A Termination Ordering for Higher Order Rewrite Systems

Olav Lysne* and Javier Piris**

Laboratoire de Recherche en Informatique
Bat. 490 - 91405 Orsay Cedex
FRANCE

COMMUNICATE TO:
Olav Lysne, Department of informatics, University of Oslo
P.O. box 1080 Blindern, 0316 Oslo, NORWAY
Phone: +47 22 85 24 34, Fax: +47 22 85 24 01
E-mail: Olav.Lysne@ifi.uio.no

Abstract. We present an extension of the recursive path ordering for the purpose of showing termination of higher order rewrite systems. Keeping close to the general path ordering of Dershowitz and Hoot, we demonstrate the necessary properties of the termination functions for our method to apply, thus describe a class of different orderings. We also give a counterexample to a previously published extension of the recursive path ordering into the higher order setting.

1 Introduction

During the last two decades a lot of work has been done on techniques for first order rewriting. There are still many open problems for the first order case, but even so we have in the last few years seen an increasing interest in extending the obtained results to a higher-order framework. This is partly because the higher order logics offer increased expressive power, and partly due to a wish to extend the applicative programming paradigm by adding a rewriting relation to β-reduction. The focus of this research is on the combination of λ-terms and its formal theory, the typed lambda calculus [1, 2, 8], with algebraic terms and term rewriting [6].

There exist several different formalisms on the integration of typed lambda calculus and rewrite systems, and on the study of the interaction between algebraic rewriting and β-reduction. The earliest one seems to be the work on *combinatory reduction systems* due to Klop [12]. In [3] Breazu-Tannen combines the rules for λ-terms with arbitrary first order rewrite rules, and modular properties of the combined system is proved. More recently Jouannaud and Okada [10] introduced the *Algebraic Functional Language* allowing to define higher order functional constants by rewrite rules which follows a general schema. Also here modularity results are given, in particular with respect to termination aspects. Another approach is given in the work of Nipkow [16] where it is presented a rewriting relation modulo β and η-conversion. The decidability of validating an equation between higher-order terms

---
* E-mail: Olav.Lysne@ifi.uio.no. On leave from Department of informatics, University of Oslo, P.O. box 1080 Blindern, 0316 Oslo, Norway. Supported by the Norwegian Research Council.

** E-mail: jpiris@dsic.upv.es. On leave from U.P.V., D.S.I.C., Camino de Vera S.N., P.O. box 22012, 46071 Valencia, Spain in the theory induced by the λ-conversions and a set of particular equations is considered, and partly solved by showing that local confluence is decidable, whenever all left hand sides of the rewrite rules are restricted to so called *Patterns*.

The work of Nipkow leans on the decidability of unification of patterns. The fact that the unification procedure for second order terms due to Huet [9] terminates for patterns, yielding a decision procedure was discovered by Miller [15]. Furthermore he showed that when two patterns are unifiable there exists a most general unifier.

We shall here consider the problem of termination of the rewrite relation due to Nipkow, by extending the recursive path ordering into his notion of higher order rewriting. In section 2 we give the basic notions needed to understand this paper. Section 3 describes a first approach to an ordering, whereas sections 4 and 5 presents an extension of this ordering wrt. flexible terms. Section 6 is devoted to an extension of our method, and in section 7 we comment on related work.

2 Basic notions

We use the following conventions for notation: Lowercase letters at the end of the alphabet denote bound variables, whereas capital letters always denote free variables. Thereby the set of bound and the set of free variables are always kept disjoint. We let $\mathcal{F}$ denote the set of *constants* (function symbols), and single constants are denoted by the lowercase letters f, g and h. *Atoms* may be either constants, bound or free variables, and are denoted by a and b.

A *simply typed λ-term* is built from constants, λ-binders, free and bound variables according to the following grammar:

$$t \;=\; F \mid x \mid f \mid \lambda x.t \mid (t\ \ u)$$

Lowercase letters in the middle of the alphabet (s, t, u, v) denote simply typed λ-terms, (*terms* for short). A term which contains no free variables is a *ground term*. We say that a term t is *consistent* if it does not contain other bound variables than those that are actually bound by a λ-binder within t itself.

By $\lambda\overline{x_n}.t$ or simply $\lambda\overline{x}.t$ we mean $\lambda x_1 \ldots \lambda x_n.t$, and that t is not a λ-abstraction. Furthermore we shall write $a(t_1, \ldots, t_n)$ meaning $((\ldots((a\ \ t_1)\ \ t_2)\ldots)\ \ t_n)$.

The set of *types* for the simply typed λ-terms is generated from a set $\mathcal{B}$ of *base types* (e.g. *nat*, *bool*, *int*...) and the binary constructor $\longrightarrow$. The constructor $\longrightarrow$ is right associative, thus $\alpha_1 \longrightarrow \alpha_2 \longrightarrow_3 \alpha_3$ means $\alpha_1 \longrightarrow (\alpha_2 \longrightarrow_3 \alpha_3)$.

Mappings from variables to terms are written $\{X_n/x_n \mapsto t_n\}$. *Substitutions* are mappings from free variables to terms and are denoted by lowercase Greek letters. Mappings are treated as (postfix) operators to terms, which simultaneously replaces the occurrences of variables with the corresponding term.

Two terms t and u are α-equivalent, written $t =_\alpha u$ if t can be reached from u by a bijective mapping which renames bound variables. A *β-reduction* is the transformation of a term $(\lambda x.s)\ \ t$ into $s\{x \mapsto t\}$. A *β-normal form* is a term which cannot be β-reduced. It is well known that β-reduction terminates for simply typed λ-terms, thus every term has a

β-normal form. An *η-reduction* is the transformation of a term $\lambda x.(t \ \ x)$ into t whenever x does not occur in t. An *η-long* form is a β-normal form $\lambda\overline{x}.a(t_1,\ldots,t_n)$ where $a(t_1,\ldots,t_n)$ is of base type and every t_i is η-long. This way every occurrence of an atom a of type $\alpha_1 \longrightarrow \ldots \longrightarrow \alpha_n \longrightarrow \beta$ in a η-long form, will be at the head of a basic subterm of the form $a(t_1,\ldots,t_n)$. Whenever we speak of a β-normal form we implicitly assume that it is η-long as well, unless stated otherwise. The β-normal (η-long) form of t is denoted $t\!\downarrow$. The *set of external λ-binders* in a β-normal form is the set containing all the outermost λ-binders in the term down to the first atom. For brevity we allow ourselves to write variables in β-normal forms as x and X instead of $\lambda\overline{y}.x(\overline{y})$ and $\lambda\overline{y}.X(\overline{y})$ whenever the outermost λ-binders are not important to the discussion. A β-normal form $\lambda\overline{x}.a(t_1,\ldots,t_n)$ is *flexible* if a is a free variable, and *rigid* otherwise. A β-normal form t is called a *Pattern* if all arguments of any given free variable in t are distinct bound variables.

