PAPER *Special Section on Foundations of Computer Science*

# Higher-Order Path Orders Based on Computability

**Keiichirou KUSAKARI**[†], ***Member***

**SUMMARY** Simply-typed term rewriting systems (STRSs) are an extension of term rewriting systems. STRSs can be naturally handle higher order functions, which are widely used in existing functional programming languages. In this paper we design recursive and lexicographic path orders, which can efficiently prove the termination of STRSs. Moreover we discuss an application to the dependency pair and the argument filtering methods, which are very effective and efficient support methods for proving termination.

***key words:*** *simply-typed term rewriting system, termination, path order, computability, dependency pair, argument filtering method*

## 1. Introduction

Term rewriting systems (TRSs) are computation models. In TRSs, terms are reduced by using a set of directed equations, called rewrite rules. The most striking feature is that TRSs themselves can be regarded as functional programming languages. For example, the following TRS defines addition on natural numbers represented by the constant 0 and the successor function $S$:

$$\left\{ \begin{array}{rcl} Add(x, 0) & \to & x \\ Add(x, S(y)) & \to & S(Add(x, y)) \end{array} \right.$$

Unfortunately, TRSs cannot handle higher-order functions. For example, the *Map*-function, which is one of the most typical higher-order function in functional programming languages, is defined as follows:

$$\left\{ \begin{array}{rcl} Map(f, Nil) & \to & Nil \\ Map(f, Cons(x, xs)) & \to & Cons(f(x), Map(f, xs)) \end{array} \right.$$

This system is a legal functional program, but an illegal TRS: the variable $f$ occurs at a non-leaf position in the right-hand side of the second rule. In order to handle such the system above, we designed simply-typed term rewriting systems (STRSs) in which variable occurrences at non-leaf positions are permitted [12].

Nipkow also introduced higher-order rewriting systems (HRSs), which are rewriting systems on algebraic $\lambda$-terms, using the $\lambda$-calculus as a meta-language [13]. Intuitively, algebraic $\lambda$-terms are simply-typed $\lambda$-terms in which occurrences of function symbols are permitted. For example, the *Map*-function in HRSs is defined as follows:

$$\left\{ \begin{array}{l} map(\lambda x.F(x), [\,]) \quad \to \quad [\,] \\ map(\lambda x.F(x), X :: Xs) \\ \qquad \to F(X) :: map(\lambda x.F(x), Xs) \end{array} \right.$$

In order to prove the termination of TRSs, recursive and lexicographic path orders are proposed [5], [11], which is based on the notion of simplification orders. These path orders are extended to HRSs by Jouannaud and Rubio [7], and by Iwami and Toyama [9], [10]. We also extend path orders to STRSs [12]. All of these path orders are based on the notion of simplification orders. Unfortunately, these path orders have unnatural restrictions for type or precedence.

On the other hand, in order to prove the termination of typed $\lambda$-calculus, the notion of computability was introduced by Tait [18] and Girard [6]. Based on computability instead of simplification orders, Jouannaud and Rubio [8] and Raamsdonk [14] introduced recursive path orders in HRSs.

In this paper, we design new recursive and lexicographic path orders in STRSs based on the notion of computability, and give an more refined proof. Unlike path orders in [8], [14], our path orders are reduction orders, that is, directly simulate reduction relations. This is a very advantage, because our path orders can combine well with the dependency pair and the argument filtering methods [12], which are very effective and efficient support methods for proving termination.

## 2. Preliminaries

We assume that the reader is familiar with notions of term rewriting systems [4].

For given binary relation $>$, the lexicographic extension $>^{lex}$ on lists is recursively defined as follows:

$$\begin{array}{llll} [a_1, \ldots, a_n] & >^{lex} & [\,] & \text{if } n > 0 \\ [a_1, \ldots, a_n] & >^{lex} & [a'_1, \ldots, a'_m] & \text{if } a_1 > a'_1 \\ [a_1, \ldots, a_n] & >^{lex} & [a'_1, \ldots, a'_m] & \\ \multicolumn{4}{l}{\quad \text{if } a_1 = a'_1 \text{ and } [a_2, \ldots, a_n] >^{lex} [a'_2, \ldots, a'_m]} \end{array}$$

A multiset on a set $A$ is a set of elements of $A$ in which elements may have multiple occurrences. We use standard set notation like $\{a, a, b\}$. It will be obvious from the context if we refer to a set or a multiset. For given binary relation $>$, multisets $M$ and $N$, we define $M >^{mul} N$ as $\forall n \in N - M.\ \exists m \in M - N.\ m > n$.

A signature $\mathcal{F}$ is a finite set of function symbols, denoted by $F, G, \ldots$. A set $\mathcal{V}$ is an enumerable set of variable symbols, denoted by $x, y, f, g, \ldots$. We assume that

Manuscript received March 23, 2003.
Manuscript revised July 6, 2003.
Final manuscript received January 1, 2004.

[†]The author is with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8603 Japan.

$\mathcal{F} \cap \mathcal{V} = \emptyset$. The set $T(\mathcal{F}, \mathcal{V})$ of terms is the smallest set such that $a(t_1, \ldots, t_n) \in T(\mathcal{F}, \mathcal{V})$ whenever $a \in \mathcal{F} \cup \mathcal{V}$ and $t_i \in T(\mathcal{F}, \mathcal{V})$. In the case of $n = 0$, we denote $a$ instead of $a()$. For $s \equiv a(s_1, \ldots, s_n)$, we often write $s(t_1, \ldots, t_m)$ instead of $a(s_1, \ldots, s_n, t_1, \ldots, t_m)$. Identity of terms is denoted by $\equiv$. We define $root(a(t_1, \ldots, t_n)) = a$. $Var(t)$ is the set of variables in $t$. The size $|t|$ of a term $t$ is the number of function and variable symbols in $t$.

A non-empty set of basic types is denoted by $\mathcal{B}$. The set $\mathcal{S}$ of simple types is generated from $\mathcal{B}$ by the constructor $\rightarrow$ as $\mathcal{S} ::= \mathcal{B} \mid (\mathcal{S} \rightarrow \mathcal{S})$. A simple type $(\alpha_1 \rightarrow (\cdots \rightarrow (\alpha_n \rightarrow \alpha) \cdots))$ is abbreviated to $\alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha$.

A type attachment $\tau$ is a function from $\mathcal{F} \cup \mathcal{V}$ to $\mathcal{S}$. We assume that for each simple type $\alpha$ there exists a variable $x$ such that $\tau(x) = \alpha$. A term $a(t_1, \ldots, t_n)$ has a simple type $\alpha$ if $\tau(a) = \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \alpha$ and each $t_i$ has the simple type $\alpha_i$. A term $t$ is said to be a simply-typed term if it has a simple type $\alpha$. Then we denote $\tau(t) = \alpha$. The set of simply-typed term with simple type $\alpha$ is denoted by $T^{\alpha}_{\tau}(\mathcal{F}, \mathcal{V})$, and the set of all simply-typed terms is denoted by $T_{\tau}(\mathcal{F}, \mathcal{V})$.

A substitution is a mapping from variables to terms. A substitution over terms is defined as $\theta(a(t_1, \ldots, t_n)) = a(\theta(t_1), \ldots, \theta(t_n))$ if $a \in \mathcal{F}$; $\theta(a(t_1, \ldots, t_n)) = a'(t'_1, \ldots, t'_k, \theta(t_1), \ldots, \theta(t_n))$ if $a \in \mathcal{V}$ with $\theta(a) = a'(t'_1, \ldots, t'_k)$. We hereafter assume that $x$ and $\theta(x)$ have the same simple type for each variable $x$. We write $t\theta$ instead of $\theta(t)$.

A context is a term which has the special symbol $\square$, called hole. We hereafter assume that $\square$ is of a basic type. $C[t]$ is the term obtained from $C[\,]$ by replacing $\square$ with $t$.

