decomposition ordering starts by defining the ordering on elementary decompositions, supposing that the ordering on terms is already recursively defined and extended to a lexicographical ordering on sequences of terms. To compare two elementary decompositions, one compares the first components and if they are equivalent, one compares the second components that contain the sequences of terms. The path decompositions are compared according to the set extension of the ordering on elementary decompositions, it is similar to the multiset ordering, except that it is restricted to sets. The decompositions are compared as multisets of path decompositions and terms are compared by the recursive decomposition ordering through their decompositions.

Definition 4 *The* decomposition ordering *is written $<_d$ and defined as follows.*

- *The* ordering on elementary decompositions $<_{ed}$ *is defined by* $\langle s, \vec{s}\rangle <_{ed} \langle t, \vec{t}\rangle$ *if and only if* $s < t$ *or* $s \simeq t$ *and* $\vec{s} <_d^{lex} \vec{t}$.
- *The* ordering $<_{pd}$ on path decompositions *is defined as the set extension of* $<_{ed}$*, often written* $<_{ed}\!<_{ed}$.
- *The* ordering on decompositions *is defined as the multiset extension* $<_{pd}\!<_{pd}$ *of the ordering* $<_{pd}$.

- *The* recursive decomposition ordering $<_d$ *of two terms s and t is defined as $s <_d t$ if and only if $dec(s) <_{pd}<_{pd} dec(t)$.*

Let us illustrate the recursive decomposition ordering on a rule taken from a classical term rewriting system, later on an ad hoc example will be used to prove an inclusion of orderings. Suppose that the basic ordering is a root-ordering based on the precedence $suc < +$. Then for this ordering $suc(x+y) <_\rho suc(x)+y$. Let us state $s = suc(x)+y$ and $t = suc(x+y)$. The decomposition ordering extends this ordering. Indeed one has $dec(s) = \{pdec(s,11), pdec(s,2)\}$ where

$$pdec(s,11) = \{\langle s, (suc(x), y)\rangle, \langle suc(x), (x)\rangle, \langle x, ()\rangle\}$$

and

$$pdec(s,2) = \{\langle s, (suc(x), y)\rangle, \langle y, ()\rangle\}$$

and $dec(t) = \{pdec(t,11), pdec(t,12)\}$ where

$$pdec(t,11) = \{\langle t, (x+y)\rangle, \langle x+y, (x,y)\rangle, \langle x, ()\rangle\}$$

and

$$pdec(t,12) = \{\langle t, (x+y)\rangle, \langle x+y, (x,y)\rangle, \langle y, ()\rangle\}.$$

$pdec(t,11) <_{pd} pdec(s,11)$ since $\langle t, (x+y)\rangle <_{ed} \langle s, (suc(x), y)\rangle$ by $t <_\rho s$ and $\langle x+y, (x,y)\rangle <_{ed} \langle s, (suc(x), y)\rangle$ by $x+y \simeq_\rho suc(x)+y$ and $(x,y) <_d^{lex} (suc(x), y)$. One may prove similarly that $pdec(t,12) <_{pd} pdec(s,2)$. In this example, one refers to a root-ordering and it is more convenient to use a precedence elementary decomposition and to write, for example $\langle +, (x,y)\rangle$ instead of $\langle x+y, (x,y)\rangle$ since only the root $+$ will be used in comparisons. This will be done whenever a root-ordering is used.

3 The Recursive Decomposition Ordering and the Recursive Path Ordering

Now before stating our main theorem, let us define the *recursive path ordering* [2, 12, 21].

Definition 5 *The* recursive path ordering *with status or the* recursive extension of $<$ *or the* lexicographical semantic path ordering *is defined as follows,*

$$f\vec{s} <_p g\vec{t}$$

iff

$$(ST)\ \ (\forall i \in [1...m]) s_i <_p t$$

and one of the following conditions is fulfilled

1. $s < t$,
2. $s \simeq t \;\&\; \vec{s} <_p^{lex} \vec{t}$,
3. $(\exists j \in [1...n])\ s <_p t_j \ \vee \ s = t_j$.