We shall in this paper use notions of positions and subterms that are standard for first order algebraic terms, and nonstandard for λ-calculus. We shall view a term in β-normal form as an ordered tree. This by first regarding the occurring λ-binders as nodes with a single subtree, and second by viewing subterms on the form $a(t_1,\ldots,t_n)$ as a tree with a as root, and with the n subtrees $t_1,\ldots,t_n$. Notice that in our notion of term tree the "application" nodes of the λ-term have disappeared, giving a more restrictive notion of subterm. A *position* in a term can be viewed as a finite sequence of natural numbers, pointing out a path from the root of this tree. The place in the term where the path ends is the actual position. By $\mathcal{P}os(t)$ we denote the set of positions in t. The subterm of t at position p, written $t|_p$, is the subterm of t which has its root symbol at position p. The result of replacing the subterm of t at position p by the term u is written $t[u]_p$. A position in a term is said to be a *flexible position* if it occurs within a flexible subterm.

A *higher order rewrite rule* is an ordered pair of consistent β-normal forms, written $t \to u$, such that t and u are of the same type. In this paper we only consider rewrite rules in which the left hand sides are basic patterns, and the right hand sides are arbitrary terms of base type. A *rewrite system* is a set of rewrite rules. A rewrite system R induces a rewrite relation $\to_R$ on the set of β-normal forms:

$$s \to_R t \Leftrightarrow \exists(l \to r) \in R, p \in \mathcal{P}os(s), \sigma \ . \ s|_p =_\alpha l\sigma \wedge t =_\alpha s[r\sigma]_p\!\downarrow$$

A sequence of rewrite steps is written $t_1 \to_R t_2 \to_R t_3 \ldots$, and consists only of terms in β-normal forms. It has been shown that local confluence of this rewrite relation is decidable [16], and that given convergence this relation may be used to decide $\alpha\cup\beta\cup\eta\cup R$-equivalence.

3 RPO for higher order rewriting

As a first approach to a termination ordering for higher order terms we present a way to interpret β-normal higher-order terms as first-order ones. These first order terms are in turn compared by a slightly modified recursive path ordering [4].

Definition 1. Let $\mathcal{M}$ interpret higher order terms as first order terms by

1. replacing all bound variables by a (one and the same) fresh constant c,

2. replacing all corresponding λ-binders λx with the new unary constant λ and
3. replacing all free variables with new constants, conserving the identity relation between the variables.

Notice that the structure of the terms are kept by $\mathcal{M}$. This means that we may use $\mathcal{M}$ as a means to view the terms as both first order and higher order, maintaining the obvious one to one correspondence between the nodes. In the following we therefore allow ourselves to switch focus between the higher order view and the first order view of expressions like $\mathcal{M}(t)$ without further ado. Each time it will be clear from the context what our view is.

We shall now define a recursive path ordering for β-normal higher order terms, staying close to the definition of the general path order of Dershowitz and Hoot [5]. We start by introducing some special mappings, called *termination functions*, from the set of terms to some partially ordered set. Examples of such functions are the following:

1. A function that returns the outermost constant symbol to be compared using a precedence.
2. A function that extracts an immediate subterm at a specified position, to be compared recursively in the induced term ordering.
3. A function that extracts an immediate subterm given by a specified selection criterion. (e.g. the i'th largest according to an ordering).

In [5] there are several additional (up to seven) classes of termination functions, opening for the definition of a wide range of different path orderings. At a later stage we shall also introduce an additional branch of termination functions, but for the time being we shall only be concerned with those mentioned above.

Definition 2. Let $\tau_0, \ldots, \tau_n$ be termination functions, $>_{\tau_0}, \ldots, >_{\tau_n}$ the corresponding partial orderings and $(>_{\tau_0} \ldots >_{\tau_n})_{lex}$ their lexicographic combination. The *recursive path ordering for mapped higher order terms*, denoted $\succ_{rpo}$ is defined as it appears in the following inference system:

$$\frac{\begin{array}{c} s_i \succsim_{rpo} t \qquad 1 \leq i \leq m \\ f \textit{ is not mapped from a free variable} \end{array}}{f(s_1, \ldots, s_m) \succ_{rpo} t} \qquad \textbf{subterm}$$

$$\frac{\begin{array}{c} s = f(s_1, \ldots, s_m), \quad t = g(t_1, \ldots, t_n) \\ f(s_1, \ldots, s_m) \succ_{rpo} t_1, \ldots, t_n \\ \langle \tau_0(s), \ldots, \tau_n(s) \rangle \ (>_{\tau_0} \ldots >_{\tau_n})_{lex} \ \langle \tau_0(t), \ldots, \tau_n(t) \rangle \\ f \textit{ and } g \textit{ are not mapped from free variables} \end{array}}{s \succ_{rpo} t} \qquad \textbf{recpath}$$

Finally we define the ordering $\succ_{\mathcal{M}}$ on higher order patterns to be such that $t \succ_{\mathcal{M}} u$ iff $\mathcal{M}(t) \succ_{rpo} \mathcal{M}(u)$ holds for the first order interpretation of t and u.

For the time being we fix the termination function τ_0 such that it returns the outermost function symbol to be compared using a precedence, and $\tau_1, \ldots, \tau_n$ to be functions that return immediate subterms according to the lexicographic or the multiset path ordering. Notice that the inference rules above represent a restriction of the general path ordering considered by Dershowitz and Hoot [5], thus inherits the well foundedness property. See also [7] for a general and elegant proof of well foundedness. The reason for the additional restrictions we have made on the leading symbols in the inference rules is that without these restrictions it is easy to construct counterexamples to stability over substitutions. This simply by letting variables be instantiated by λ-terms which disregards all of its input arguments.

