Vol. 42 No. SIG 7(PRO 11) IPSJ Transactions on Programming 2001

*Regular Paper*

# On Proving Termination of Term Rewriting Systems with Higher-Order Variables

Keiichirou Kusakari†

We define term rewriting systems with higher-order variables, which are defined without a meta-language like the λ-calculus. On the other hand, our systems express actual functional programming languages like ML and Haskell. In this paper, we extend the recursive path order, the dependency pair and the argument filtering method to our systems. In a different framework, Nipkow also introduced higher-order rewriting systems (HRSs). However, it is more complicated and restricted to extend the recursive path order and the dependency pair method to HRSs. Moreover the argument filtering method cannot be designed essentially in HRSs. These fact proves a usefulness of our systems.

## 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 term rewriting systems themselves can be regarded as functional programming languages. For example, the following TRS defines addition of natural numbers represented by the constant 0 and the successor function $S$:

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

Termination of TRSs is in general an undecidable property. Nevertheless, it is often necessary to prove the termination for a particular system. To prove termination, we commonly design a reduction order by which all rules are ordered. The most important concept for designing reduction orders is the notion of simplification orders introduced by Dershowitz[5),6)]. Based on the notion, the recursive path order was introduced[6)]. On the other hand, proving termination by simplification orders has a theoretical limitation. In fact, there exist terminating TRSs that cannot be proved the termination by simplification orders. To prove the termination of such TRSs, Arts and Giesl introduced the notion of dependency pairs[2)]. 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)].

Unfortunately, TRSs cannot treat higher-order functions. For example, the $Map$-function, which is one of the most standard higher-order function in functional programs, is defined as follows:

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

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.

To introduce an application symbol may be a most simple way to express such systems. In the framework the $Map$-function is defined by the following first-order TRS:

$$\left\{\begin{array}{l} @(@(Map,f),[\,]) \;\rightarrow\; [\,] \\ @(@(Map,f),@(@(::,x),xs)) \\ \quad\rightarrow @(@(::,@(f,x)),@(@(Map,f),xs)) \end{array}\right.$$

However this formulation is quite useless in our motivation. In fact, any recursive path order fails to order the second rule, because all symbols occurred in non-leaf positions are the application symbol @. Hence path orderings cannot work well in this formulation.

On the other hand, Nipkow introduced higher-order rewriting systems (HRSs), which are rewriting systems on algebraic λ-terms, using the λ-calculus as a meta-language[14),15)]. Intuitively, algebraic λ-terms are simply-typed λ-terms in which occurrences of function symbols are permitted. For example, the $Map$-function in HRSs is defined as follows:

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

Jouannaud and Rubio gave a simple definition

† Research Institute of Electrical Communication, Tohoku University

of a recursive path order for HRSs by using the type structure[8], and the order was improved by Iwami and Toyama[11),12]. However, to keep the consistency between the term structure and the type structure, these order cannot simultaneously treat the $Map$-function over the type $Nat$ and functions with the type $NatList \to Nat$ like the *Length*-function defined by

$$\begin{cases} Length([\,]) & \to \quad 0 \\ Length(x :: xs) & \to \quad S(Length(xs)). \end{cases}$$

Moreover, Jouannaud and Rubio designed a recursive path order on polymorphic algebraic $\lambda$-terms[9]. Note that the well-foundedness of the order was proved by using the termination proof of the typed $\lambda$-calculus instead of simplification orders.

The notion of dependency pairs extends to HRSs by Watanabe, Kawaguti, Sakai, Sakabe and Inagaki[17]. However, the dependency pair method in HRSs essentially requires that weak reduction orders have the subterm property, which is very restricted in practice. In fact, we cannot design the argument filtering method in HRSs, because the argument filtering method eliminates several subterms.

In this paper, we define term rewriting systems with higher-order variables ($\mathrm{TRS}_{hv}$ ), which can naturally denote higher-order functions like the $Map$-function. Hence $\mathrm{TRS}_{hv}$s turn out to be more powerful than TRSs in expressing actual functional programming languages like ML[13),16] and Haskell[10]. Since we define $\mathrm{TRS}_{hv}$s without a meta-language like the $\lambda$-calculus, we can easily extend results in TRSs to $\mathrm{TRS}_{hv}$s . Moreover, since $\mathrm{TRS}_{hv}$s are designed on untyped terms, $\mathrm{TRS}_{hv}$s are applicable to arbitrary typed systems. To prove termination of $\mathrm{TRS}_{hv}$s , we first extend the recursive path order to $\mathrm{TRS}_{hv}$s in Section 4. Next, we extend the dependency pair method to $\mathrm{TRS}_{hv}$s in Section 5, and the argument filtering method to $\mathrm{TRS}_{hv}$s in Section 6. Finally, in Section 7, we restrict $\mathrm{TRS}_{hv}$s by simply-typed systems, and discuss with the previous methods.

## 2. Preliminaries

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

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 variables with $\mathcal{F} \cap \mathcal{V} = \emptyset$ denoted by $x, y, f, g, \ldots$. The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of terms constructed from $\mathcal{F}$ and $\mathcal{V}$ is the smallest set such that $a(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ whenever $a \in \mathcal{F} \cup \mathcal{V}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. If $n = 0$, we write $a$ instead of $a()$. We define $root(a(t_1, \ldots, t_n)) = a$. $Var(t)$ is the set of variables in $t$. A term is said to be a ground term if no variable occurs in the term. Identity of terms is denoted by $\equiv$. The size $|t|$ of a term $t$ is the number of function symbols and variables in $t$. A term position is a sequence of positive integers. We denote the empty sequence by $\varepsilon$. We recursively define $t|_p$ the subterm of $t$ at position $p$ as $t|_\varepsilon = t$ and $a(t_1, \ldots, t_n)|_{i \cdot p} = t_i|_p$.