If the basic ordering satisfies a condition called weak F-stability relative to the recursive path ordering, then the recursive path ordering is F- stable [17]. This property is trivially satisfied when the basic ordering is a root ordering.

Definition 6 *An ordering is F-*stable *if*

$$s < t \Longrightarrow f(\ldots, s, \ldots) < f(\ldots, t, \ldots).$$

An ordering $<$ *is* weakly F-stable *relative to* $<_p$ *if*

$$s <_p t \Longrightarrow f(\ldots, s, \ldots) < f(\ldots, t, \ldots) \text{ or } f(\ldots, s, \ldots) = f(\ldots, t, \ldots).$$

On another hand, the recursive path ordering has the subterm property ST. Then provided one proves it is a strict quasi-ordering, it is a simplification ordering and according to Higman theorem [6], the relation $<_p\ \&\ =$ is a well-quasi ordering, hence the recursive path ordering $<_p$ is well-founded. The irreflexivity $s \not<_p s$ is obvious. Therefore, the proof of transitivity is important. Since it is very simple, it is worth to be given (see [8] for another slightly more complex proof). Next we prove that the recursive path ordering is included in the recursive decomposition ordering. By a counter-example the inclusion will be shown to be strict.

Proposition 1 $<_p$ *is transitive.*

Proof:

Suppose that s, t and u are three terms such that $s <_p t$ and $t <_p u$. Let us prove $s <_p u$, by induction on the ordering $\lhd \times \lhd \times \lhd$. For the (ST) part of the definition, if $s_i <_p t$, then $(s_i, t, u) \lhd \times \lhd \times \lhd (s, t, u)$, therefore by transitivity, $s_i <_p u$. Consider now the several methods for proving $s <_p t$ and $t <_p u$ in the second part of the definition.

- Only 1 and 2 are used, then, by transitivity of $<$, $s < u$ which implies $s <_p u$.
- $t <_p u$ is proved by 3, i.e., $(\exists k \in [1...p])\ t <_p u_k \ \vee \ t = u_k$, then by induction and transitivity $(\exists k \in [1...p]) s <_p u_k$, therefore $s <_p u$.
- $s <_p t$ is proved by 3, i.e., $(\exists j \in [1...n])\ s <_p t_j \ \vee \ s = t_j$, then using the first part of the proof of $t_j <_p u$ and by induction and transitivity, $s <_p u$.

□

Theorem 1 $s <_p t \Rightarrow s <_d t$.

Proof: By induction on $\lhd \times \lhd$. Suppose $s <_p t$ and $(s', t') \lhd \times \lhd (s, t) \Rightarrow [s' <_p t' \Rightarrow s' <_d t']$. Therefore, $(\forall i \in [1..m]) s_i <_p t$, then by induction $s_i <_d t$ and for all $i\pi$ path in s, there exists a path ϕ in t such that $pdec(s_i, \pi) <_d pdec(t, \phi)$.

Notice that $pdec(s, i\pi) = pdec(s_i, \pi) \cup \langle f, \vec{s} \rangle$, that $pdec(t, \phi) \ni \langle g, \vec{t} \rangle$ and that $pdec(s_i, \pi) \not\ni \langle g, \vec{t} \rangle$. Therefore, if $f < g$ or $f = g$ and $\vec{s} <_p^{lex} \vec{t}$ i.e., by induction $\vec{s} <_d^{lex} \vec{t}$, $\langle f, \vec{s} \rangle <_{ed} \langle g, \vec{t} \rangle$ then $pdec(s, i\pi) <_d pdec(t, \phi)$ where $i\pi$ and ϕ are associated as above. Thus, in this case, one concludes that $s <_d t$.