In order to prove that a term rewriting system is terminating it is not sufficient in to show that every rewrite rule is contained in some well founded ordering. It is also necessary to show that every possible rewrite step is contained in the well founded ordering. This is usually implied by the ordering being *monotonous*. We shall establish what this amounts to in our higher order setting, and begin by presenting some basic facts on the rewriting relation and the β-normal forms we consider:

Lemma 3. *Let u be in β-normal η-long form. Then for any substitution σ containing only β-normal η-long forms we reach $u\sigma\downarrow$ from $u\sigma$ by β-reductions only.*

Proof. If the β-normal form of $u\sigma$ was not η-long, then subterms of form $a(t_1, \ldots, t_n)$ that are not of base type must have been introduced either by u, σ or a β-reduction step. All three are clearly impossible. □

Thereby we need not worry about the effect of η-reduction in the following.

Lemma 4. *If u is of base type then $t\downarrow [u\downarrow]_p$ is in β-normal form.*

Proof. Since $t\downarrow$ and $u\downarrow$ are in normal forms the only possibility for $t\downarrow [u\downarrow]_p$ not to be in normal form is if the insertion of u created a β-reduction redex. Since u is of base type this cannot be. □

Finally we state that in a proof of termination of a higher order rewrite system we only need to consider derivations with ground terms:

Lemma 5. *If there exists an infinite rewriting derivation for a set R of higher order rewrite rules, then there also exists one which only contains ground terms.*

Proof. Analogously to the first order case we start with an infinite derivation, and replace all occurring free variables by fresh constants. The rest follows by observing that this will not result in any changes in the matchings and the β-reductions. □

This leaves us with the following definition for monotonicity over application of higher order rewrite rules:

Definition 6. An ordering $\succ$ on higher order rewrite rules is

- *stable wrt. substitutions* if $t\downarrow \succ u\downarrow \Rightarrow (t\downarrow \sigma)\downarrow \succ (u\downarrow \sigma)\downarrow$ for all consistent basic patterns t, basic terms u and ground substitutions σ.
- *stable wrt. context application* if $t\downarrow \succ u\downarrow \Rightarrow v\downarrow [t\downarrow]_p \succ v\downarrow [u\downarrow]_p$ for all positions p, ground contexts v and basic ground terms t and u, such that t is an instance of a basic pattern.

An ordering is *monotonous* if it is stable wrt. substitutions and context application.

Let us now present a basic fact on the effect of first applying a substitution to a term and then finding the β normal form.

Lemma 7. *Let $a(s_1,\ldots,s_n)$ be a term in β-normal form, where a is either a lambda-binder, constant or a bound variable. Then $a(s_1,\ldots,s_n)\sigma\downarrow = a(s_1\sigma\downarrow,\ldots,s_n\sigma\downarrow)$.*

Proof. Assume that a is a lambda-binder. The lemma is obvious for all other cases. In that case we must show that $(\lambda x.t)\sigma\downarrow = \lambda x.(t\sigma\downarrow)$, which is obvious from a study of the possible β-reduction redexes. □

Obviously if a is a bound variable in the above lemma then the term is not consistent. The lemma is however stated like that in order to be used in induction on the complexity of the terms, and clearly non-consistent subterms occur in consistent terms. The following lemma implies that the ordering is *sound* in the sense that any rewrite system contained in the ordering actually terminates:

Lemma 8. *The relation $\succ_{\mathcal{M}}$ is monotonous.*

Proof. Stability wrt. substitutions: We show that $t\sigma\downarrow \succ_{\mathcal{M}} u\sigma\downarrow$ by cases of $t \succ_{\mathcal{M}} u$ using induction on the depth of t and u.

subterm: Here $t = f(t_1,\ldots,t_n)$ and $t_i \succsim_{\mathcal{M}} u$ for some i. By induction we get $t_i\sigma\downarrow \succsim_{\mathcal{M}} u\sigma\downarrow$, and according to lemma 7 we have $t\sigma\downarrow = f(t_1\sigma\downarrow,\ldots,t_n\sigma\downarrow)$. Thus by the **subterm** inference $t\sigma\downarrow \succ_{\mathcal{M}} u\sigma\downarrow$.

recpath: Now $t = f(t_1,\ldots,t_n)$ and $u = g(u_1,\ldots,u_m)$. If f has precedence over g then we have $f(t_1,\ldots,t_n) \succ_{\mathcal{M}} u_1,\ldots,u_m$, and by induction $f(t_1,\ldots,t_n)\sigma\downarrow \succ_{\mathcal{M}} u_1\sigma\downarrow,\ldots,u_m\sigma\downarrow$. From lemma 7 we have $u\sigma\downarrow = g(u_1\sigma\downarrow,\ldots,u_m\sigma\downarrow)$ thus the **recpath** rule gives $t\sigma\downarrow \succ_{\mathcal{M}} u\sigma\downarrow$. Otherwise f and g have the same precedence and $t_1,\ldots,t_n$ is either lexicographically or multiset greater than $u_1,\ldots,u_m$. By induction we easily get that $t_1\sigma\downarrow,\ldots,t_n\sigma\downarrow$ is respectively lexicographically or multiset greater than $u_1\sigma\downarrow,\ldots,u_m\sigma\downarrow$. In the same way as above lemma 7 and the **recpath** rule finishes the proof.

Stability wrt. context application: This is immediate from the corresponding proof for the first order case. □

Example 1. This example is fetched from [16]. The following is a higher order rewrite system for computing the negation normal form of formulae in classical first-order logic. We have two base types, *term* and *form* of terms and formulae, and here are the profiles of the constants:

$$\neg : \mathit{form} \longrightarrow \mathit{form}$$

$$\_\wedge\_,\_\vee\_ : \mathit{form} \longrightarrow \mathit{form} \longrightarrow \mathit{form}$$

$$\forall,\exists : (\mathit{term} \longrightarrow \mathit{form}) \longrightarrow \mathit{form}$$

Terms like $\forall(\lambda x.P(x))$ should be interpreted as $\forall x.P(x)$. Following Nipkow we find it sufficient to present half of the rewrite system:

$$\neg(\neg(\lambda x.P(x))) \to \lambda x.P(x)$$
$$\neg(\lambda x.P(x) \wedge \lambda x.Q(x)) \to (\neg(\lambda x.P(x)) \vee \neg(\lambda x.Q(x)))$$
$$\neg(\forall(\lambda x.P(x))) \to \exists(\lambda x.\neg(P(x)))$$

This rewrite system is shown to be confluent in [16], and termination may easily be proven by $\succ_{\mathcal{M}}$ with the precedence $\neg$ greater than everything.