A rewrite rule is a pair of terms, written by $l \rightarrow r$, such that $root(l) \in \mathcal{F}$ and $Var(l) \supseteq Var(r)$. A simply-typed term rewriting system (STRS) is a finite set $R$ of rules such that $\tau(l) = \tau(r) \in \mathcal{B}$ for all $l \rightarrow r \in R$. A reduction relation $\underset{R}{\rightarrow}$ in STRS $R$ is defined as $s \underset{R}{\rightarrow} t$ iff $s \equiv C[l\theta]$ and $t \equiv C[r\theta]$ for some rule $l \rightarrow r \in R$, context $C[\,]$ and substitution $\theta$. For example, the $Map$-function is defined as the following STRS:

$$\left\{ \begin{array}{rcl} Map(f, Nil) & \rightarrow & Nil \\ Map(f, Cons(x, xs)) & \rightarrow & Cons(f(x), Map(f, xs)) \end{array} \right.$$

In the system, we have the following reduction relation sequence.

$$\begin{array}{l} Map(S, Cons(S(0), Cons(0, Nil))) \\ \quad \underset{R}{\rightarrow} Cons(S(S(0)), Map(S, Cons(0, Nil))) \\ \quad \underset{R}{\rightarrow} Cons(S(S(0)), Cons(S(0), Map(S, Nil))) \\ \quad \underset{R}{\rightarrow} Cons(S(S(0)), Cons(S(0), Nil)) \end{array}$$

A STRS $R$ is terminating if there exists no infinite reduction sequence. We often omit the subscript $R$ whenever no confusion arises.

Suppose that $T \subseteq T(\mathcal{F}, \mathcal{V})$ and $\gg$ is a binary relation on $T$. $\gg$ is said to be monotonic in $T$ if $s \gg t \Rightarrow C[s] \gg C[t]$ for all $s, t, C[s], C[t] \in T$, and said to be stable in $T$ if $s \gg t \Rightarrow s\theta \gg t\theta$ for all $s, t, s\theta, t\theta \in T$. $\gg$ is said to be well-founded in $T$ if there exists no infinite decreasing sequence of $\gg$ in $T$. A term $t$ is said to be terminating with respect to $\gg$ in $T$ if there exists no infinite decreasing sequence of $\gg$ in $T$ starting from $t$.

**Definition 2.1:** Suppose that $T \subseteq T(\mathcal{F}, \mathcal{V})$ and $\gg$ is a binary relation on $T$. A binary relation $\gg$ is said to be a reduction order in $T$ if $\gg$ is monotonic, stable and well-founded in $T$.

**Proposition 2.2:** [12] Let $R$ be a STRS. Then $R$ is terminating iff there exist $T \subseteq T(\mathcal{F}, \mathcal{V})$ and a reduction strict order $>$ in $T$ satisfying $T_{\tau}(\mathcal{F}, \mathcal{V}) \subseteq T$ and $l > r$ for all $l \rightarrow r \in R$.

We do not assume that a reduction order is a strict order, that is, transitive and irreflexive, because path orders designed in this paper are not transitive. However it is no problem for proving termination, because $>^{+}$ is a reduction strict order if $>$ is a reduction order. Hence we obtain the following theorem from the proposition above.

**Theorem 2.3:** Let $R$ be a STRS. Then $R$ is terminating iff there exist $T \subseteq T(\mathcal{F}, \mathcal{V})$ and a reduction order $>$ in $T$ satisfying $T_{\tau}(\mathcal{F}, \mathcal{V}) \subseteq T$ and $l > r$ for all $l \rightarrow r \in R$.

Keep in mind that we distinguish the notions of reduction order and reduction strict order in this paper.

Finally, we introduce the notion of well-formed terms, which is mainly used throughout the paper.

**Definition 2.4:** Suppose that $\mathcal{B}$ is a set of basic types, and $\tau$ is a type attachment. We introduce a special basic type $*$, which represent any basic type. We define a function $skel$ as $skel(\alpha) = *$ if $\alpha \in \mathcal{B}$; $skel(\alpha \rightarrow \beta) = skel(\alpha) \rightarrow skel(\beta)$. We define a type attachment $\tau_0$ as $\tau_0(\alpha) = skel(\tau(\alpha))$. The embedding relation $\sqsubseteq$ on skeletons is defined as follows:

- $* \sqsubseteq *$
- $\alpha'_1 \rightarrow \alpha'_2 \sqsubseteq \alpha_1 \rightarrow \alpha_2 \quad$ if $\alpha'_i \sqsubseteq \alpha_i$ $(i = 1, 2)$
- $\alpha_i \sqsubseteq \alpha_1 \rightarrow \alpha_2 \quad$ for $i = 1, 2$.

A term $a(t_1, \ldots, t_n)$ is said to be well-formed with skeleton $\alpha$ if $\tau_0(a) = \alpha_1 \rightarrow \cdots \alpha_n \rightarrow \alpha$ and for each $i$, there exists $\alpha'_i$ such that $t_i$ is well-formed with $\alpha'_i$ and $\alpha'_i \sqsubseteq \alpha_i$. We denote by $T^{\sqsubseteq,\alpha}_{\tau}(\mathcal{F}, \mathcal{V})$ the set of all well-formed terms with skeleton $\alpha$. We also denote $wftype_{\tau}(t) = \alpha$ if $t \in T^{\sqsubseteq,\alpha}_{\tau}(\mathcal{F}, \mathcal{V})$, and denote by $T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$ the set of all well-formed terms.

We notice that any simply-typed term is a well-formed term, that is, $T_{\tau}(\mathcal{F}, \mathcal{V}) \subseteq T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$, but the equality does not hold in general. For example, letting $\tau(0) = Nat$, $\tau(Nil) = NatList$ and $\tau(Map) = (Nat \rightarrow Nat) \rightarrow NatList \rightarrow NatList$. Then $S(Nil)$, $Map(0, Nil) \in T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$, but $S(Nil)$, $Map(0, Nil) \notin T_{\tau}(\mathcal{F}, \mathcal{V})$.

## 3. Computability

In order to prove the termination of typed $\lambda$-calculus, the notion of computability was introduced by Tait [18] and Girard [6]. In the following sections, we will make use of the notion to prove the well-foundedness of path orders. This approach is also used in [8], [14].

Intuitivity, the notion of computability corresponds to termination as function. For instance, considering STRS $R = \{F(0) \to F(0)\}$. Then the term $F(0)$ is not terminating, but the term $F$ is terminating. We often call $F$ non-terminating function. The notion of computability handles this view point, i.e., the term $F$ is terminating but not computable.

A notion of computability is inductively defined on types with respect to the embedding relation $\sqsubseteq$ as follows:

**Definition 3.1:** Let $\alpha = \alpha_1 \to \cdots \to \alpha_n \to *$. A well-formed term $t \in T_\tau^{\sqsubseteq,\alpha}(\mathcal{F},\mathcal{V})$ is said to be computable with respect to $\gg$ if $t(t_1,\ldots,t_n)$ is terminating with respect to $\gg$ for all computable terms $t_i$ $(i = 1,\ldots,n)$ such that $wftype_\tau(t_i) \sqsubseteq \alpha_i$.

**Definition 3.2:** Let $\gg$ be a binary relation on well-formed terms $T_\tau^{\sqsubseteq}(\mathcal{F},\mathcal{V})$. $\gg$ is said to be a subject relation if $s \gg t \Rightarrow wftype_\tau(s) = wftype_\tau(t)$. $\gg$ has the supplement property if $s \gg t \Rightarrow s(u) \gg t(u)$ for any well-formed terms $s$, $t$ and $u$ such that $s(u)$ and $t(u)$ are well-formed.

Also not only approaches in [6], [18] but ones in [8], [14], four properties in the following lemma hold the key to computable approaches. Hence subject and supplement properties are essential in computable approaches so that clearly from the following lemma.

**Lemma 3.3:** Let $\gg$ be a subject relation on $T_\tau^{\sqsubseteq}(\mathcal{F},\mathcal{V})$ with the supplement property. Then the following properties with respect to $\gg$ hold.