A substitution is a mapping from variables to terms. A substitution over terms is defined as

- $a \in \mathcal{F}$
  $\theta(t) = a(\theta(t_1), \ldots, \theta(t_n))$,
- $a \in \mathcal{V}$ with $\theta(a) = a'(t'_1, \ldots, t'_k)$
  $\theta(t) = a'(t'_1, \ldots, t'_k, \theta(t_1), \ldots, \theta(t_n))$,

where $t \equiv a(t_1, \ldots, t_n)$. We write $t\theta$ instead of $\theta(t)$. A context is a term which has a special symbol $\Box$, called hole, at a leaf position. A term $C[t]$ denotes the result of placing $t$ in the hole of a context $C[\,]$. A suffix context is a term which has the symbol $\Box$ at the root position. A term $S[t]$ denotes the result of placing $t$ in the hole of a suffix context $S[\,]$, where this replacement is defined as similar to substitutions. For example, $S[Add(0)] \equiv Add(0, S(0))$ for $S[\,] \equiv \Box(S(0))$. A term $s$ is called a subterm of $t$ if $t \equiv C[S[s]]$ for some context $C[\,]$ and suffix context $S[\,]$. A subterm $s$ is called an immediate subterm of $t$ if $t$ has a form $a(\cdots, s, \cdots)$.

A rewrite rule is a pair of terms, written by $l \to r$, such that $root(l) \notin \mathcal{V}$ and $Var(l) \supseteq Var(r)$. A term rewriting system with higher-order variables ($\mathrm{TRS}_{hv}$ ) is a finite set of rules. A reduction relation $\underset{R}{\to}$ is defined as $s \underset{R}{\to} t$ iff $s \equiv C[S[l\theta]]$ and $t \equiv C[S[r\theta]]$ for some rule $l \to r \in R$, context $C[\,]$, suffix context $S[\,]$ and substitution $\theta$. For example, the $Map$-function is defined by the following $\mathrm{TRS}_{hv}$ $R_{Map}$:

$$\begin{cases} Map(f, [\,]) & \to \quad [\,] \\ Map(f, x :: xs) & \to \quad f(x) :: Map(f, xs) \end{cases}$$

Here $[\,]$ and $x :: xs$ are syntax sugars for terms $Nil$ and $Cons(x, xs)$. We hereafter use these syntax sugars through the paper. In $R_{Map}$, we have the following reduction relation sequence.

$$\begin{aligned} & Map(S, S(0) :: 0 :: [\,]) \\ & \underset{R}{\to} S(S(0)) :: Map(S, 0 :: [\,]) \\ & \underset{R}{\to} S(S(0)) :: S(0) :: Map(S, [\,]) \\ & \underset{R}{\to} S(S(0)) :: S(0) :: [\,] \end{aligned}$$

We often omit the subscript $_R$ whenever no confusion arises. The set of defined symbols in a $\mathrm{TRS}_{hv}$ $R$ is denoted by $DF(R) = \{root(l) \mid l \to r \in R\}$. A $\mathrm{TRS}_{hv}$ $R$ is terminating if there is no infinite reduction sequence such that $t_0 \underset{R}{\to} t_1 \underset{R}{\to} t_2 \underset{R}{\to} \cdots$. Note that if we don't use suffix contexts in the definition of the reduction relation, $\mathrm{TRS}_{hv}$s are too restrictive to model of functional programming languages. For instance, the following $\mathrm{TRS}_{hv}$ is not confluent without suffix contexts.

$$\begin{cases} I(x) & \to & x \\ App(f,x) & \to & f(x) \end{cases}$$

In the system, we have

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

However, to reduce $I(f,x)$ to $f(x)$ we need to apply the first rule in the suffix context $\Box(x)$ which has the hole at a non-leaf position.

A binary relation $>$ is said to be a strict order if $>$ is transitive and irreflexive. A binary relation $\geq$ is said to be a partial order if $\geq$ is reflexive, transitive and antisymmetric. A binary relation $\gtrsim$ is said to be a quasi-order if $\gtrsim$ is transitive and reflexive. The strict part of a quasi-order $\gtrsim$, written by $\gnsim$, is defined as $\gtrsim \setminus \lesssim$. The partial extension of a strict-order $>$, written by $\geq$, is defined by its reflexive closure. Note that if $\gtrsim$ is a quasi-order then its strict part $\gnsim$ is a strict order, and if $>$ is a strict order then its partial extension $\geq$ is a partial order. A binary relation $\Upsilon$ on terms is said to be stable if $s \Upsilon t \Rightarrow s\theta \Upsilon t\theta$ for all substitutions $\theta$. A binary relation $\Upsilon$ on terms has the replacement property if $s \Upsilon t \Rightarrow a(\cdots s \cdots) \Upsilon a(\cdots t \cdots)$, has the supplement property if $a(\vec{s_i}) \Upsilon a'(\vec{t_j}) \Rightarrow a(\vec{s_i}, u) \Upsilon a'(\vec{t_j}, u)$. By an easy induction, we can prove that any transitive binary relation with the replacement property is closed contexts, and any transitive binary relation with the supplement property is closed suffix contexts. A strict order $>$ has the subterm property if $a(\cdots t \cdots) > t$ for all $t$ and $a \in \mathcal{F} \cup \mathcal{V}$. A strict order $>$ has the deletion property if $a(\ldots, t_i, \ldots) > a(\ldots, \ldots)$ for all $a(t_1, \ldots, t_n)$ and $i$. A well-founded strict order $>$ is said to be a reduction order if $>$ has the replacement, the supplement and the stability property.