4 Handling higher order variables

In the previous section we had to restrict the comparison of flexible terms due to the fact that for these terms it is hard to control stability wrt. substitutions. The effect of this restriction is that we may only prove termination of rules that have patterns in the right hand side as well as the left hand side. This excludes most of the higher order programs that one would like to write, e.g. the definition of the *map* function:

$$map(\lambda x.F(x), \epsilon) \to \epsilon$$
$$map(\lambda x.F(x), X :: Y) \to F(X) :: map(\lambda x.F(x), Y)$$

The strengthening of the ordering proceeds in two steps. The first step is to relieve the first termination function τ_0 from being a precedence on the outermost function symbols. The second step is to extend the underlying inference system.

Let us concentrate on τ_0 first. We leave open the exact definition of it, and instead we indicate what the requirements are on τ_0 in order for the proof of monotonicity of our extended ordering to go through.

Definition 9. Let τ_0 be a termination function taking a higher order term as input. Furthermore let $\succ_{\mathcal{M}^{\tau_0}}$ be the corresponding recursive path ordering on higher order terms. We say that τ_0 is *monotonous* if it is a mapping into a well founded set of values such that

1. $\tau_0(t) > \tau_0(u) \Rightarrow \tau_0(t\sigma\downarrow) > \tau_0(u\sigma\downarrow)$ for every pair of terms t and u both in β-normal form, and every ground substitution σ.
2. $\tau_0(t) = \tau_0(u) \Rightarrow \tau_0(t\sigma\downarrow) = \tau_0(u\sigma\downarrow)$ for every pair of terms t and u both in β-normal form, and every ground substitution σ.
3. $t \succ_{\mathcal{M}^{\tau_0}} u \Rightarrow \tau_0(v[t\sigma\downarrow]_p) \geq \tau_0(v[u\sigma\downarrow]_p)$ for every ground term v in β-normal form, consistent basic pattern t and consistent basic term u both in β-normal form, position p not at the top, and ground substitution σ.

The first two of the above restrictions ensures that $\succ_{\mathcal{M}^{\tau_0}}$ is stable over substitutions, whereas the last is there for stability over context application. The above restrictions on τ_0 do not suffice to guarantee that τ_0 falls into one of the categories of termination functions of Dershowitz and Hoot. This because according to these categories a termination function that maps terms into a well founded set must either be a precedence on the outermost function symbols, or a homomorphism. Neither of the two are implied by our notion of monotonicity, and in a later section we shall indeed propose an instance of τ_0 that is monotonous but no homomorphism. Still the proof of well foundedness of their ordering (which is stated to be akin to Kamin and Lévy [11]) trivially extends to ours. The proof is only based on the facts that the ordering contains the subterm relation, and that each of the termination functions $\tau_0, \ldots, \tau_n$ induces a well founded ordering. Both of these facts are kept by monotonicity of τ_0.

The next definitions establish the basic concepts in our proposal for the treatment of flexible subterms. First we take into account the paths from the top of a higher order ground term down to the subterms that are prone to change due to β-reduction.

Definition 10. Let t be a higher order ground term. The set of *critical positions* in t consists of the positions in t that lies on a path from the outermost λ-binders down to an occurrence of a corresponding bound variable, excluding the λ-binders and the bound variables themselves.

Example 2. The set of symbols occurring in critical positions in the term $\lambda xy.f(g(x), h(h', y))$ is $\{f, g, h\}$.

Definition 11. A higher order variable is *simple* if all of its arguments are basic.

Now a lemma on the effect of β-reduction on simple variables:

Lemma 12. *Let X be a simple higher order variable and let $u_1, \ldots, u_n$ be terms of base type. Then for all σ we have that $X(u_1, \ldots, u_n)\sigma\!\downarrow$ may be obtained from $X\sigma\!\downarrow$ by replacing all occurrences of variables bound by outermost lambdas by $u_i\sigma$ for some i, and removing the outermost lambda-binders.*

The lemma might seem obvious, but notice that it is no longer valid if one u_i is not of base type.

And now we prepare for the control of the β-reduction by indicating how the τ_0 function may be used to control free variables.

Definition 13. Let t be a basic pattern, and let X be a higher order free variable. We say that $\tau_0(t)$ *dominates* X, iff $\tau_0(t\sigma\!\downarrow) > \tau_0(X\sigma\!\downarrow|_p)$ for all positions p that are critical in $X\sigma\!\downarrow$.

Informally the idea is that when $\tau_0(t)$ dominates a free variable X, then the comparison of $t\sigma\downarrow$ with $X(s_1, \ldots, s_n)\sigma\downarrow$ in the ordering shall be reduced to the comparison of $t\sigma\downarrow$ with the terms $s_1\sigma\downarrow, \ldots, s_n\sigma\downarrow$. This by recursive applications of the **recpath** rule along the critical positions in $X\sigma$. At this point in the development we encounter the reason why we need to be able to maintain also the higher order view on expressions like $\mathcal{M}(t)$. This is because the information needed to compute critical positions is lost in the first order version of t.

In order to carry through the idea sketched above, we need to require another property of τ_0, namely that it is in a way insensitive to β-reduction:

Definition 14. A termination function τ is said to be *stable under β-reduction* if whenever $\tau(t) > \tau(G\sigma|_p)$ for G a simple variable and p a critical position in $G\sigma$ then $\tau(t) > \tau_0(G(v_1, \ldots, v_n)\sigma\downarrow|_{p'})$ where p' is obtained from p by removing the j first 1's and j is the number of external λ-binders in $G\sigma$.