(1) If there exists at least one computable well-formed term $u$ with $wftype_\tau(u) = *$, then any computable well-formed term is terminating.
(2) If $s$ is computable and $s \gg t$ then $t$ is computable.
(3) If $wftype_\tau(s) = *$ and $t$ is computable for any $t$ with $s \gg t$, then $s$ is computable.
(4) Let $t(t_1,\ldots,t_n) \in T_\tau^{\sqsubseteq,\alpha}(\mathcal{F},\mathcal{V})$. If $t$ and each $t_i$ are computable then $t(t_1,\ldots,t_n)$ is computable.

**Proof :**

(1) Let $t \in T_\tau^{\sqsubseteq,\alpha}(\mathcal{F},\mathcal{V})$ be a computable term with $\alpha = \alpha_1 \to \cdots \to \alpha_n \to *$. Assume that there exists an infinite decreasing sequence $t \equiv t_0 \gg t_1 \gg \cdots$. Since $\gg$ is a subject relation, $wftype_\tau(t_i) = \alpha$ for each $i$. Thanks to the assumption, letting $u$ be a computable term with $wftype_\tau(u) = *$. From the definition, $* \sqsubseteq \alpha_i$ for each $i$, hence each $t_j(u,\ldots,u)$ is well-formed with $wftype_\tau(t_j(u,\ldots,u)) = *$. It follows from the supplement property that $t_0(u,\ldots,u) \gg t_1(u,\ldots,u) \gg \cdots$. It is a contradiction with the computability of $t$.
(2) Assume that $t \in T_\tau^{\sqsubseteq,\alpha}(\mathcal{F},\mathcal{V})$ is not computable, where $\alpha = \alpha_1 \to \cdots \to \alpha_n \to *$. Then there exist computable terms $u_i$ with $wftype_\tau(u_i) \sqsubseteq \alpha_i$ $(i = 1,\ldots,n)$ such that $t(u_1,\ldots,u_n)$ is not terminating. Since $\gg$ is a subject relation, $wftype_\tau(s) = \alpha$ and hence $wftype_\tau(s(u_1,\ldots,u_n)) = *$. From the supplement property, $s(u_1,\ldots,u_n) \gg t(u_1,\ldots,u_n)$. Thus $s(u_1,\ldots,u_n)$ is not terminating. It is a contradiction with the computability of $s$.
(3) Assume that $s$ is not computable, that is, not terminating. Then there exists an infinite decreasing sequence $s \gg t_1 \gg t_2 \gg \cdots$. From the assumption, $t_1$ is computable. Since $\gg$ is a subject relation, $wftype_\tau(t_1) = *$. Hence $t_1$ is terminating. It is a contradiction.
(4) Let $\alpha = \alpha_1 \to \cdots \to \alpha_m \to *$. Assume that $t(t_1,\ldots,t_n)$ is not computable. Then there exist computable well-formed terms $u_i$ for each $i$ such that $t(t_1,\ldots,t_n,u_1,\ldots,u_m) \in T_\tau^{\sqsubseteq,*}(\mathcal{F},\mathcal{V})$ is not terminating. It is a contradiction with the computability of $t$. □

## 4. Recursive Path Order

In the previous section we have seen that the subject and the supplement properties are essential in computable approaches. Based on this viewpoint, we design a recursive path order by giving the subject and the supplement properties to original one [5].

**Definition 4.1** (Recursive Path Order):
A precedence $\rhd$ is a strict order on $\mathcal{F}$.

For well-formed terms $s \equiv a(s_1,\ldots,s_n)$ and $t \equiv a'(t_1,\ldots,t_m)$, we have $s >_{rpo} t$ if $wftype_\tau(s) = wftype_\tau(t)$ and satisfying one of the following rules:

(RPO1) $wftype_\tau(s) = *$, $a \rhd a'$, and for all $j$, either $s >_{rpo} t_j$ or $\exists$i. $s_i \geq_{rpo} t_j$,
(RPO2) $a = a'$ and $\{s_1,\ldots,s_n\} >_{rpo}^{mul} \{t_1,\ldots,t_m\}$, or
(RPO3) there exists a number $k$ such that $\exists i.\ s_i \geq_{rpo} a'(t_1,\ldots,t_k)$ and $\forall j > k.\ \exists i_j.\ s_{i_j} \geq_{rpo} t_j$,

where $\geq_{rpo}$ is defined as $>_{rpo} \cup \equiv$.

In the definition above, the restriction $wftype_\tau(s) = wftype_\tau(t)$ guarantees that $>_{rpo}$ is a subject relation, and the rule (RPO3) and the restriction $wftype_\tau(s) = *$ in (RPO1) give $>_{rpo}$ the supplement property.

**Lemma 4.2:** $>_{rpo}$ is a subject relation.

**Proof :** Trivial. □

The subject property is essential for designing $>_{rpo}$. For example, letting $F \rhd G$, $\tau(F) = \alpha \to \alpha \to \alpha$ and $\tau(G) = (\alpha \to \alpha) \to \alpha \to \alpha$. If we omit the restriction $wftype_\tau(s) = wftype_\tau(t)$ whenever $s >_{rpo} t$, then we obtain $F(x,y) >_{rpo} G(F(x),y)$ by (RPO1), and $G(F(x),y) >_{rpo} F(x,y)$ by (RPO3).

**Lemma 4.3:** $>_{rpo}$ has the supplement property.

**Proof :** Let $s \equiv a(s_1,\ldots,s_n) >_{rpo} a'(t_1,\ldots,t_m) \equiv t$, $wftype_\tau(s) = \alpha \to \beta$ and $u \in T_\tau^{\sqsubseteq,\alpha'}(\mathcal{F},\mathcal{V})$ for some $\alpha' \sqsubseteq \alpha$. We prove that $s(u) >_{rpo} t(u)$ by induction on the definition of $>_{rpo}$. According to the rule applied to $s >_{rpo} t$, we distinguish the following cases:

RPO1: This is not the case, because $wftype_\tau(s) \neq *$.

RPO2: Then $\{s_1, \ldots, s_n\} >^{mul}_{rpo} \{t_1, \ldots, t_m\}$. Since $\{s_1, \ldots, s_n, u\} >^{mul}_{rpo} \{t_1, \ldots, t_m, u\}$, we have $s(u) >_{rpo} t(u)$ by (RPO2).

RPO3: Then there exists a number $k$ such that $\exists i.\ s_i \geq_{rpo} a'(t_1, \ldots, t_k)$ and $\forall j > k.\ \exists i_j.\ s_{i_j} \geq_{rpo} t_j$. We have also $s(u) \equiv a(s_1, \ldots, s_n, u) >_{rpo} a'(t_1, \ldots, t_m, u) \equiv t(u)$ by (RPO3), because of $u \geq_{rpo} u$. □

Thanks to the subject and the supplement properties, we can obtain the following key lemma correspond to Lemma 3.3.

**Lemma 4.4:** In well-formed terms, the following properties with respect to $>_{rpo}$ hold.

(1) Any computable well-formed term is terminating.
(2) If $s$ is computable and $s >_{rpo} t$ then $t$ is computable.
(3) If $wftype_\tau(s) = *$ and $t$ is computable for any $t$ with $s >_{rpo} t$, then $s$ is computable.
(4) Let $t(t_1, \ldots, t_n) \in T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$. If $t$ and each $t_i$ are computable then $t(t_1, \ldots, t_n)$ is computable.

**Proof :** From Lemmata 4.2 and 4.3, $>_{rpo}$ has the subject and the supplement properties. We recall that we assume the existence of a basic-typed variable $x$, i.e., $wftype_\tau(x) = *$. From the definition of $>_{rpo}$, the variable is minimal with respect to $>_{rpo}$. Therefore this lemma follows from Lemma 3.3. □

By using four properties above, we prove the well-foundedness of $>_{rpo}$ on well-formed terms by proving that any well-formed term is computable. The proof is constructed by the following two lemmata. We note that the following lemma makes use of induction on a triple as in [8].