If there exists j such that $s <_p t_j$, by induction $s <_d t_j$ and for each π path in s, there exists a path $j\phi$ in t such that $pdec(s, \pi) <_d pdec(t_j, \phi)$. Since $pdec(t, j\phi) = \langle g, \vec{t} \rangle \cup pdec(t_j, \phi)$, then $pdec(s, \pi) <_d pdec(t, j\phi)$, $dec(s) <_d <_d dec(t)$ and $s <_d t$. □

The implication goes only in one direction, as shown by the following counter-example. Indeed, one may have $s <_d t$ and $s \not<_p t$. Let s be $x \star h(x \star x)$ and t be $h(h(x) \star x)$. One notices that if $h \sharp \star$ and the precedence satisfies the variable condition, then $s \not<_p t$. However, $s <_d t$, because

$$pdec(s, 1) = \{\langle \star, (x, h(x \star x)) \rangle, \langle x, () \rangle\},$$

$$pdec(s, 211) = \{\langle \star, (x, h(x \star x)) \rangle, \langle h, (x \star x) \rangle, \langle \star, (x, x) \rangle, \langle x, () \rangle\},$$

$$pdec(s, 212) = pdec(s, 211)$$

and

$$pdec(t, 111) = \{\langle h, (h(x) \star x) \rangle, \langle \star, (h(x), x) \rangle, \langle h, (x) \rangle, \langle x, () \rangle\},$$

$$pdec(t, 12) = \{\langle h, (h(x) \star x)\rangle, \langle \star, (h(x), x)\rangle, \langle x, ()\rangle\},$$

it's easily seen that $pdec(s, 1) <_{pd} pdec(t, 111)$,$pdec(s, 211) <_{pd} pdec(t, 111)$ and $pdec(s, 212) <_{pd} pdec(t, 111)$. It is interesting to notice that if $\star < h$ or $\star = h$ or $\star > h$ then $s <_p t$, which means in some sense that the precedence is useless and which is translated by the decomposition ordering.

4 Totality on quasi-total precedences

Let us recall that an ordering is total if for all pairs (x, y) one has either $x < y$ or $x = y$ or $x > y$. In many cases, one wants to get an ordering which is total on ground terms. Total precedences give such orderings, but they are not the only ones with this property. There are cases where one may want other precedences, like in group with division (see below). The basic idea is to allow symbols to be equivalent provided they have different arity. In this case, the arity discriminates among terms with equivalent root. Such a precedence is called a quasi-total precedence.

Definition 7 *A precedence is* quasi-total *if for all pairs* (f, g) *one has either* $f < g$ *or* $f \simeq g$ *or* $f > g$ *and* $f \simeq g$ *implies that* f *and* g *have different arities. This last property is called the* arity condition*.*

Usually when we mean that a precedence is total or quasi-total, this does not affect the variables, but only the function symbols. The variables are as usual, pairwise incomparable and incomparable with any other function symbol. Suppose f and g are two symbols such that the arity of f is less than the arity of g. A precedence with $f < g$ gives a recursive path ordering which is different from this given by a precedence with $f \simeq g$. Indeed with $a < b < f < g$ one has $f(b) <_p g(a, b)$ and with $a < b < f \simeq g$ one has $g(a, b) <_p f(b)$. The practical interest of the quasi-totality is illustrated by the following example extracted from a canonical rewriting system for deciding *groups with left division* [15]. The termination of the other non mentioned rules is proved by the subterm property.

$$x\backslash e \to i(x)$$

$$i(x\backslash y) \to y\backslash x$$

$$(x\backslash y)\backslash z \to y\backslash(i(x)\backslash z)$$

The termination of this term rewriting system can only be proved by using a recursive path ordering or a recursive decomposition ordering based on the quasi-total precedence that contains the pair $i \simeq \backslash$. The reader may check that no other choice for i and $\backslash$ works. REVE that implements an automatic method of suggestion gives the same conclusion [4]. In [14] p. 43, Kapur and Sivakumar report how they were faced to this problem. Especially, in this case, one has $y <_p i(x\backslash y)$, $x <_p i(x\backslash y)$. Therefore, $y\backslash x <_p i(x\backslash y)$ comes from $\backslash \simeq i$ and $[y; x] <_p^{lex} [x\backslash y]$. One has $y <_p (x\backslash y)\backslash z$ and $i(x)\backslash z <_p (x\backslash y)\backslash z$. On another hand, $y\backslash(i(x)\backslash z) <_p (x\backslash y)\backslash z$ comes from $\backslash = \backslash$ and $[y; i(x)\backslash z] <_p^{lex} [x\backslash y; z]$.