The definition of β-stability above is rather technical, but the last part regarding the computation of p' from p is only there to remedy the fact that β-reduction will remove all outermost λ-binders from $G\sigma\downarrow$ in $G(v_1,\ldots,v_n)\sigma\downarrow$.

At this point we extend $\succ_{rpo}$ into $\succ_{erpo}$ by changing the name of the actual relation in **subterm** and **recpath** rules, and adding the following inference rule for flexible subterms:

$$\frac{\tau_0(s) \text{ dominates } G \qquad s \succ_{erpo} t_1,\ldots,t_n \quad \\ G \textit{ is simple and occurs in } s \textit{ without free variables above it}}{s \succ_{erpo} G(t_1,\ldots,t_n)} \quad \textbf{flexrecpath}$$

Example 3. Let us study the problem we started this section with, namely the rule

$$map(\lambda x.F(x), X :: Y) \to F(X) :: map(\lambda x.F(x), Y)$$

Let us assume that $\tau_0(map(\lambda x.F(x), X :: Y)) > \tau_0(F(X) :: map(\lambda x.F(x), Y))$, and that $\tau_0(map(\lambda x.F(x), X :: Y)) \geq \tau_0(map(\lambda x.F(x), Y))$. Then we would be able to prove that this higher order rewrite rule is contained in $\succ_{\mathcal{M}^{\tau_0}}$ with $\succ_{erpo}$ at the bottom, granted that $\tau_0(map(\lambda x.F(x), X :: Y))$ dominates F.

We are now ready to prove that prove that $\succ_{\mathcal{M}^{\tau_0}}$ is monotonous.

Lemma 15. *Let τ_0 be a monotonous and β-stable termination function, and let $\tau_1,\ldots,\tau_n$ correspond to lexicographic or multiset comparison of immediate subterms. Furthermore let $\succ_{\mathcal{M}^{\tau_0}}$ be the corresponding ordering on higher order terms based on $\succ_{erpo}$. Then $\succ_{\mathcal{M}^{\tau_0}}$ is monotonous.*

Proof. Stability wrt. substitutions: The proof is constructed in the same manner as the proof of lemma 8:

subterm: Identical to the corresponding case in the proof of lemma 8.

recpath: Clear from the corresponding case of lemma 8 by replacing the comparison of outermost function symbols by a monotonous termination function τ_0.

flexrecpath: Then we are in the situation $s \succ_{\mathcal{M}^{\tau_0}} G(t_1,\ldots,t_n)$, where s is a pattern, $\tau_0(s)$ dominates G and G occurs in s in such a way that it cannot disappear by β-reduction. We must prove $s\sigma\downarrow \succ_{\mathcal{M}^{\tau_0}} G(t_1,\ldots,t_n)\sigma\downarrow$.
Consider the topmost position in $G(t_1,\ldots,t_n)\sigma\downarrow$. From lemma 12 we have one of three cases:

1. The topmost position stems from a non-critical position in $G\sigma$. From lemma 12 and the definition of critical positions we get that $G(t_1,\ldots,t_n)\sigma\downarrow$ is a subterm of $G\sigma\downarrow$, and by the occurrence of G in s this case is finished by the **subterm** rule.
2. The topmost position stems from a critical position in $G\sigma$. Since $\tau_0(s)$ dominates G we get that $\tau_0(s\sigma\downarrow) > \tau_0(G\sigma\downarrow|_p)$ where p is the topmost position in $G\sigma$ which is not a λ-binder. Since τ_0 is stable under β-reduction we must have that $\tau_0(s\sigma\downarrow) > \tau_0(G(t_1,\ldots,t_n)\sigma\downarrow)$. This means that we may use the **recpath** inference and transform the proof obligation into that of proving $s\sigma\downarrow \succ_{\mathcal{M}^{\tau_0}} u$ for all immediate subterms u of $G(t_1,\ldots,t_n)\sigma\downarrow$. The proof follows by induction on the depth of $G(t_1,\ldots,t_n)\sigma\downarrow$.

3. $G(t_1,\ldots,t_n)\sigma\downarrow$ equals $t_i\sigma$ for some i such that $1 \leq i \leq n$. In that case we must prove that $s\sigma \succ_{\mathcal{M}^{\tau_0}} t_i\sigma$ The inference rule **flexrecpath** gives us that $s \succ_{\mathcal{M}^{\tau_0}} t_i$, thus this follows by induction.

Stability wrt. context application: Since τ_0 is monotonous this follows by a straightforward adaptation of corresponding proofs from the first order case. □

Theorem 16. *Let τ_0 be a monotonous and β-stable termination function, and let $\succ_{\mathcal{M}^{\tau_0}}$ be the corresponding recursive path ordering on higher order terms based on $\succ_{erpo}$. Then any higher order rewrite system which is contained in $\succ_{\mathcal{M}^{\tau_0}}$ is terminating.*

Proof. By analysis of the proof of lemma 15 we see that every ground rewrite step of R is reducing by $\succ_{\mathcal{M}^{\tau_0}}$ using the inference rules of $\succ_{rpo}$ only. We have already shown that $\succ_{\mathcal{M}^{\tau_0}}$ with $\succ_{rpo}$ at the bottom is well founded. □

5 A monotonous and β-stable termination function

So let us now present an example of a termination function which has the desired properties. Let us for the sake of simplicity first restrict ourselves to ground terms. The idea is as follows: Let a denote a constant symbol or a bound variable. If a has a higher order argument, we make sure that $\tau_0(a(\ldots,\lambda x.t,\ldots)) > \tau_0(\lambda x.t|_p)$ for all p that are critical in $\lambda x.t$. Following this idea we may assume that $\tau_0(a(\ldots,X,\ldots))$ dominates the higher order variable X.

We start by assigning an integer value, $val(a)$ to each symbol $a \in \mathcal{F} \cup \{\lambda, c\}$. As a means to control the critical positions in a term we define a function $\mathcal{O}$:

Definition 17. Let $\mathcal{O}(t)$ be $max(\tau_0(t|_p))$ for all critical positions p in t.

And now for the definition of τ_0 having the properties announced above:

Definition 18. We hereby fix the termination function τ_0 such that
$\tau_0(a(t_1,\ldots,t_m)) = val(a) + max(\mathcal{O}(t_i))$ for $1 \leq i \leq m$.