**Lemma 4.5:** Let $s \equiv a(s_1, \ldots, s_n) >_{rpo} a'(t_1, \ldots, t_m) \equiv t$ such that $wftype_\tau(s) = *$. If any $s_i$ is computable with respect to $>_{rpo}$, then so is $t$.

**Proof :** The proof proceeds by induction on triples $(a, \{s_1, \ldots, s_n\}, |t|)$ with respect to the order $\gg$ defined as the lexicographic combination of $\rhd$, $>^{mul}_{rpo}$ and the usual strict order over natural numbers. We notice that $>^{mul}_{rpo}$ is well-founded on multisets of computable terms, because of Lemma 4.4 (1).

According to the rule applied to $s >_{rpo} t$, we distinguish the following cases:

RPO1: We first prove that any $t_j$ is computable. In the case of $s_i \geq_{rpo} t_j$, $t_j$ is computable by Lemma 4.4 (2). In the case of $s >_{rpo} t_j$, $t_j$ is computable by the induction hypothesis, because $(a, \{s_1, \ldots, s_n\}, |t|) \gg (a, \{s_1, \ldots, s_n\}, |t_j|)$. Hence any $t_j$ is computable.
From Lemma 4.2, $wftype_\tau(t) = wftype_\tau(s) = *$. Thanks to Lemma 4.4 (3) and $wftype_\tau(t) = *$, it suffices to show that any $u$ is computable whenever $t >_{rpo} u$. It follows from the induction hypothesis, because $(a, \{s_1, \ldots, s_n\}, |t|) \gg (a', \{t_1, \ldots, t_m\}, |u|)$.

RPO2: Since $\{s_1, \ldots, s_n\} >^{mul}_{rpo} \{t_1, \ldots, t_m\}$, for any $t_j$ there exists $s_i$ such that $s_i \geq_{rpo} t_j$. From Lemma 4.4 (2), any $t_j$ is computable.
As in the case (RPO1), it suffices to show that any $u$ is computable whenever $t >_{rpo} u$. It follows from the induction hypothesis, because $(a, \{s_1, \ldots, s_n\}, |t|) \gg (a', \{t_1, \ldots, t_m\}, |u|)$.

RPO3: Suppose that $s_i \geq_{rpo} a'(t_1, \ldots, t_k)$ and $s_{i_j} \geq_{rpo} t_j$ for all $j > k$. From Lemma 4.4 (2), $a'(t_1, \ldots, t_k)$ and each $t_j$ $(j > k)$ are computable. From Lemma 4.4 (4), $t \equiv a'(t_1, \ldots, t_k, t_{k+1}, \ldots, t_m)$ is computable. □

**Lemma 4.6:** $>_{rpo}$ is well-founded in well-formed terms $T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$.

**Proof :** By Lemma 4.4 (1), any computable term is terminating in well-formed terms. Hence, it suffices to show that any well-formed term $t$ is computable with respect to $>_{rpo}$. The proof proceeds by induction on $|t|$. Let $t \equiv a(t_1, \ldots, t_m)$ and $wftype_\tau(t) = \alpha_1 \to \cdots \to \alpha_n \to *$.

Any $t_i$ is computable by the induction hypothesis. Assume that $t$ is not computable. Then there exist computable terms $u_i$ such that $t(u_1, \ldots, u_n)$ is a non-terminating well-formed term. Let $v$ be an arbitrary term such that $a(t_1, \ldots, t_m, u_1, \ldots, u_n) >_{rpo} v$. From Lemma 4.5, $v$ is computable. Hence $t(u_1, \ldots, u_n)$ is computable by Lemma 4.4 (3). From the definition, $t(u_1, \ldots, u_n)$ is terminating. It is a contradiction. □

In order to prove that $>_{rpo}$ is a reduction order, the monotonicity and the stability are shown only remains. These properties guarantees that $>_{rpo}$ can simulate the reduction relation, i.e., $s \underset{R}{\to} t \Rightarrow s >_{rpo} t$ if $l >_{rpo} r$ for all $l \to r \in R$.

**Lemma 4.7:** $>_{rpo}$ is monotonic in $T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$.

**Proof :** Let $s >_{rpo} t$. It is trivial that $wftype_\tau(C[s]) = wftype_\tau(C[t])$ for any context $C[\,]$. We prove $C[s] >_{rpo} C[t]$ by structural induction on $C[\,]$. The case of $C[\,] \equiv \square$ is trivial. Suppose that $C[\,] \equiv a(\ldots, C'[\,], \ldots)$. From the induction hypothesis, $C'[s] >_{rpo} C'[t]$. Hence $\{\ldots, C'[s], \ldots\} >^{mul}_{rpo} \{\ldots, C'[t], \ldots\}$. We obtain $C[s] >_{rpo} C[t]$ by (RPO2). □

In the proof above, we use the fact that $\square$ is of a basic type.

**Lemma 4.8:** $>_{rpo}$ is stable in $T^{\sqsubseteq}_{\tau}(\mathcal{F}, \mathcal{V})$.

**Proof :** Let $s \equiv a(s_1, \ldots, s_n) >_{rpo} a'(t_1, \ldots, t_m) \equiv t$. It is trivial that $wftype_\tau(s\theta) = wftype_\tau(t\theta)$ for any substitution $\theta$. We prove $s\theta >_{rpo} t\theta$ by induction on $|s| + |t|$. According to the rule applied to $s >_{rpo} t$, we distinguish the following cases:

RPO1: From the induction hypothesis, for all $j$ either $s\theta >_{rpo} t_j\theta$ or $\exists i.\ s_i\theta \geq_{rpo} t_j\theta$. Hence we obtain $s\theta >_{rpo} t\theta$ by (RPO1).

RPO2: Let $a\theta = a''(\vec{u_i})$. From the induction hypothesis, $\{s_1\theta, \ldots, s_n\theta\} >^{mul}_{rpo} \{t_1\theta, \ldots, t_m\theta\}$. Hence $\{\vec{u_i}, s_1\theta, \ldots, s_n\theta\} >^{mul}_{rpo} \{\vec{u_i}, t_1\theta, \ldots, t_m\theta\}$. Since $root(s\theta) = a'' = root(t\theta)$, we obtain $s\theta >_{rpo} t\theta$ by (RPO2).

RPO3: Suppose that $\exists i.\ s_i \geq_{rpo} a'(t_1,\ldots,t_k)$ and $\forall j > k.\ \exists i_j.\ s_{i_j} \geq_{rpo} t_j$ for some $k$. From the induction hypothesis, $s_i\theta \geq_{rpo} a'(t_1,\ldots,t_k)\theta$ and $s_{i_j}\theta \geq_{rpo} t_j\theta$. Hence $s\theta \equiv a''(\vec{u_i}, s_1\theta,\ldots,s_n\theta) >_{rpo} (a'(t_1,\ldots,t_k)\theta)(t_{k+1}\theta,\ldots,t_m\theta) \equiv t\theta$ by (RPO3), where $a\theta = a''(\vec{u_i})$. □

**Theorem 4.9:** $>_{rpo}$ is a reduction order in well-formed terms $T^{\sqsubseteq}_{\tau}(\mathcal{F},\mathcal{V})$.

**Proof :** From Lemmata 4.6, 4.7 and 4.8. □

**Corollary 4.10:** $>_{rpo}$ is a reduction order and $>^{+}_{rpo}$ is a reduction strict order in simply-typed terms $T_{\tau}(\mathcal{F},\mathcal{V})$.

We notice that $>_{rpo}$ is not transitive. For example, letting $G \rhd F$, $\tau(F) = \tau(G) = (\alpha \to \alpha) \to \alpha \to \alpha$, $\tau(f) = \alpha \to \alpha$ and $\tau(x) = \alpha$, then $F(G(f),x) >_{rpo} G(f,x) >_{rpo} F(f,f(x))$ but $F(G(f),x) \not>_{rpo} F(f,f(x))$.

Note that the transitivity is unnecessary for proving termination (refer to Theorem 2.3).