**Theorem 2.1** Let $R$ be a $\mathrm{TRS}_{hv}$. Then $R$ is terminating iff there exists a reduction order $>$ satisfying $l > r$ for all $l \to r \in R$.

**Proof**. From the definition, it is trivial that $R$ is terminating iff the transitive closure of its reduction relation $\overset{+}{\underset{R}{\to}}$ is well-founded. Moreover, $\overset{+}{\underset{R}{\to}}$ is well-founded iff $\overset{+}{\underset{R}{\to}}$ is a reduction order, which obviously satisfies $l \overset{+}{\underset{R}{\to}} r$ for all $l \to r \in R$. On the other hand, letting $l > r$ for all $l \to r \in R$, if $s \underset{R}{\to} t$ then $s > t$ by the definition of reduction relation. Hence the well-foundedness of $>$ ensures that $R$ is terminating. $\Box$

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 strict order $>$, multisets $M$ and $N$, we define $M >^{mul} N$ as follows:

$$\forall n \in N - M.\ \exists m \in M - N.\ m > n$$

## 3. Simplification Orders

In TRSs, one of the most important concept for designing reduction orders is the notion of simplification orders by Dershowitz[6)].

**Definition 3.1** A simplification order is a strict order with the replacement, the deletion and the subterm property.

Any ground terms in our definition are also ground terms in the definition of 6), because no variable occurs in ground terms and the definition of terms in 6) does not fix the arity for each symbol, too. Hence the following theorem is derived from First Termination Theorem in 6).

**Theorem 3.2** Any simplification order is well-founded over ground terms.

## 4. Recursive Path Orders

Based on the notion of simplification orders, the recursive path order was introduced by Dershowitz[6)]. In this section, we extend the order over terms with higher-order variables.

**Definition 4.1** (Recursive Path Order) A precedence $\trianglerighteq$ is a quasi-order on $\mathcal{F}$, and $\sim$ is the equivalence part of $\trianglerighteq$. In this definition, we identify symbols $F$ and $G$ if $F \sim G$, that is, $a(t_1, \ldots, t_n) \equiv a'(t'_1, \ldots, t'_n)$ if $\forall i.\ t_i \equiv t'_i$ and either $a = a' \in \mathcal{V}$ or $a \sim a' \in \mathcal{F}$.
For terms $s \equiv a(s_1, \ldots, s_n)$ and $t \equiv a'(t_1, \ldots, t_m)$, we define $s >_{rpo} t$ as follows:

(1) $a \rhd a'$ and $s >_{rpo} t_j$ for all $j$,
(2) $a \sim a'$ and
$\{s_1, \ldots, s_n\} >_{rpo}^{mul} \{t_1, \ldots, t_m\}$,
(3) $s_i \geq_{rpo} t$ for some $i$,
(4) $a = a' \in \mathcal{V}$ and
$\{s_1, \ldots, s_n\} >_{rpo}^{mul} \{t_1, \ldots, t_m\}$, or
(5) $a' \in \mathcal{V}$, $a$ is the greatest w.r.t. $\trianglerighteq$ and
$\{s_1, \ldots, s_n\} \geq_{rpo}^{mul} \{a', t_1, \ldots, t_m\}$.

Rules (1), (2) and (3) correspond to the original recursive path order in TRSs. Since rules (4) and (5) are applied only the case when we treat variables, it is trivial that $>_{rpo}$ is a simplification order over ground terms. Moreover, the following theorem holds.

**Theorem 4.2** The recursive path order is a simplification order.

**Proof**. Let $\trianglerighteq$ be a precedence. It is obvious that $>_{rpo}$ has the irrefflexivity, the deletion and the subterm property. First we prove the replacement property. Let $s >_{rpo} t$. For any $F \in \mathcal{F}$, it is obvious that $F(\cdots s \cdots) >_{rpo} F(\cdots t \cdots)$ by rule (2). For any $f \in \mathcal{V}$, it is obvious that $f(\cdots s \cdots) >_{rpo} f(\cdots t \cdots)$ by rule (4). Finally, letting $t_1 >_{rpo} t_2 >_{rpo} t_3$, we prove the transitivity, that is $t_1 >_{rpo} t_3$, by induction on $|t_1|+|t_2|+|t_3|$. Let $t_1 >_{rpo} t_2$ by rule $(\alpha)$ and $t_2 >_{rpo} t_3$ by rule $(\beta)$. It suffices to show the cases $(\alpha, \beta) = (2,5), (5,4)$. We suppose that $t_i \equiv a_i(t_{ij}, \ldots, t_{ik_i})$ for $i = 1, 2, 3$.

$(2,5)$: Since $a_2$ is the greatest, so is $a_1$. From the assumption, $\{\vec{t_{1j}}\} >_{rpo}^{mul} \{\vec{t_{2j}}\} \geq_{rpo}^{mul} \{a_3, \vec{t_{3j}}\}$. Hence $\{\vec{t_{1j}}\} >_{rpo}^{mul} \{a_3, \vec{t_{3j}}\}$. It follows that $t_1 >_{rpo} t_3$ by rule (5).