The ordering assigned to τ_0 is simply the *greater than* relation on natural numbers. Notice that the function τ_0 is not a homomorphism. (A mapping function ϕ is a homomorphism if $\phi(a(t_1,\ldots,t_n)) = a_\phi(\phi(t_1),\ldots,\phi(t_n))$ for some function a_ϕ.) This because it is impossible to compute $\tau_0(a(t_1,\ldots t_m))$ from $\tau_0(t_1),\ldots,\tau_0(t_m)$. Also note that τ_0 needs the higher order version of the term. For terms with free variables, it is straightforward to extend the ordering, e.g by letting $\{\mathcal{O}(t)\} \geq \{\mathcal{O}(u)\}$ whenever u is a proper subterm of t in a non-flexible position.

In order to be able to use this version of τ_0 in the method we have described, we must prove that is is monotonous and β-stable. First we need to demonstrate a simple technical property.

Lemma 19. *Let X be a free variable, $t_1,\ldots,t_n$ be arbitrary terms, and σ be a substitution. Consider the term $X(t_1,\ldots,t_n)\sigma$. Then the term introduced for the X-occurrence by σ will not contain variables bound in any of $t_1,\ldots,t_n$ and vice versa.*

Proof. We have that $X(t_1,\ldots,t_n)\sigma = ((\ldots(X\sigma\ \ t_1\sigma)\ldots)\ \ t_n\sigma)$, thus the scoping rules of λ-binders give the rest. □

Lemma 20. *The domination relation induced by τ_0 is stable under β-reduction.*

Proof. Obvious from lemma 19. □

Lemma 21. *The termination function τ_0 is monotonous.*

Proof. Regarding the three points stated definition 9 the proofs that τ_0 fulfils the requirements of points 1. and 2. are trivial. To enhance the reading of the proof of point 3. we restate the proof obligation:

$t \succ_{\mathcal{M}^{\tau_0}} u \Rightarrow \tau_0(v[t\sigma\downarrow]_p) \geq \tau_0(v[u\sigma\downarrow]_p)$ *for every ground term v in β-normal form, consistent basic pattern t and consistent basic term u both in β-normal form, position p not at the top, and ground substitution σ.*

Let us assume that we have a counterexample to the above. Without loss of generality we may assume that the term assigned do v in the counterexample is minimal, in the sense that no subterm of v provides another counterexample. According to the definition of τ_0 this would mean that there exists an immediate subterm v' of v, such that $\mathcal{O}(v'[t\sigma\downarrow]_{p'}) < \mathcal{O}(v'[u\sigma\downarrow]_{p'})$. Also v' must be the smallest term satisfying this latter inequality.

Therefore we need consider the τ_0-values of the critical positions in $v'[t\sigma\downarrow]_{p'}$ and in $v'[u\sigma\downarrow]_{p'}$. Since v provided the smallest counterexample, the τ_0 values cannot have increased above position p', thus critical positions at and below p' are the only ones we need to study.

In order to abstract away the context v' of the above, we let $\lambda x_1 \ldots \lambda x_j$ be the outermost λ-binders of v'. The counterexample may now be rewritten as the inequality

$$\mathcal{O}(\lambda x_1 \ldots \lambda x_j \;\; t\sigma\downarrow) < \mathcal{O}(\lambda x_1 \ldots \lambda x_j \;\; u\sigma\downarrow)$$

Let $\mathcal{O}'$ be the slight modification of $\mathcal{O}$ that only considers the at most j first outermost λ-binders in the computation of critical positions. We shall prove that

$$\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; t\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; u\sigma\downarrow)$$

Notice that for t and u of base type $\mathcal{O}$ and $\mathcal{O}'$ coincide in the above expression, thus this actually suffices to finish this subproof. We make the proof by cases of $\mathcal{M}(t) \succ_{erpo} \mathcal{M}(u)$ by induction on the depth of t and u. In the following a and b denotes elements in $F \cup \{c, \lambda\}$.

subterm Here we know that $t = a(\overline{v})$, and that $\mathcal{M}(v_i) \succ_{erpo} \mathcal{M}(u)$ for some index i. By induction it follows that $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; v_i\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; u\sigma\downarrow)$ which clearly implies $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; a(\overline{v})\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; u\sigma\downarrow)$.

recpath Let $t = a(\overline{v})$ and $u = b(\overline{s})$. The inference rule gives that $\mathcal{M}(t) \succ_{erpo} \mathcal{M}(s_i)$ for all i. By induction we get $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; a(\overline{v})\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; s_i\sigma\downarrow)$ thus the position with the biggest τ_0-value in $\overline{s\sigma\downarrow}$ does not have a bigger τ_0 value than the corresponding one in $t\sigma\downarrow$. We have $\tau_0(a(\overline{v})) \geq \tau_0(b(\overline{s}))$ from the inference rule and therefore $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; a(\overline{v})\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; b(\overline{s})\sigma\downarrow)$.

flexrecpath Now $t = a(\overline{v})$,$u = G(\overline{s})$ and $\tau_0(a(\overline{v}))$ dominates G, and G occurs in $a(\overline{v})$ in a non-flexible position. Let us study the position of interest in $G(\overline{s})\sigma\downarrow$ with the biggest τ_0-value. According to lemma 12 the subterm of this position falls into one of three classes:

1. It is a subterm of $G\sigma\downarrow$. In that case it is also a subterm of $a(\overline{v})\sigma\downarrow$, thus the τ_0 value of the subterm cannot be greater than $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; a(\overline{v})\sigma\downarrow)$.
2. It is a subterm of $s_i\sigma\downarrow$ for some i. In this case the inference rule **flexrecpath** states that $\mathcal{M}(a(\overline{v})) \succ_{erpo} \mathcal{M}(s_i)$. By induction we get $\mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; a(\overline{v})\sigma\downarrow) \geq \mathcal{O}'(\lambda x_1 \ldots \lambda x_j \;\; s_i\sigma\downarrow)$ thus this case is unproblematic.
3. The position of the subterm corresponds to a critical position in $G\sigma\downarrow$. In that case the β-stability and the fact that $\tau_0(a(\overline{v}))$ dominates G guarantees that the τ_0-value is less than $\tau_0(a(\overline{v}))$.