**Example 4.11:** Let $R_1$ be the following STRS

$$\begin{cases} Map(f,Nil) & \to & Nil \\ Map(f,C(x,xs)) & \to & C(f(x),Map(f,xs)) \end{cases}$$

We define the precedence by $Map \rhd C$. Then $Map(f,C(x,xs)) >_{rpo} f(x)$ by (RPO3), and $Map(f,C(x,xs)) >_{rpo} Map(f,xs)$ by (RPO2). Thus $Map(f,C(x,xs)) >_{rpo} C(f(x),Map(f,xs))$ by (RPO1). We have also $Map(f,Nil) >_{rpo} Nil$ by (RPO3). Hence $R_1$ is terminating.

## 5. Lexicographic Path Order

We also design a lexicographic path order by giving the subject and the supplement properties to original one [11].

**Definition 5.1** (Lexicographic Path Order): A precedence $\rhd$ is a strict order on $\mathcal{F}$.

For well-formed terms $s \equiv a(s_1,\ldots,s_n)$ and $t \equiv a'(t_1,\ldots,t_m)$, we have $s >_{lpo} t$ if $wftype_{\tau}(s) = wftype_{\tau}(t)$ and satisfying one of the following rules:

(LPO1) $wftype_{\tau}(s) = *$, $a \rhd a'$, and for all $j$, either $s >_{lpo} t_j$ or $\exists$i. $s_i \geq_{lpo} t_j$,

(LPO2) $a = a'$, $[s_1,\ldots,s_n] >^{lex}_{lpo} [t_1,\ldots,t_m]$, and for all $j$, either $s >_{lpo} t_j$ or $\exists$i. $s_i \geq_{lpo} t_j$, or

(LPO3) there exists a number $k$ such that $\exists i.\ s_i \geq_{lpo} a'(t_1,\ldots,t_k)$ and $\forall j > k.\ \exists i_j.\ s_{i_j} \geq_{lpo} t_j$,

where $\geq_{lpo}$ is defined as $>_{lpo} \cup \equiv$.

**Theorem 5.2:** $>_{lpo}$ is a reduction order in well-formed terms $T^{\sqsubseteq}_{\tau}(\mathcal{F},\mathcal{V})$.

We omit the proof, because all proofs are essentially similar as proofs of the recursive path order.

**Corollary 5.3:** $>_{lpo}$ is a reduction order and $>^{+}_{lpo}$ is a reduction strict order in simply-typed terms $T_{\tau}(\mathcal{F},\mathcal{V})$.

By using $>_{lpo}$, we can also prove the termination of STRS $R_1$ in Example 4.11.

## 6. Dependency Pair and Argument Filtering Methods

The notion of dependency pairs in first-order TRSs was introduced by Arts and Giesl [1]–[3]. The notion was extended to higher-order systems by Sakai, Watanabe and Sakabe [15], and by Kusakari [12]. In the dependency pair method, weak reduction orders play an important role instead of reduction orders. To design weak reduction orders, Arts and Giesl introduced the argument filtering method, which is designed by eliminating unnecessary subterms [3]. After that the method was extended to STRSs by Kusakari [12]. First we introduce some results for the dependency pair and the argument filtering methods in STRSs [12].

**Definition 6.1:** The set $DF(R)$ of defined symbols in $R$ is defined as $\{root(l) \mid l \to r \in R\}$. $\mathcal{F}^{\#} = \{F^{\#} \mid F \in \mathcal{F}\}$ is a set of marked symbols disjoint from $\mathcal{F} \cup \mathcal{V}$. We define the root-marked term by $(F(t_1,\ldots,t_n))^{\#} = F^{\#}(t_1,\ldots,t_n)$. If $root(t) \in \mathcal{V}$ then we identify $t^{\#}$ with $t$. A pair $\langle u^{\#}, v^{\#}\rangle$ of terms is a dependency pair of STRS $R$ if there exists a rule $u \to C[v] \in R$ such that $root(v) \in DF(R) \cup \mathcal{V}$, $v$ is of a basic type, and $v$ itself is not a variable. We denote by $DP(R)$ the set of dependency pairs of $R$.

**Definition 6.2:** We define that a term $l$ is said to be a pattern if any variable occurrences in $l$ is at a leaf position, that is, $n = 0$ whenever $l \equiv C[x(u_1,\ldots,u_n)]$ and $x \in \mathcal{V}$. A STRS $R$ is said to be a pattern STRS if the left-hand side $l$ for any rule $l \to r$ in $R$ is a pattern, and a binary relation $\gg$ over terms is said to be stable for pattern if $l \gg r \Rightarrow l\theta \gg r\theta$ for any $l$ and $r$ such that $l$ is a pattern.

In the following, we consider pattern STRSs, because the argument filtering method in [12] requires the restriction by pattern.

**Definition 6.3:** A pair $(\gtrsim, >)$ of binary relations on $T_{\tau}(\mathcal{F},\mathcal{V})$ is said to be a reduction pair for pattern STRSs if it satisfies the following conditions:

- $\gtrsim$ is monotonic and stable for pattern,
- $>$ is well-founded and stable for pattern,
- $\gtrsim \cdot > \ \subseteq\ >$ or $> \cdot \gtrsim \subseteq\ >$,
- $\gtrsim$ satisfies the marked condition for $DF(R)$[†] ($v \gtrsim v^{\#}$ if $root(v) \in DF(R)$ and $\tau(v) \in \mathcal{B}$).

Specially, a binary relation $\gtrsim$ is said to be a weak reduction order for pattern STRSs if $(\gtrsim, >)$ is a reduction pair for pattern STRSs, where $>$ is the strict part of $\gtrsim$.

**Proposition 6.4:** [12] Let $R$ be a pattern STRS and $(\gtrsim, >)$ be a reduction pair for pattern STRSs. If $R \subseteq \gtrsim$ and $DP(R) \subseteq >$ then $R$ is terminating.

**Definition 6.5:** An argument filtering function is a function $\pi$ such that for any $F \in \mathcal{F}$, $\pi(F)$ is a list of positive integers $[i_1,\ldots,i_k]$ with $i_1 < \cdots < i_k \leq n$, where

[†]This condition is slightly modified from the condition in [12].

$\tau(F) = \alpha_1 \to \cdots \alpha_n \to \beta$ and $\beta \in \mathcal{B}$. $\pi(F)^{\leq n}$ denotes the maximal sub-list $[i_1, \ldots, i_m]$ of $\pi(F)$ such that $i_m \leq n$. We can naturally extend $\pi$ over terms as follows:

$$\begin{cases} \pi(a(t_1,\ldots,t_n)) = a(\pi(t_1),\ldots,\pi(t_n)) \\ \qquad\qquad\qquad \text{if } a \in \mathcal{V} \\ \pi(a(t_1,\ldots,t_n)) = a(\pi(t_{i_1}),\ldots,\pi(t_{i_m})) \\ \qquad \text{if } a \in \mathcal{F} \text{ and } \pi(a)^{\leq n} = [i_1,\ldots,i_m] \end{cases}$$

For given argument filtering function $\pi$ and binary relation $>$, we define $s \gtrsim^\pi t$ by $\pi(s) \geq \pi(t)$. We hereafter assume that if $\pi(F)$ is not defined explicitly then it is intended to be $[1, \ldots, n]$, where $\tau(F) = \alpha_1 \to \cdots \alpha_n \to \beta$ and $\beta \in \mathcal{B}$.

We notice that for each $> \in \{>_{rpo}, >_{lpo}\}$ the strict part $>^\pi$ of $\gtrsim^\pi$ can be directly defined as $s >^\pi t \iff \pi(s) > \pi(t)$, because $>$ is irreflexive, which follows from the well-foundedness.

**Proposition 6.6:** [12] Let $>$ be a reduction order in $T(\mathcal{F}, \mathcal{V})$. If $>$ has the deletion property[†] and $\gtrsim^\pi$ satisfies the marked condition for $DF(R)$, then $\gtrsim^\pi$ is a weak reduction order for pattern STRSs.