Theorem 2 *If the precedence is total or quasi-total then the recursive path ordering is total.*

Proof: Let s and t be two terms. By induction on $\lhd \times \lhd$, for all pairs $(s', t') \lhd \times \lhd (s, t)$ one has either $s' <_p t'$ or $s' = t'$ or $s' >_p t'$. This is specifically true if one takes for (s', t') either (s_i, t) or (s, t_j). Therefore, one may distinguish essentially three cases:

- either $(\exists i \in [1...m]) s_i >_p t$ or $s_i = t$, then $s >_p t$,
- or $(\exists j \in [1...n]) s <_p t_j$ or $s = t_j$, then $s <_p t$,
- or $(\forall i \in [1..m]) s_i <_p t$ and $(\forall j \in [1..n]) s >_p t_j$ then
  - if $f < g$, then $s <_p t$,
  - if $f > g$, then $s >_p t$,
  - if $f \simeq g$, then
    - if $\vec{s} <_p^{lex} \vec{t}$, then $s <_p t$,
    - if $\vec{s} >_p^{lex} \vec{t}$, then $s >_p t$,
    - and if $\vec{s} = \vec{t}$ then $f = g$ and $s = t$.

□

Corollary 1 *If the precedence is total or quasi-total, the recursive decomposition ordering is total and therefore coincides with the recursive path ordering.*

Proof: Obvious, from Theorem 1 and Theorem 2. □

The next corollary is interesting in some uses of ordering in theorem proving with equations, where such orderings that can be extended to total orderings on ground terms are necessary [7]. They are called *complete simplification orderings.*

Corollary 2 *Each recursive path ordering or recursive decomposition ordering based on a precedence that satisfies the arity condition can be extended to a total ordering on ground terms.*

Proof: Obvious, since each precedence can be extended into a total or quasi-total one. □

5 The Closure Ordering

We introduce now an ordering which contains strictly both, the recursive path ordering and the recursive decomposition ordering. We feel that, in some sense, it is the largest that can be built based on precedences and the recursive path ordering principle, this is why we call it the *closure ordering.* A variant was already presented by R. Forgaard in [3] (see also [2] p. 97). In Section 2 we have set the variable condition on the precedence to insure stability by substitution. Here, a different attitude is adopted, the idea is to give he variables all the possible positions in a quasi-total precedence.

Definition 8 *The* closure ordering *w.r.t. a precedence $<$ is written $<_c$ and defined as follows.*

$$s <_c t$$

iff for all precedences $\prec$ that contain $<$ and whose restriction to F is quasi-total

$$s \prec_p t.$$

Notice that, according to Corollary 1, it makes no difference to use the recursive decomposition ordering or the recursive path ordering when speaking about quasi-total precedence. A difference between the ordering presented here and the previous versions [2, 3] is that variables can be ordered by the precedence. This gives a stronger ordering; for instance, the examples below could not be ordered by Forgaard's version. Notice also that two variables, say x and y, can be made equivalent in $\prec$, but this has no consequence, since it corresponds to a renaming of x to y. Thus in the definition of $<_c$ it makes no difference to use $\prec_p$ or $\prec_d$.

Proposition 2 $s <_d t \Rightarrow s <_c t$.

Proof: If $s <_d t$ then for any $\prec$ that contains $<$, $s <_d t$ or $s <_p t$, therefore $s <_c t$. □

The opposite implication is false as shown by the counter-example:

$$x \star h(x \star y) <_c h(h(x) \star y)$$

with an empty precedence. It is similar to the counter-example in section 3, in the sense that the former is an instance of the latter. Another counter-example is

$$(x \star y) \star (x \star y) <_c ((x \star x) \star x) \star (y \star y)$$

which is related to one that appears in [11]. The closure ordering is stable by substitution.