□

Let us now state explicitly what it takes for a variable to be dominated by a τ_0-expression:

Lemma 22. *The expression $\tau_0(a(t_1, \ldots, t_n))$ dominates a simple higher order variable F, whenever $t_i = \lambda\overline{y}.F(\overline{x})$ for some i, $1 \leq i \leq n$, and a is not a free variable.*

Example 4. In order to prove termination of the rewrite system from example 3 we needed to establish the following three facts:

- $\tau_0(map(\lambda x.F(x), X :: Y)) > \tau_0(F(X) :: map(\lambda x.F(x), Y))$. This reduces to requiring $max(\mathcal{O}(\lambda x.F(x)),\ val(map)) > val(::)$. Here $val(map) > val(::)$ suffices.
- $\tau_0(map(\lambda x.F(x), X :: Y)) \geq \tau_0(map(\lambda x.F(x), Y))$. By simple inspection we get that the two expressions have identical values.
- $\tau_0(map(\lambda x.F(x), X :: Y))$ dominates F, which is clear from lemma 22.

Example 5. Let us take a look at another higher order function that is common in functional programming, namely the function *iter* which iterates the application of a function on the elements of a list. Here we use *iter* to define a function *sum* which computes the sum of the elements in a list of natural numbers:

$$iter(\lambda x.\lambda y.F(x,y), X, []) \rightarrow X \tag{1}$$

$$iter(\lambda x.\lambda y.F(x,y), X, Y :: L) \rightarrow iter(\lambda x.\lambda y.F(x,y), F(X,Y), L) \tag{2}$$

$$sum(L) \rightarrow iter(\lambda x.\lambda y.(x+y), 0, L) \tag{3}$$

The first rule is trivially contained in our ordering, the second follows by letting $\tau_1, \ldots, \tau_n$ give the lexicographic ordering of subterms from right to left. The third rule is handled by letting $val(sum)$ be greater than $val(\lambda)$ and greater than the sum of $val(+)$ and $val(iter)$.

6 Handling non-simple variables

The key to the approach we have described above is hidden in lemma 12. This because in some cases this lemma allows us to handle the comparison of an expression $\tau_0(f(u_i, \ldots, u_n))$ with a higher order variable G. These are the cases where the **recpath** rule will let the expression $\tau_0(f(u_i, \ldots, u_n))$ survive through all the critical positions of $G\sigma$ down to a subterm over which we have control. This scheme works for simple variables G, because then any substitution σ will leave $t_i\sigma\downarrow$ unchanged in $G(t_i, \ldots, t_n)\sigma\downarrow$, and therefore controllable. If any of t_i are of higher order, however, this pleasant state of affairs is altered.

Let us now study what happens if t_i is second order. Then we face the same problem as we faced with $G\sigma$ previously, namely that we must control the ability of an expression $\tau_0(f(u_i,\ldots,u_n))$ to survive through the critical positions of $t_i\sigma\downarrow$ as well. This problem may of course also be solved in the same way, by altering the **flexrecpath** rule such that it makes sure that $\tau_0(f(u_i,\ldots,u_n))$ "dominates" each of $t_i,\ldots,t_n$. In that case $\tau_0(f(u_i,\ldots,u_n))$ will survive down to a term which must be a subterm of a bound variable in $G\sigma$. If we allow t_i to be of arbitrary order, then this subterm of $G\sigma$ may also be of higher order, thus $\tau_0(f(u_i,\ldots,u_n))$ must survive down through this subterm in the same way, and so on. The scheme we have presented here may therefore be iterated in order to create an ordering where **flexrecpath** can handle flexible subterms with non-simple variables on top.

7 Related work

To our knowledge there have been two previous attempts to extend the recursive path ordering into higher order rewriting. The first is a short and tentative paper by Loría-Sáenz and Steinbach [14]. This paper focuses on the more general relation being the union between R reduction and β-reduction. It is clear that this relation subsumes the rewrite relation of Nipkow. Rewrite rules consists of β-normal η-long forms, with the additional restriction that variables in the left hand sides can only occur as leaf nodes in the term tree. This means that free higher order variables in the left hand side must be written as F rather than $\lambda\overline{x}.F(\overline{x})$.

From [14], however, it is not clear which restrictions have to be made for their ordering to work, and below we present a counterexample to the main result of the paper. Let us project their ordering into our setting. Their version of τ_0, denoted τ_0', map every term of form $a(t_1,\ldots,t_n)$ into a term $a(u_1,\ldots,u_m)$. Here $u_1,\ldots,u_m$ is the subset of $t_1,\ldots,t_n$ consisting of higher order terms. The ordering assigned to τ_0' is simply the recursive path ordering itself. We shall see that this scheme has to be restricted in a way not indicated in the paper in order to be correct: Consider the higher order rewrite system

$$g(F,Y) \to F(Y) \tag{4}$$

$$f(\lambda x.h(g'(f'))) \to g(\lambda z.f(\lambda x.h(z)), g'(f')) \tag{5}$$

When we let f' be greater in the precedence than all other constant symbols, this set of rewrite rules is contained in the ordering in [14]. For rule (4) we get that $\tau_0'(g(F,Y)) = g(F)$ is greater than $\tau_0'(F(Y)) = F$ by the subterm property. For rule (5) we get that $\tau_0'(f(\lambda x.h(g'(f')))) = f(\lambda x.h(g'(f')))$ is greater than $\tau_0'(g(\lambda z.f(\lambda x.h(z)), g'(f'))) = g(\lambda z.f(\lambda x.h(z)))$ and also greater than $\tau_0'(f(\lambda x.h(z))) = f(\lambda x.h(z))$ because neither of the two latter contain the most significant constant symbol f'. For both of the rewrite rules the rest of the reasoning needed in order to show that it is contained in the ordering is straightforward. But this set is not terminating since it gives rise to the infinite sequence

$$g(\lambda z.f(\lambda x.h(z)), g'(f')) \to_{(4)} f(\lambda x.h(g'(f'))) \to_{(5)} g(\lambda z.f(\lambda x.h(z)), g'(f'))\ldots$$

What seems like an immediate remedy, is to restrict the ability of any constant symbol to be greater than bound variables and lambda-binders. In the paper there is such a restriction on the type of the constant f', but notice that in the above example f' always occurs within the context of a constant symbol g', thus may be given an arbitrary complex type, simply by adding arguments to it in the example.