$(5,4)$: From the assumption, $\{\vec{t_{2j}}\} \geq_{rpo}^{mul} \{\vec{t_{3j}}\}$. Hence $\{a_3, \vec{t_{2j}}\} \geq_{rpo}^{mul} \{a_3, \vec{t_{3j}}\}$. From $a_2 = a_3$, $\{\vec{t_{1j}}\} \geq_{rpo}^{mul} \{a_3, \vec{t_{3j}}\}$. It follows that $t_1 >_{rpo} t_3$ by rule (5). □

Next we prove the stability of recursive path order.

**Theorem 4.3** The recursive path order is stable.

**Proof**. Let $s \equiv a(s_1, \ldots, s_n) >_{rpo} a'(t_1, \ldots, t_m) \equiv t$ and $\theta$ be a substitution. By induction on $|s| + |t|$, we prove $s\theta >_{rpo} t\theta$. According to the definition of $s >_{rpo} t$, we have the following five cases:

• Rule (1): From the induction hypothesis, $s\theta >_{rpo} t_j\theta$ for all $j$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (1).

• Rule (2): From the induction hypothesis, it is obvious that $\{s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{t_1\theta, \ldots, t_m\theta\}$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (2).

• Rule (3): From the induction hypothesis, $s_i\theta \geq_{rpo} t\theta$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (3).

• Rule (4): Let $\theta(a) \equiv a''(\vec{u_i})$. From the induction hypothesis, it is obvious that $\{s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{t_1\theta, \ldots, t_m\theta\}$. Hence $\{\vec{u_i}, s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{\vec{u_i}, t_1\theta, \ldots, t_m\theta\}$. Therefore it follows that $s\theta >_{rpo} t\theta$ by rule (2) or (4).

•Rule (5): Let $\theta(a') \equiv a''(u_1, \ldots, u_k)$. We have the following four cases.

– $a \rhd a''$: From the induction hypothesis, it is obvious that $\{s_1\theta, \ldots, s_n\theta\} \geq_{rpo}^{mul} \{a''(u_1, \ldots, u_k), t_1\theta, \ldots, t_m\theta\}$. From the subterm property, for all $j$ there exists a $s_i$ such that $s\theta >_{rpo} s_i\theta \geq_{rpo} a''(u_1, \ldots, u_k) >_{rpo} u_j$, and for all $j$ there exists a $s_i$ such that $s\theta >_{rpo} s_i\theta \geq_{rpo} t_j\theta$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (1).

– $a \sim a''$ and $\exists p.s_p \equiv a'$: From the assumption, $\{s_1, \ldots, s_n\} - \{s_p\} \geq_{rpo}^{mul} \{t_1, \ldots, t_m\}$. From the induction hypothesis, it is obvious that $\{s_1\theta, \ldots, s_n\theta\} - \{s_p\theta\} \geq_{rpo}^{mul} \{t_1\theta, \ldots, t_m\theta\}$. Since $s_p\theta >_{rpo} u_i$ for all $i$ by the subterm property, $\{s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{u_1, \ldots, u_k, t_1\theta, \ldots, t_m\theta\}$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (2).

– $a \sim a''$ and $\forall i.s_i \not\equiv a'$: From the assumption, there exists $s_p \in \{s_1, \ldots, s_n\} - \{t_1, \ldots, t_m\}$ such that $s_p >_{rpo} a'$. Without loss of generality, we assume that there is a number $q$ such that $s_p >_{rpo} t_i$ for all $i \leq q$ and $\{s_1, \ldots, s_n\} - \{s_p\} \geq_{rpo}^{mul} \{t_{q+1}, \ldots, t_m\}$. From the induction hypothesis, it is obvious that $\{s_1\theta, \ldots, s_n\theta\} - \{s_p\theta\} \geq_{rpo}^{mul} \{t_{q+1}\theta, \ldots, t_m\theta\}$. From the induction hypothesis and the subterm property, $s_p\theta >_{rpo} t_j\theta$ for all $j \leq q$, and $s_p\theta >_{rpo} a''(u_1, \ldots, u_k) >_{rpo} u_i$ for all $i$. Thus it follows that $\{s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{u_1, \ldots, u_k, t_1\theta, \ldots, t_m\theta\}$. Hence it follows that $s\theta >_{rpo} t\theta$ by rule (2).

– $a'' \in \mathcal{V}$: From the induction hypothesis, $\{s_1\theta, \ldots, s_n\theta\} \geq_{rpo}^{mul} \{a''(\vec{u_i}), t_1\theta, \ldots, t_m\theta\}$. Suppose that $k = 0$. Since $\{\vec{s_i\theta}\} \geq_{rpo}^{mul} \{a'', \vec{t_j\theta}\}$, $s\theta >_{rpo} t\theta$ by rule (5). Suppose that $k > 0$. It follows that $a''(\vec{u_i}) >_{rpo} u_i$ for any $i$ by the subterm property and $a''(\vec{u_i}) >_{rpo} a''$ by rule (4). Thus $\{a''(\vec{u_i}), t_1\theta, \ldots, t_m\theta\} >_{rpo}^{mul} \{a'', \vec{u_i}, t_1\theta, \ldots, t_m\theta\}$. Hence $\{s_1\theta, \ldots, s_n\theta\} >_{rpo}^{mul} \{a'', \vec{u_i}, t_1\theta, \ldots, t_m\theta\}$. Therefore it follows that $s\theta >_{rpo} t\theta$ by rule (5). $\square$

Unfortunately, the recursive path order does not have the supplement property. For example, letting $G \rhd F$, then $F(G(0)) >_{rpo} G(0)$, but $F(G(0), x) <_{rpo} G(0, x)$. In order to solve the difficulty, we need the dependency pair and the argument filtering method. Here we introduce the notion of semi-reduction orders, which is defined as reduction orders without supplement property.