Unlike path orders in [12], definitions of path orders in this paper dependent on type attachment. Hence we introduce a type attachment $\tau^\pi$ after argument filtering.

**Definition 6.7:** For each type attachment $\tau$, we define $\tau^\pi$ as $\tau^\pi(x) = \tau(x)$ for all $x \in \mathcal{V}$; $\tau^\pi(F) = \alpha_{i_1} \to \cdots \alpha_{i_k} \to \beta$ if $\pi(F) = [i_1, \ldots, i_k]$ and $\tau(F) = \alpha_1 \to \cdots \alpha_n \to \beta$ with $\beta \in \mathcal{B}$.

**Lemma 6.8:** If $t \in T^\alpha_\tau(\mathcal{F}, \mathcal{V})$ then $\pi(t) \in T^{\sqsubseteq, \alpha'}_{\tau^\pi}(\mathcal{F}, \mathcal{V})$ for some $\alpha' \sqsubseteq \alpha$.

**Proof :** The proof proceeds by induction on $|t|$. Let $t \equiv a(t_1, \ldots, t_n) \in T^\alpha_\tau(\mathcal{F}, \mathcal{V})$, $\tau(a) = \alpha_1 \to \cdots \to \alpha_n \to \alpha$ and $\alpha = \alpha_{n+1} \to \cdots \to \alpha_m \to \beta$ with $\beta \in \mathcal{B}$.

From the induction hypothesis, for each $i \leq n$, there exists $\alpha'_i$ such that $\pi(t_i) \in T^{\sqsubseteq, \alpha'_i}_{\tau^\pi}(\mathcal{F}, \mathcal{V})$ and $\alpha'_i \sqsubseteq \alpha_i$. In the case of $a \in \mathcal{V}$, it is trivial that $\pi(t) = a(\pi(t_1), \ldots, \pi(t_n)) \in T^{\sqsubseteq, \alpha}_{\tau^\pi}(\mathcal{F}, \mathcal{V})$. Suppose that $a \in \mathcal{F}$, $\pi(a) = [i_1, \ldots, i_l]$ and $\pi(a)^{\leq n} = [i_1, \ldots, i_k]$. Then $\pi(t) = a(\pi(t_{i_1}), \ldots, \pi(t_{i_k}))$. Hence $\pi(t) \in T^{\sqsubseteq, \gamma}_{\tau^\pi}(\mathcal{F}, \mathcal{V})$ and $\gamma = \alpha_{i_{k+1}} \to \cdots \to \alpha_{i_l} \to \beta$. Since $\gamma \sqsubseteq \alpha_{n+1} \to \cdots \to \alpha_m \to \beta = \alpha$, the claim holds. □

Unfortunately, our recursive and lexicographic path orders do not have the deletion property, because $a(\ldots, u, \ldots)$ and $a(\ldots, \ldots)$ have different type even if both terms are well-formed.

On the other hand, the deletion property and the restriction by pattern are required only for stability. Hence, combining with Lemma 6.8 and Proposition 6.4, we are able to reformulate Proposition 6.6.

**Theorem 6.9:** Let $R$ be a STRS and $>$ be a reduction order on $T^{\sqsubseteq}_{\tau^\pi}(\mathcal{F}, \mathcal{V})$. Suppose that the following properties hold:

- $\gtrsim^\pi$ has the marked condition for $DF(R)$, i.e., $v \gtrsim v^\#$ if $root(v) \in DF(R)$ and $\tau(v) \in \mathcal{B}$.
- $\gtrsim^\pi$ are stable for $R$, i.e., for all $l \to r \in R$, $l \gtrsim^\pi r \Rightarrow l\theta \gtrsim^\pi r\theta$.
- $>^\pi$ are stable for $DP(R)$, i.e., for all $\langle u, v\rangle \in DP(R)$, $u >^\pi v \Rightarrow u\theta >^\pi v\theta$.

If $R \subseteq \gtrsim$ and $DP(R) \subseteq >$ then $R$ is terminating.

In this paper, we have the general assumption $\tau(x) = \tau(\theta(x))$ for each $x$, however this does not guarantee $wftype_{\tau^\pi}(\pi(t)) = wftype_{\tau^\pi}(\pi(t\theta))$.

**Lemma 6.10:**

- $wftype_{\tau^\pi}(\pi(t)) = wftype_{\tau^\pi}(\pi(t\theta))$ if $root(t) \in \mathcal{F}$
- $wftype_{\tau^\pi}(\pi(t)) \sqsupseteq wftype_{\tau^\pi}(\pi(t\theta))$ if $root(t) \in \mathcal{V}$

**Proof :** It is a directly consequence from definitions of *wftype* and $\pi$ over terms. □

In general, $wftype_{\tau^\pi}(\pi(t)) = wftype_{\tau^\pi}(\pi(t\theta))$ does not hold. For example, letting $t = f(H)$, $\theta(f) = F$, $\tau(f) = \tau(F) = (\alpha \to \alpha) \to \alpha \to \alpha$, $\tau(H) = \alpha \to \alpha$ and $\pi(F) = [1]$, then $wftype_{\tau^\pi}(\pi(t)) = * \to *$ and $wftype_{\tau^\pi}(\pi(t\theta)) = wftype_{\tau^\pi}(F(H)) = *$. This problem destroys the stability, because $f(H) >^\pi_{rpo} H$ but $F(H) \not>^\pi_{rpo} H$. Hence we need a suitable restriction.

**Lemma 6.11:** Let $\pi$ be an argument filtering function and $R$ be a STRS satisfying the following condition:

(P) Let $l \to r \in R$, $\pi(l) \equiv a(\pi(l_1), \ldots, \pi(l_n))$ and $x \in Var(\pi(l))$. If $\tau(x) \notin \mathcal{B}$ then for each $i$ either $x \equiv l_i$ or $x \notin Var(\pi(l_i))$.

Then $>^\pi_{rpo}$, $\gtrsim^\pi_{rpo}$, $>^\pi_{lpo}$ and $\gtrsim^\pi_{lpo}$ are stable for $R \cup DP(R)$.

**Proof :** It is easily shown by induction on $s$ that $\pi(s\theta) \equiv \pi(t\theta)$ for any $s$ and $t$ such that $\pi(s) \equiv \pi(t)$. Hence it suffices to show the case of $>^\pi_{rpo}$ and $>^\pi_{lpo}$. Moreover we omit the case of $>^\pi_{lpo}$, because the proof is essentially similar as proof of the case $>^\pi_{rpo}$.

Let $\langle u, v\rangle \in R \cup DP(R)$, $s \equiv a(s_1, \ldots, s_n)$, $t \equiv a'(t_1, \ldots, t_m)$, $\pi(s) \equiv a(\pi(s_{i_1}), \ldots, \pi(s_{i_{n'}}))$ and $\pi(t) \equiv a'(\pi(t_{j_1}), \ldots, \pi(t_{j_{m'}}))$. Suppose that $\pi(u) >_{rpo} \pi(v)$ and $\pi(s) >_{rpo} \pi(t)$ is a sub-proof of $\pi(u) >_{rpo} \pi(v)$. We prove $\pi(s\theta) >_{rpo} \pi(t\theta)$ by induction on $|s| + |t|$.

In the case of $a, a' \in \mathcal{F}$, thanks to Lemma 6.10, $\pi(s\theta) >_{rpo} \pi(t\theta)$ can be proved as similar to Lemma 4.8.

In the case of $a \in \mathcal{V}$, $\pi(s) \equiv a >_{rpo} \pi(t)$ from the condition (P). It is a contradiction.