In his thesis [13], Loría-Sáenz describes an ordering which is claimed to be a restriction of the one presented in [14]. In the thesis all higher order terms are regarded as first order terms by viewing lambda binders as unary constants and bound and free variables as constants of the proper arity. There is of course a restriction saying that free variables are not comparable to anything. In addition the bound variables are restricted in such a way that they are incomparable to every other constant symbol except for that they may be equivalent to another bound variable of equivalent type. The standard recursive path ordering is then extended by extending the subterm relation. This roughly by extending the **subterm** inference rule such that it considers a term obtained by applying t_i to u be a subterm of $f(t_1, \ldots, t_i, \ldots, t_n)$ whenever u is a basic subterm of any of $t_1 \ldots t_{n-1}, t_{n+1} \ldots t_n$.

Let us compare the ordering of Loría-Sáenz to our ordering parameterised by the termination function τ_0 that we have described. We are able to show termination of rules like $f(F, zero) \rightarrow F(succ(zero))$ by letting $f > succ$ and $f > zero$. The ordering of Loría-Sáenz is not able to cope with this case. This because the only way it can handle higher order variables in the right hand side, is by the extended subterm relation. With this subterm notion there are two subterms of the left hand side which contains F, namely $F(zero)$ and F itself. Neither of these can be greater than $F(succ(zero))$. It is also fairly easy to see that if t is a subterm of u in the extended sense of [13], then we are able to show that $t \succ_{\mathcal{M}^{\tau_0}} u$. Thus we conjecture that our approach parameterised by the version of τ_0 that we have given is a proper extension of the approach in [13].

Finally we mention that the method of "termination by interpretation in a well-founded monotone algebra" described by Zantema in [18] has been extended to a higher order setting by van de Pol [17]. Here we do not make an explicit comparison with our method, since it is clear that the methods inherit the differences that are present in first order case.

8 Conclusion

We have extended the recursive path ordering so that it may be used to prove termination of higher order rewrite systems. Our method is parameterised by a function which collects information from the subterms of each occurring constant symbol. We have demonstrated what properties this function must entail in order for the soundness of the ordering to be maintained. This means that our approach really defines a family of termination orderings. We believe that further work on the development of different versions of τ_0 will provide a wide platform for proving termination of many different classes of higher order rewrite systems.

There exist some previous attempts to utilise the recursive path ordering in a higher order setting, and we have presented a counterexample to the results presented in [14]. Regarding the ordering put forward in [13] we have demonstrated that our approach is stronger when it is parameterised by the particular extraction function we have presented in this paper.

A natural path of future work will be that of non-simple free variables indicated in section 6. In addition it is reasonable to believe that the presented ideas are not limited to the lexicographic and multiset path orderings that we have considered here, but carry over to the full general path ordering of Dershowitz and Hoot.

Acknowledgements

We are grateful to Hubert Comon and Jean-Pierre Jouannaud, both for initiating this research and for numerous discussions. Special thanks are due to Jean-Pierre Jouannaud for carefully reading and commenting on previous versions of this paper.

References

1. H. P. Barendregt. *The Lambda Calculus, its Syntax and Semantics.* North Holland, Amsterdam, 2nd ed., 1984.
2. H. P. Barendregt. *Handbook of Logic in Computer Science*, chapter Typed lambda calculi. Oxford Univ. Press, 1993. eds. Abramsky et al.
3. V. Breazu-Tannen. Combining algebra and higher-order types. In *Proceedings 3rd IEEE Symposium on Logic in Computer Science, Edinburgh (UK)*, July 1988.
4. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
5. N. Dershowitz and C. Hoot. Topics in termination. In *Proceedings 5th Conference on Rewriting Techniques and Applications, Montreal (Canada)*, volume 690 of *Lecture Notes in Computer Science*, pages 198–212. Springer-Verlag, 1993.
6. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6. Elsevier, Amsterdam, 1990.
7. M. C. F. Ferreira and H. Zantema. Well-foundedness of term orderings. In *Proceedings 4th International Workshop on Conditional Term Rewriting Systems, Jesuralem (Israel)*, 1994. To be published by Springer Verlag.
8. R. Hindley and J. Seldin. *Introduction to Combinators and λ-calculus.* Cambridge University Press, 1986.
9. G. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1(1):27–57, June 1975.
10. J.-P. Jouannaud and M. Okada. A computation model for executable higer-order algebraic specification languages. In *Proceedings 6th IEEE Symposium on Logic in Computer Science*, pages 350–361, 1991.
11. S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. Unpublished Note, Department of Computer Science, University of Illinois, Urbana, IL, 1980.
12. J. W. Klop. *Combinatory Reduction Systems.* Mathematical Centre Tracts 127, Mathematisch Centrum,Amsterdam, 1980.
13. C. Loría-Sáenz. *A Theoretical Framework for Reasoning about Program Construction based on Extensions of Rewrite Systems.* PhD thesis, Fachbereich Informatik der Universität Kaiserslautern, 1993.
14. C. Loría-Sáenz and J. Steinbach. Termination of combined (rewrite and λ-calculus) systems. In *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems, Pont-a-Mousson (France)*, volume 656 of *Lecture Notes in Computer Science*, pages 143–147. Springer-Verlag, 1992.
15. D. Miller. A logic programming language with lambda-abstraction function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497–536, 1991.
16. T. Nipkow. Higher order critical pairs. In *Proceedings 6th IEEE Symposium on Logic in Computer Science*, pages 342–349, 1991.
17. J. van de Pol. Termination proofs for higher-order rewrite systems. In *First International Workshop on Higher-Order Algebra, Logic and Term Rewriting*, volume 816 of *Lecture Notes in Computer Science*, pages 305–325. Springer-Verlag, 1993.
18. H. Zantema. Termination of term rewriting by interpretation. In *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems, Pont-a-Mousson (France)*, volume 656 of *Lecture Notes in Computer Science*, pages 155–167. Springer-Verlag, 1992.

This article was processed using the LaTeX macro package with LLNCS style