**Theorem 4.4** The recursive path order is a semi-reduction order.

**Proof**. Thanks to Theorems 4.2 and 4.3, it suffices to show that the well-foundedness. Assume that there exists an infinite decreasing sequence $t_0 >_{rpo} t_1 >_{rpo} t_2 >_{rpo} \cdots$. From the definition of $>_{rpo}$, it is trivial that $Var(t_i) \supseteq Var(t_{i+1})$ for all $i$. We define the substitution $\theta$ by $\theta(x) = F$ for all $x \in Var(t_0)$, where $F$ is a function symbol. From Theorem 4.3, $t_0\theta >_{rpo} t_1\theta >_{rpo} t_2\theta >_{rpo} \cdots$. Because all $t_i\theta$ is a ground term, $>_{rpo}$ is not well-founded over ground terms. It is a contradiction to Theorem 3.2. $\square$

## 5. Dependency Pairs

The notion of dependency pairs was introduced for proving termination of first-order TRSs by Arts and Giesl[1),2)]. This notion was extended to Nipkow's system, which is a different framework than ours, by Watanabe, Kawaguti, Sakai, Sakabe and Inagaki[17)]. In this section, we define the dependency pair of $\mathrm{TRS}_{hv}$s .

**Definition 5.1** $\mathcal{F}^\# = \{F^\# \mid F \in \mathcal{F}\}$ is a set of marked symbols disjoint from $\mathcal{F} \cup \mathcal{V}$. We define the root-marked terms by $(F(t_1, \ldots, t_n))^\# = F^\#(t_1, \ldots, t_n)$. If $root(t) \in \mathcal{V}$ then we identify $t^\#$ with $t$.
Let $R$ be a TRS. A pair $\langle u^\#, v^\# \rangle$ of terms is an outer dependency pair of $R$ if there exists a rule $u \to v \in R$ such that $root(v) \in DF(R) \cup \mathcal{V}$. A pair $\langle u^\#, v^\# \rangle$ of terms is an inner dependency pair of $R$ if there exist a rule $u \to C[v] \in R$ with $C[\,] \not\equiv \square$ such that $root(v) \in DF(R) \cup \mathcal{V}$ and $v$ itself is not a variable. The sets $DP^\#_{out}(R)$ and $DP^\#_{in}(R)$ are defined by all outer and inner dependency pairs, respectively. The set $DP^\#(R)$ of dependency pairs of $R$ is defined by $DP^\#(R) = DP^\#_{out}(R) \cup DP^\#_{in}(R)$.

**Definition 5.2** A term $t$ is said to be almost terminating if any immediate subterm of $t$ is terminating but $t$ is not terminating. A sequence of dependency pairs $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \cdots$ is said to be a dependency chain of $R$ if for each $i$ there exist a substitution $\theta_i$ and a suffix context $S_i[\,]$ such that $S_i[u_i\theta_i]$ is almost terminating and satisfying the following conditions:

- if $\langle u_i^\#, v_i^\# \rangle$ is an outer dependency pair: then $S_i[v_i\theta_i]^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$,
- if $\langle u_i^\#, v_i^\# \rangle$ is an inner dependency pair: then $(v_i\theta_i)^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$.

For example, in the following $\mathrm{TRS}_{hv}$ $R$

$$\begin{cases} F & \to & G(A) \\ G(A, f) & \to & f(F) \end{cases}$$

we have two outer dependency pairs $\langle F^\#, G^\#(A) \rangle$ and $\langle G^\#(A, f), f(F) \rangle$, and one inner dependency pair $\langle G^\#(A, F), F^\# \rangle$. This system is not terminating, because of

$F(F) \to G(A, F) \to F(F) \to G(A, F) \to \cdots$.

The following infinite dependency chain simulates this infinite reduction sequence:

$\langle F^\#, G^\#(A) \rangle \langle G^\#(A, f), f(F) \rangle \langle F^\#, G^\#(A) \rangle \cdots$.

We notice that the suffix context $S[\,] \equiv \square(f)$ is necessary for the infinite dependency chain.

**Lemma 5.3**

(i) If $t$ is terminating then any subterm of $t$ is terminating.

(ii) If $t$ is almost terminating then $root(t)$ is a defined symbol.

**Proof**. It is trivial. $\square$

**Lemma 5.4** Let $t$ be an almost terminating term. Then there exists a dependency pair $\langle u^\#, v^\# \rangle$, a substitution $\theta$ and a suffix context $S[\,]$ such that $t^\# \xrightarrow{*} S[u\theta]^\#$, $S[u\theta]$ is almost terminating, and satisfying the following properties:

- $S[v\theta]$ is almost terminating if $\langle u^\#, v^\# \rangle$ is an outer dependency pair.
- $v\theta$ is almost terminating if $\langle u^\#, v^\# \rangle$ is an inner dependency pair.