Theorem 3 *For each substitution σ,*

$$s <_c t \Rightarrow \sigma(s) <_c \sigma(t).$$

Proof: Suppose given a substitution σ and a precedence $\prec$ which is quasi-total on F and contains $<$. Let us prove that $s <_c t$ implies $\sigma(s) \prec_p \sigma(t)$. We just have to find a precedence, say $\prec'$, that contains $<$ and is such that $s \prec'_c t$ implies $\sigma(s) \prec_p \sigma(t)$. This precedence is an ordering on $Var(t) \cup F$ which is built from the orderings $\prec_p$ and $\prec$ and from the substitution σ as follows.

$$\begin{array}{lcl}
x \prec' y & \Leftrightarrow & \sigma(x) \prec_p \sigma(y) \\
x =' y & \Leftrightarrow & \sigma(x) = \sigma(y) \\
x \prec' g & \Leftrightarrow & (\forall h \in F \cap \sigma(x)) h \prec g \\
f \prec' x & \Leftrightarrow & (\exists h \in F \cap \sigma(x)) f \prec h \vee (f = h \,\&\, f \neq \sigma(x)) \\
f =' x & \Leftrightarrow & f \text{ is a constant and } \sigma(x) = f \\
f \prec' g & \Leftrightarrow & f \prec g.
\end{array}$$

The proof of the (ST) part for $\sigma(s) \prec_p \sigma(t)$ from $s \prec'_c t$ is easy by induction on $\lhd \times \lhd$ as for the proofs of part 2 and 3. The proof of 1 is by case on the root of s and t. It uses the definition of the precedence $\prec'$. For instance, if $s \equiv x$, $t = g\vec{t}$ and $x \prec' g$, then $\sigma(s) \equiv \sigma(x)$, $\sigma(t) \equiv g(\sigma(t_1), \ldots, \sigma(t_n))$ and by definition of $s \prec' g$ one can demonstrate $\sigma(x) \prec \sigma(t)$. $\Box$

6 A hierarchy of orderings

In this section, we recall what we have proved on the relationships between extensions of precedences in the case of a lexicographical status. Let first define an embedding or what Higman [6] calls a *divisibility order*.

Definition 9 *The* embedding *is the least ordering $\hookleftarrow$ that satisfies*

$$s \hookleftarrow t \textit{ implies } f(\ldots, s, \ldots) \hookleftarrow f(\ldots, t, \ldots)$$

and

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

Actually there exists an alternative definition more manageable and that provides trivially the inclusion into the recursive path ordering. Here $\hookleftarrow^{cart}$ is the cartesian product of the ordering $\hookleftarrow$ extended to arbitrary long sequences of terms, it is strictly contained in the lexicographical product.

Proposition 3 *$\hookleftarrow$ is the least ordering that satisfies*

$$f\vec{s} \hookleftarrow g\vec{t}$$

if

$$(ST) \quad (\forall i \in [1...m]) s_i \hookleftarrow t$$

and one of the following conditions is fulfilled

1. *$f \equiv g \;\&\; \vec{s} \hookleftarrow^{cart} \vec{t}$,*
2. *$(\exists j \in [1...n]) \; s \hookleftarrow t_j \;\vee\; s = t_j$.*

Proof: The proof is by induction on $\lhd \times \lhd$ and the transitivity of $\hookleftarrow$ plays an important role. □

By examining its definition, one sees that the recursive path ordering is a fixed point of the previous definition, then it contains obviously the least fixed point, namely the embedding $\hookleftarrow$. Then one gets the following proposition, that summarizes all the results of this paper.

Proposition 4 *Given a precedence $<$ one has the following strict inclusions:*

$$\lhd \subset \hookleftarrow \subset <_p \subset <_d \subset <_c .$$

In the case of a multiset status the second inclusion is due to Dershowitz [1] and means that the orderings higher in the hierarchy are simplification orderings and well-quasi orderings, the third is due to Jouannaud, Lescanne and Reinig [11] and the fourth to Forgaard [3] in a restricted case. A hierarchy of orderings was also presented by Rusinowitch in the case of a multiset status [20]. It contains orderings we didn't try to extend in the case of a lexicographical status, namely orderings due Plaisted [18], to Kapur and Narendran [13] and to himself. However he does not consider the closure ordering.