Suppose that $a \in \mathcal{F}$ and $a' \in \mathcal{V}$. Then (RPO3) is only applicable to $\pi(s) >_{rpo} \pi(t)$. Let $k$ be a number such that $\exists i_p.\ \pi(s_{i_p}) \geq_{rpo} a'(\pi(t_{i_1}), \ldots, \pi(t_{i_k}))$ and $\forall j_q > i_k.\ \exists i_{j_q}.\ \pi(s_{i_{j_q}}) \geq_{rpo} \pi(t_{j_q})$. In the case of $\tau(a') \in \mathcal{B}$, $\pi(s\theta) >_{rpo} \pi(t\theta) \equiv \pi(a'\theta)$ is trivially holds. Suppose that $\tau(a') \notin \mathcal{B}$. From the condition (P), $\pi(s) \equiv \pi(u)$, $s_{i_p} = a'$ and $k = 0$. Hence $\pi(s_{i_p}\theta) \equiv \pi(a'\theta)$. From the induction hypothesis, $\pi(s_{i_{j_q}}\theta) \geq_{rpo} \pi(t_{j_q}\theta)$. Since $wftype_{\pi^\tau}(\pi(u)) = wftype_{\pi^\tau}(\pi(s)) = wftype_{\pi^\tau}(\pi(t)) = *$, we have $wftype_{\pi^\tau}(\pi(s\theta)) = wftype_{\pi^\tau}(\pi(t\theta)) = *$ by Lemma 6.10. Therefore we obtain $\pi(s\theta) >_{rpo} \pi(t\theta)$ by (RPO3). □

[†]In [12], this condition is missing for Corollary 7.7, however it is required in Theorem 6.8, on which the corollary is based. In fact, the deletion property is necessary to guarantee the stability for pattern.

Finally, we present an effective and efficient methods for proving termination. In general, $\gtrsim^{\pi}_{rpo}$ and $\gtrsim^{\pi}_{lpo}$ does not satisfy the marked condition. Hence we need a suitable restriction for precedence.

**Theorem 6.12:** Let $\pi$ be an argument filtering function and $R$ be a STRS satisfying the following condition:

(M) for any $F \in DF(R)$, either $F^{\#}$ is identified to $F$ or $F \rhd F^{\#}$ and $\pi(F) \supseteq \pi(F^{\#})$.

(P) Let $l \to r \in R$, $\pi(l) \equiv a(\pi(l_1), \ldots, \pi(l_n))$ and $x \in Var(\pi(l))$. If $\tau(x) \notin \mathcal{B}$ then for each $i$ either $x \equiv l_i$ or $x \notin Var(\pi(l_i))$.

For each $> \in \{>_{rpo}, >_{lpo}\}$, if $R \subseteq \gtrsim^{\pi}$ and $DP(R) \subseteq >^{\pi}$ then $R$ is terminating.

**Proof :** Thanks to Theorem 6.9 and Lemma 6.11, it suffices to show that the condition (M) guarantees the marked condition for $DF(R)$. Let $v \equiv a(v_1, \ldots, v_n)$ be a well-formed term such that $a \in DF(R)$ and $\tau(v) \in \mathcal{B}$. Since $wftype_{\tau}(v) = wftype_{\tau}(v^{\#}) = *$, it follows from Lemma 6.8 that $wftype_{\tau^{\pi}}(\pi(v)) = wftype_{\tau^{\pi}}(\pi(v^{\#})) = *$. Hence $\pi(v) \geq_{rpo} \pi(v^{\#})$ and $\pi(v) \geq_{lpo} \pi(v^{\#})$. □

**Example 6.13:** Let $\tau(F) = (\alpha \to \alpha) \to \alpha \to \alpha$, $\tau(G) = \tau(f) = \alpha \to \alpha$ and $\tau(x) = \alpha$. We define STRS $R_2$ as follows:

$$R_2 = \{F(f, F(G, x)) \to F(f, G(f(F(G, x))))\}$$

Then there exist three dependency pairs:

$$\langle F^{\#}(f, F(G, x)), F^{\#}(f, G(f(F(G, x))))\rangle$$
$$\langle F^{\#}(f, F(G, x)), f(F(G, x))\rangle$$
$$\langle F^{\#}(f, F(G, x)), F^{\#}(G, x)\rangle$$

Suppose that $\pi(G) = [\,]$ and $F^{\#}$ is identified to $F$. Then for the rule

$$\begin{aligned}\pi(F(f, F(G, x))) &\equiv F(f, F(G, x))\\ &>_{rpo} F(f, G)\\ &\equiv \pi(F(f, G(f(F(G, x)))))\end{aligned}$$

and for all dependency pairs

$$\begin{aligned}\pi(F^{\#}(f, F(G, x))) &\equiv F^{\#}(f, F(G, x))\\ &>_{rpo} F^{\#}(f, G)\\ &\equiv \pi(F^{\#}(f, G(f(F(G, x)))))\\ \pi(F^{\#}(f, F(G, x))) &\equiv F^{\#}(f, F(G, x))\\ &>_{rpo} f(F(G, x))\\ &\equiv \pi(f(F(G, x)))\\ \pi(F^{\#}(f, F(G, x))) &\equiv F^{\#}(f, F(G, x))\\ &>_{rpo} F^{\#}(G, x)\\ &\equiv \pi(F^{\#}(G, x))\end{aligned}$$

Therefore $R_2$ is terminating.

**Example 6.14:** Let $R_3$ be the following STRS:

$$\begin{aligned}Add(x, 0) &\to x\\ Add(x, S(y)) &\to S(Add(x, y))\\ Sub(x, 0) &\to x\\ Sub(0, y) &\to 0\\ Sub(S(x), S(y)) &\to Sub(x, y)\\ G(0, y) &\to y\\ G(S(x), y) &\to 0\\ Div(0, S(y)) &\to 0\\ Div(S(x), S(y))&\\ &\to G(Sub(y, x), S(Div(Sub(x, y), S(y))))\\ Len(Nil) &\to 0\\ Len(C(x, xs)) &\to S(Len(xs))\\ Sum(Nil) &\to 0\\ Sum(C(x, xs)) &\to Add(x, Sum(xs))\\ Map(f, Nil) &\to Nil\\ Map(f, C(x, xs)) &\to C(f(x), Map(f, xs))\\ F(f, xs)&\\ &\to Div(Sum(Map(f, xs)), Len(xs))\end{aligned}$$

We suppose that $\mathcal{B} = \{Nat, NatList, Bool\}$, The function symbol $Map$ has the type $(Nat \to Nat) \to NatList \to NatList$, and other symbols have usual types. We notice that the normal form of $F(f, [x_1, \ldots, x_n])$ by $R_3$ correspond to $\frac{f(x_1)+\cdots+f(x_n)}{n}$. Then there exist 15 dependency pairs:

$$\begin{aligned}&\langle Add^{\#}(x, S(y)), Add^{\#}(x, y)\rangle\\ &\langle Sub^{\#}(S(x), S(y)), Sub^{\#}(x, y)\rangle\\ &\langle Div^{\#}(S(x), S(y)),\\ &\qquad G^{\#}(Sub(y, x), S(Div(Sub(x, y), S(y))))\rangle\\ &\langle Div^{\#}(S(x), S(y)), Sub^{\#}(y, x)\rangle\\ &\langle Div^{\#}(S(x), S(y)), Div^{\#}(Sub(x, y), S(y))\rangle\\ &\langle Div^{\#}(S(x), S(y)), Sub^{\#}(x, y)\rangle\\ &\langle Len^{\#}(C(x, xs)), Len^{\#}(xs)\rangle\\ &\langle Sum^{\#}(C(x, xs)), Add^{\#}(x, Sum(xs))\rangle\\ &\langle Sum^{\#}(C(x, xs)), Sum^{\#}(xs)\rangle\\ &\langle Map^{\#}(f, C(x, xs)), f(x)\rangle\\ &\langle Map^{\#}(f, C(x, xs)), Map^{\#}(f, xs)\rangle\\ &\langle F^{\#}(f, xs), Div^{\#}(Sum(Map(f, xs)), Len(xs))\rangle\\ &\langle F^{\#}(f, xs), Sum^{\#}(Map(f, xs))\rangle\\ &\langle F^{\#}(f, xs), Map^{\#}(f, xs)\rangle\\ &\langle F^{\#}(f, xs), Len^{\#}(xs)\rangle\end{aligned}$$

We suppose that $H^{\#}$ is identified to $H$ for all $H \in \mathcal{F}$. We define the argument filtering function $\pi$ by $\pi(Sub) = [1]$, and define the precedence $\rhd$ by $Add \rhd S \rhd 0$, $Div \rhd G$, $Div \rhd S \rhd Sub$, $Len \rhd S$, $Sum \rhd Add$, $Map \rhd C$ and $F \rhd D$ for any $D \in \{Div, Sum, Map, Len\}$. Then it is routine to check that $\pi(l) \geq_{rpo} \pi(r)$ for all $l \to r \in R_3$ and $\pi(u^{\#}) >_{rpo} \pi(v^{\#})$ for all $\langle u^{\#}, v^{\#}\rangle \in DP(R_3)$. Therefore $R_3$ is terminating.