**Proof**. Since all immediate subterm of $t$ is terminating, any infinite reduction sequence from $t$ include some root position reductions. Hence there exist a rule $l \to r$, a substitution $\theta$ and a suffix context $S[\,]$ such that $t^\# \xrightarrow{*} S[l\theta]^\#$ and $S[r\theta]$ is not terminating. It is obvious that $S[l\theta]$ is almost terminating. We have three following cases:

**(a)** $r\theta$ is terminating:
Then $S[r\theta]$ is almost terminating. From Lemma 5.3, $root(S[r\theta])$ is a defined symbol. Hence $root(r\theta)$ is a defined symbol. It follows that $root(r)$ is a defined symbol or a variable. In both cases, it is obvious that $\langle l^\#, r^\# \rangle$ is an outer dependency pair.

**(b)** $r\theta$ is almost terminating:
From Lemma 5.3, $root(r\theta)$ is a defined symbol. It follows that $root(r)$ is a defined symbol or a variable. In both cases, it is obvious that $\langle l^\#, r^\# \rangle$ is an outer dependency pair. Moreover it is obvious that $S[r\theta]$ is almost terminating.

**(c)** Otherwise: Let $t'$ be a minimal size term in $\{r\theta|_p \mid r\theta|_p$ is not terminating $\}$. From the minimality, $t'$ is almost terminating. For any $x \in Var(r) \subseteq Var(l)$, $x\theta$ is a subterm of some immediate subterm of $l\theta$. Since $S[l\theta]$ is almost terminating, any immediate subterm of $l\theta$ is terminating. From Lemma 5.3 (i), $x\theta$ is terminating for any $x \in Var(r) \subseteq Var(l)$. Hence $t' \equiv r\theta|_p \equiv r|_p\theta$. Since $t'$ is almost terminating, $root(r|_p\theta)$ is a defined symbol. Hence $root(r|_p)$ is a defined symbol or a variable. In the former case, $\langle l^\#, (r|_p)^\# \rangle$ is an inner dependency pair. In the latter case, if $p$ is a leaf position in $r$ then $r|_p\theta$ is terminating, because $r|_p\theta$ is a subterm of $(l\theta)^\#$. It is a contradiction. If $p$ is not a leaf position in $r$ then $\langle l^\#, r|_p \rangle$ is an inner dependency pair. $\square$

**Theorem 5.5** $R$ is not terminating iff there exists an infinite dependency chain of $R$.

**Proof**. ($\Leftarrow$) Let $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \langle u_2^\#, v_2^\# \rangle \cdots$ be an infinite dependency chain with substitutions $\theta_i$ and suffix contexts $S_i[\,]$ such that $S_i[v_i\theta_i]^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP^\#_{out}(R)$, and $(v_i\theta_i)^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP^\#_{in}(R)$. From the definition of dependency pairs, there exist $C'_i[\,]$ such that $u_i \to C'_i[v_i] \in R$. We define contexts $C_i[\,]$ as $C_i[\,] \equiv \square$ for $\langle u_i^\#, v_i^\# \rangle \in DP^\#_{out}(R)$, and $C_i[\,] \equiv S_i[C'_i[\,]]$ for $\langle u_i^\#, v_i^\# \rangle \in DP^\#_{in}(R)$. Then $S_i[u_i\theta_i] \xrightarrow{+} C_i[S_{i+1}[u_{i+1}\theta_{i+1}]]$ for all $i$. Hence there exists an infinite reduction sequence $S_0[u_0\theta_0] \xrightarrow{+} C_0[S_1[u_1\theta_1]] \xrightarrow{+} C_0[C_1[S_2[u_2\theta_2]]] \xrightarrow{+} \cdots$.
($\Rightarrow$) Let $t_0$ be a minimal size counterexample, i.e., the size of $t_0$ is minimal in all nonterminating terms. Since $t_0$ is almost terminating, from Lemma 5.4, there exist a dependency pair $\langle u_0^\#, v_0^\# \rangle$, a substitution $\theta_0$ and a suffix context $S_0[\,]$ such that $t_0^\# \xrightarrow{*} S_0[u_0\theta_0]^\#$, $S_0[u_0\theta_0]$ is almost terminating, and either $v_0\theta_0$ or $S_0[v_0\theta_0]$ is almost terminating.

Let $t_1$ be either $v_0\theta_0$ or $S_0[v_0\theta_0]$, which is almost terminating. Applying Lemma 5.4 to $t_1$, we obtain a dependency pair $\langle u_1^\#, v_1^\# \rangle$, a substitution $\theta_1$ and a suffix context $S_1[\,]$ such that $t_1^\# \xrightarrow{*} S_1[u_1\theta_1]^\#$, $S_1[u_1\theta_1]$ is almost terminating, and either $v_1\theta_1$ or $S_1[v_1\theta_1]$ is almost terminating. Then it is easily seen that $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle$ is a dependency chain. Let $t_2$ be either $v_1\theta_1$ or $S_1[v_1\theta_1]$, which is almost terminating.

Repeating this procedure to $t_0, t_1, t_2, \ldots$, we obtain an infinite dependency chain $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \langle u_2^\#, v_2^\# \rangle \cdots$. $\square$

**Definition 5.6** A pair $(\gtrsim, >)$ of binary relations on terms is said to be a reduction pair for $R$ if it satisfies the following conditions:

- $\gtrsim$ has the replacement property, and the stability for $R$, that is, $l \gtrsim r \Rightarrow l\theta \gtrsim r\theta$ for all $l \to r \in R$.
- $>$ is well-founded, and stable for $DP^\#(R)$, that is, $u^\# > v^\# \Rightarrow u^\#\theta \gtrsim v^\#\theta$ for all $\langle u^\#, v^\# \rangle \in DP^\#(R)$.
- $\gtrsim \cdot > \;\subseteq\; >$ or $> \cdot \gtrsim \subseteq\; >$.
- $\gtrsim$ satisfies the marked condition, that is, $t \gtrsim t^\#$ for all $t$.
- $\gtrsim$ and $>$ have the supplement property for $R \cup DP^\#_{out}(R)$, that is, for any $(s,t) \in R \cup DP^\#_{out}(R)$, $S[\,]$ and $\theta$, $s\theta \gtrsim t\theta \Rightarrow S[s\theta] \gtrsim S[t\theta]$ and $s\theta > t\theta \Rightarrow S[s\theta] > S[t\theta]$.
- $S[t] \gtrsim t$ for any suffix context $S[\,]$.