7 Conclusion

All the orderings of this paper have been implemented except the closure ordering, but this could easily be done as we have mentioned, this only requires to check that the disjunction of the suggestions made by REVE is a tautology, which means it covers all the possible cases. Actually we run the examples this way, in REVE. Other hierarchies could be considered based on extensions of the embedding [19].

References

[1] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[2] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1 & 2):69–116, 1987.

[3] R. Forgaard. *A program for generating and analyzing term rewriting systems.* Technical Report 343, Laboratory for Computer Science, Massachusetts Institute of Technology, 1984. Master's Thesis.

[4] R. Forgaard and D. Detlefs. An incremental algorithm for proving termination of term rewriting systems. In J-P Jouannaud, editor, *Proceedings 1st International Conference on Rewriting Techniques and Applications*, pages 255–270, Springer Verlag, 1985.

[5] D. Harel. On folk theorems. *Communications of the Association for Computing Machinery*, 23(7):379–389, 1980.

[6] G. Higman. Ordering by divisibility in abstract algebra. *Proc. London Math. Soc.*, 3(2), 1952.

[7] J. Hsiang and M. Rusinowitch. On word problem in equational theories. In Th. Ottmann, editor, *Proceedings of 14th International Colloquium on Automata, Languages and Programming, Karlsruhe (West Germany)*, Springer-Verlag, July 1987. Lecture Notes in Computer Science, volume 267.

[8] G. Huet. *Formal Structures for Computation and Deduction.* Technical Report, INRIA, May 1986.

[9] J. P. Jouannaud and P. Lescanne. La réécriture. *Techniques et Sciences Informatiques*, 5(6):433–452, 1987.

[10] J. P. Jouannaud and P. Lescanne. Rewriting systems. *Technology and Sciences of Informatics*, 6(3):180–199, June 1987. Translated from [9].

[11] J. P. Jouannaud, P. Lescanne, and F. Reinig. Recursive decomposition ordering. In Bjørner D., editor, *Formal Description of Programming Concepts 2*, pages 331–348, North Holland, Garmisch-Partenkirchen, RFA, 1982.

[12] S. Kamin and J-J. Lévy. Two generalizations of the recursive path ordering. 1980. Unpublished manuscript.

[13] D. Kapur, P. Narendran, and G. Sivakumar. A path ordering for proving termination of term rewriting systems. In H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, editors, *Proceedings of the 6th Conference on Automata, Algebra and Programming*, Springer-Verlag, 1985.

[14] D. Kapur and G. Sivakumar. Experiments with an architecture of RRL, a rewrite rule laboratory. In *Proceedings of an NSF Workshop on the Rewrite Rule Laboratory*, pages 33–56, 1983.

[15] P. Lescanne. Computer experiments with the REVE term rewriting systems generator. In *Proceedings, 10th ACM Symposium on Principles of Programming Languages*, ACM, 1983.

[16] P. Lescanne. Uniform termination of term rewriting systems. recursive decomposition ordering with status. In B. Courcelle, editor, *Proceedings 9th Colloque les Arbres en Algebre et en Programmation*, pages 182–194, Cambridge University Press, Bordeaux (France), 1984.

[17] J-J. Lévy. Dershowitzeries. 1981. Unpublished manuscript.

[18] D. Plaisted. *A recursively defined ordering for proving termination of term rewriting systems.* Technical Report R-78-943, U. of Illinois, Dept of Computer Science, 1978.

[19] L. Puel. Bon préordres sur les arbres associés à des ensembles inévitables et preuves de terminaison de systèmes de réécriture. Thèse d'Etat, September 1987. Université Paris VII.

[20] M. Rusinowitch. Path of subterms ordering and recursive decomposition ordering revisited. *J. of Symbolic Computation*, 3(1 & 2):117–132, 1987.

[21] K. Sakai. An ordering method for term rewriting systems. In *Proc. First Int. Conf. on Fifth Generation Computer Systems*, Tokyo, Japan, November 1984.