Note that above STRSs $R_2$ and $R_3$ are not simply terminating, which is a difficult class for proving termination. In fact, not only path orders in [8], [14] but also our path orders cannot prove the termination of $R_2$ and $R_3$. When we try to prove the termination of non-simply terminating STRSs, the argument filtering method is very effective. Moreover $R_3$ cannot be proved by methods in [12], which has combined path orders with dependency and argument filtering methods, because path orders in [12] requires $Map$ is greatest with respect to $\rhd$ in order to orient rules for *Map*, however the requirement destroys orientation for the last rule.

## 7. Concluding Remarks

A reader may think that the supplement property can be given by adding the rule $(s >_{rpo} t \Rightarrow s(u) >_{rpo} t(u))$ to original one [5], that is, for well-formed terms $s \equiv a(s_1, \ldots, s_n)$ and $t \equiv a'(t_1, \ldots, t_m)$, we define $s >_{rpo} t$ if $wftype_\tau(s) = wftype_\tau(t)$ and satisfying one of the following rules:

(RPO'1) $a \rhd a'$, and for all $j$, either $s >_{rpo} t_j$ or $\exists$i. $s_i \geq_{rpo} t_j$,
(RPO'2) $a = a'$ and $\{s_1, \ldots, s_n\} >^{mul}_{rpo} \{t_1, \ldots, t_m\}$,
(RPO'3) $\exists i.\ s_i \geq_{rpo} t$, or
(RPO'4) $a(s_1, \ldots, s_{n-1}) >_{rpo} a'(t_1, \ldots, t_{m-1})$ and $s_n \geq_{rpo} t_m$.

We feel that the definition is correspond to ones in [8], [14]. However this definition is out of the framework in this paper, specifically, destroys the proof of Lemma 4.5. It is the reason why terms with functional type (ex. *Add*, *Add*(0)) are not accepted in the framework in [14], although they are accepted in STRSs. Note that terms *Add* and *Add*(0) are represented by $\lambda xy.Add(x, y)$ and $\lambda y.Add(0, y)$ in [8], [14]. We overcome the problem by adding the restriction $wftype_\tau(s) = *$ in (RPO1).

Unlike path orders in [14], our path orders can combine well with the dependency pair and the argument filtering methods. In fact, the recursive path order in [14] guarantees that $l >_{rpo} d \twoheadrightarrow_\beta r \Rightarrow \exists u.\ l^\sigma\!\downarrow >_{rpo} u \twoheadrightarrow_\beta r^\sigma\!\downarrow$. So we must delicately analyze relations between the premise and the consequence after argument filtering, in order to generate weak reduction orders by the argument filtering method. We feel that it is not so easy.

As a model of functional programs, the polymorphic types is more useful than simple types. It is easy to define polymorphic term rewriting systems (PTRSs) by using polymorphic types instead of simple types. Then we should change the definition of reduction relation. For example, letting PTRS $R = \{I(x) \to x,\ App(f, x) \to f(x)\}$. In the system, we have

$$I(f, x) \leftarrow App(I(f), x) \to App(f, x) \to f(x).$$

Hence $R$ is not confluent, which is too restrictive as model. In order to reduce $I(f, x)$ to $f(x)$ we need to apply the first rule in the suffix context $\square(x)$ which has the hole at the root position. Hence, in [12], we define the reduction relation as $s \underset{R}{\to} t$ iff $s \equiv C[S[l\theta]]$ and $t \equiv C[S[r\theta]]$, where $S[\,]$ is a suffix context. Fortunately, our path orders are closed to suffix contexts, that is, our path orders have the supplement property. On the other hand, path orders in [12] does not have the supplement property. Hence our path orders may gain an advantage over ones in [12], when we try to prove termination of PTRSs.

### References

[1] T. Arts, Automatically Proving Termination and Innermost Normalization of Term Rewriting Systems, Ph.D. Thesis, Univ. of Utrecht, 1997.
[2] T. Arts and J. Giesl, "Automatically proving termination where simplification orderings fail," TAPSOFT '97, LNCS 1214, pp.261–272, 1997.
[3] T. Arts and J. Giesl, "Termination of term rewriting using dependency pairs," Theor. Comput. Sci., vol.236, pp.133–178, 2000.
[4] F. Baader and T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
[5] N. Dershowitz, "Orderings for term-rewriting systems," Theor. Comput. Sci., vol.17, pp.279–301, 1982.
[6] J.-Y. Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, Ph.D. Thesis, University of Paris VII, 1972.
[7] J.-P. Jouannaud and A. Rubio, "Rewrite orderings for higher-order terms in $\eta$-long $\beta$-normal form and the recursive path ordering," Theor. Comput. Sci., vol.208, no.1/2, pp.33–58, 1998.
[8] J.-P. Jouannaud and A. Rubio, "The higher-order recursive path ordering," Proc. 14th IEEE Symposium on Logic in Computer Science, pp.402–411, IEEE Computer Society Press, 1999.
[9] M. Iwami and Y. Toyama, "Simplification ordering for higher-order rewrite systems," IPSJ Trans. Programming, vol.40, no.SIG 4 (PRO 3), pp.1–10, 1999.
[10] M. Iwami, Termination of Higher-Order Rewrite Systems, Ph.D. Thesis, Japan Advanced Institute of Science and Technology, 1999.
[11] S. Kamin and J.-J. Lévy, Two Generalizations of the Recursive Path Ordering, University of Illinois at Urbana-Champaign, Unpublished manuscript, 1980.
[12] K. Kusakari, "On proving termination of term rewriting systems with higher-order variables," IPSJ Trans. Programming, vol.42, no.SIG 7 (PRO 11), pp.35–45, July 2001.
[13] T. Nipkow, "Higher-order critical pairs," Proc. 6th IEEE Symp. Logic in Computer Science, pp.342–349, IEEE Computer Society Press, 1991.
[14] F. Raamsdonk, "On termination of higher-order rewriting," Proc. 12th Int. Conf. on Rewriting Techniques and Applications, LNCS 2051 (RTA2001), pp.261–275, 2001.
[15] M. Sakai, Y. Watanabe, and T. Sakabe, "An extension of dependency pair method for proving termination of higher-order rewrite systems," IEICE Trans. Inf. & Syst., vol.E84-D, no.8, pp.1025–1032, Aug. 2001.
[16] M. Sakai and K. Kusakari, "On new dependency pair method for proving termination of higher-order rewrite systems," The International Workshop on Rewriting in Proof and Computation (RPC '01), pp.176–187, 2001.
[17] M. Sakai and K. Kusakari, "On proving termination of higher-order rewrite systems by dependency pair technique," The First International Workshop on Higher-Order Rewriting (HOR '02), p.25, 2002.
[18] W.W. Tait, "Intensional interpretation of functionals of finite type," J. Symb. Log., vol.32, pp.198–212, 1967.

**Keiichirou Kusakari** received B.E. from Tokyo Institute of Technology in 1994, M.E. and D.E. from Japan Advanced Institute of Science and Technology (JAIST) in 1996 and 2000. From 2000 to 2003, he had been a research associate of Research Institute of Electrical Communication (RIEC), Tohoku University. He is currently an assistant professor of Graduate School of Information Science, Nagoya University. His research interests include term rewriting systems, program theory, and automated theorem proving. He is a member of IPSJ and JSSST.