Specially, a quasi-order $\gtrsim$ is said to be a weak reduction order if $(\gtrsim, \not\lesssim)$ is a reduction pair.

**Theorem 5.7** Let $R$ be a $\mathrm{TRS}_{hv}$ and $(\gtrsim, >)$ a reduction pair. If $l \gtrsim r$ for all $l \to r \in R$ and $u^\# > v^\#$ for all $\langle u^\#, v^\# \rangle \in DP^\#(R)$ then $R$ is terminating.

**Proof**. Without loss of generality, we suppose that $\gtrsim$ is a quasi-order and $>$ is a strict order, because it is obvious that $(\gtrsim^*, >^+)$ is also a reduction pair such that it satisfies assumptions, $\gtrsim^*$ is a quasi-order, and $>^+$ is a strict order.

We assume that $R$ is not terminating. From Theorem 5.5, there exists an infinite dependency chain $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \langle u_2^\#, v_2^\# \rangle \cdots$ with substitutions $\theta_i$ and suffix contexts $S_i[\,]$ such that $S_i[v_i\theta_i]^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP_{out}^\#(R)$, and $(v_i\theta_i)^\# \xrightarrow{*} S_{i+1}[u_{i+1}\theta_{i+1}]^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP_{in}^\#(R)$.

From the assumption and the stability of $>$, we have $u_i^\#\theta_i > v_i^\#\theta_i$ for all $i$. Since $root(u_i)$ is a function symbol, $u_i^\#\theta_i \equiv (u_i\theta_i)^\#$. From the marked condition, $v_i^\#\theta_i \gtrsim (v_i\theta_i)^\#$. Hence $(u_i\theta_i)^\# > \cdot \gtrsim (v_i\theta_i)^\#$. From the supplement property, $S_i[(u_i\theta_i)^\#] > \cdot \gtrsim S_i[(v_i\theta_i)^\#]$. From the assumption, $S_i[v_i\theta_i]^\# \gtrsim (v_i\theta_i)^\#$. It follows that $S_i[u_i\theta_i]^\# > \cdot \gtrsim S_i[v_i\theta_i]^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP_{out}^\#(R)$, and $S_i[u_i\theta_i]^\# > \cdot \gtrsim (v_i\theta_i)^\#$ for $\langle u_i^\#, v_i^\# \rangle \in DP_{in}^\#(R)$.

Since $\gtrsim$ has the replacement, the supplement, and the stable property, it follows that $\xrightarrow{*} \subseteq \gtrsim$.

Therefore, we obtain an infinite decreasing sequence $S_0[u_0\theta_0]^\# > \cdot \gtrsim S_1[u_1\theta_1]^\# > \cdot \gtrsim S_2[u_2\theta_2]^\# > \cdot \gtrsim \cdots$. It is a contradiction to the well-foundedness of $>$. $\square$

Note that the theorem correspond to Theorem 5.5 for HRSs essentially requires the subterm relation, i.e., the definition of dependency chains in HRSs uses the condition $v_i\theta_i \xrightarrow[in]{*} \trianglerighteq_{sub} u_{i+1}\theta_{i+1}$ instead of $(v_i\theta_i)^\# \xrightarrow{*} (u_{i+1}\theta_{i+1})^\#$ (See 17)). Hence weak reduction orders in HRSs must have the subterm property. The fact is very restrictive, because the argument filtering method, introduced in the next section, cannot work well in HRSs. On the other hand, in TRS$_{hv}$s the argument filtering method is still effective.

## 6. Argument Filtering Method

The argument filtering method, which designs weak reduction orders from arbitrary reduction orders, was first proposed by Arts and Giesl in TRSs[3]. In this section, we extend the method to TRS$_{hv}$s .

**Definition 6.1** 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 $i_1 < \cdots < i_k$. $\pi(F)^{\leq n}$ 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: letting $t \equiv a(t_1, \ldots, t_n)$

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

For a given argument filtering function $\pi$ and binary relation $>$, we define $s \gtrsim_\pi t$ by $\pi(s) \geq \pi(t)$.

**Lemma 6.2** If $>$ is a semi-reduction order then $\gtrsim_\pi$ has the replacement property.

**Proof**. Let $C[\,] \equiv a(\ldots, \square, \ldots)$. If $\pi(C[\,])$ does not include $\square$ then $\pi(C[s]) \equiv \pi(C[\,]) \equiv \pi(C[t])$, otherwise $s \gtrsim_\pi t \Rightarrow \pi(s) \geq \pi(t) \Rightarrow \pi(C)[\pi(s)] \geq \pi(C)[\pi(t)] \Rightarrow \pi(C[s]) \geq \pi(C[t]) \Rightarrow C[s] \gtrsim_\pi C[t]$. $\square$

**Lemma 6.3** If $>$ is a semi-reduction order then $\gtrsim\!\!\!\!\!/_{\pi}$ is well-founded.

**Proof**. If $t_0 \gtrsim\!\!\!\!\!/_{\pi}\ t_1 \gtrsim\!\!\!\!\!/_{\pi}\ t_2 \gtrsim\!\!\!\!\!/_{\pi} \cdots$, then $\pi(t_0) > \pi(t_1) > \pi(t_2) > \cdots$. Therefore, the well-foundedness of $>$ ensures that of $\gtrsim\!\!\!\!\!/_{\pi}$. $\square$

**Lemma 6.4** If a strict order $>$ has the deletion property, then $S[t] \gtrsim_\pi t$.

**Proof**. From the definition of the argument filtering, if $\pi(S[t]) \equiv a(t_1, \ldots, t_n)$, then $\pi(t) \equiv a(t_1, \ldots, t_m)$ for some $m \leq n$. From the deletion property, $a(t_1, \ldots, t_n) \geq a(t_1, \ldots, t_m)$. Hence $S[t] \gtrsim_\pi t$. $\square$

Unfortunately, $\gtrsim_\pi$ does not have the stability property. For example, let $\trianglerighteq$ be the precedence with $2 \triangleright 1 \triangleright 0$, $\theta(f) = F$ and $\pi(F) = [2]$, then $\pi(f(2,0)) \equiv f(2,0)$, $\pi(f(1,1)) \equiv f(1,1)$, $\pi(f(2,0)\theta) \equiv F(0)$, and $\pi(f(1,1)\theta) \equiv F(1)$. Thus we obtain the following counterexample:

$$f(2,0) >_{rpo} f(1,1), \text{ but } F(0) <_{rpo} F(1).$$

Hence we need a suitable restriction.

**Lemma 6.5** Let $>$ be a strict order with the replacement and the deletion property, and $\theta_\pi$ be the substitution defined as $\theta_\pi(x) = \pi(x\theta)$ for all $x \in \mathcal{V}$. Then $\pi(t)\theta_\pi \geq \pi(t\theta)$. Moreover, if no variable occurs at non-leaf positions in $t$ then $\pi(t)\theta_\pi \equiv \pi(t\theta)$.

**Proof**. Let $t \equiv a(t_1, \ldots, t_n)$. We prove $\pi(t)\theta_\pi \geq \pi(t\theta)$ by induction on $|t|$. In the case $a \in \mathcal{F}$ with $\pi(a)^{\leq n} = [i_1, \ldots, i_m]$, the claim holds as follows:

$\pi(a(t_1, t_2, \ldots, t_n))\theta_\pi$

$$
\begin{aligned}
&\equiv a(\pi(t_{i_1}),\pi(t_{i_2}),\ldots,\pi(t_{i_m}))\theta_\pi\\
&\equiv a(\pi(t_{i_1})\theta_\pi,\pi(t_{i_2})\theta_\pi,\ldots,\pi(t_{i_m})\theta_\pi)\\
&\geq a(\pi(t_{i_1}\theta),\pi(t_{i_2})\theta_\pi,\ldots,\pi(t_{i_m})\theta_\pi)\\
&\vdots\\
&\geq a(\pi(t_{i_1}\theta),\pi(t_{i_2}\theta),\ldots,\pi(t_{i_m}\theta))\\
&\equiv \pi(a(t_1\theta,t_2\theta,\ldots,t_n\theta))\\
&\equiv \pi(a(t_1,t_2,\ldots,t_n)\theta)
\end{aligned}
$$

In the case $a \in \mathcal{V}$ and $root(\theta(a)) \in \mathcal{F}$, we suppose that $\theta(a) = a'(t'_1,\ldots,t'_k)$, $[i_1,\ldots,i_p] = \pi(a')^{\leq k+n}$ and $[i_1,\ldots,i_q] = \pi(a')^{\leq k}$, then the claim holds as follows:

$$
\begin{aligned}
&\pi(a(t_1,\ldots,t_n))\theta_\pi\\
&\quad\equiv\ a(\pi(t_1),\ldots,\pi(t_n))\theta_\pi\\
&\quad\equiv\ a'(\pi(t'_{i_1}),\ldots,\pi(t'_{i_q}),\pi(t_1)\theta_\pi,\ldots,\pi(t_n)\theta_\pi)\\
&\quad\geq\ a'(\pi(t'_{i_1}),\ldots,\pi(t'_{i_q}),\\
&\qquad\qquad \pi(t_{i_{q+1}-k})\theta_\pi,\ldots,\pi(t_{i_p-k})\theta_\pi)\\
&\quad\geq\ a'(\pi(t'_{i_1}),\ldots,\pi(t'_{i_q}),\\
&\qquad\qquad \pi(t_{i_{q+1}-k}\theta),\ldots,\pi(t_{i_p-k}\theta))\\
&\quad\equiv\ \pi(a'(t'_1,\ldots,t'_k,t_1\theta,\ldots,t_n\theta))\\
&\quad\equiv\ \pi(a(t_1,\ldots,t_n)\theta)
\end{aligned}
$$

In the case $a \in \mathcal{V}$ and $root(\theta(a)) \in \mathcal{V}$, we suppose that $\theta(a) = a'(t'_1,\ldots,t'_k)$, then the claim holds as follows:

$$
\begin{aligned}
&\pi(a(t_1,\ldots,t_n))\theta_\pi\\
&\quad\equiv\ a(\pi(t_1),\ldots,\pi(t_n))\theta_\pi\\
&\quad\equiv\ a'(\pi(t'_1),\ldots,\pi(t'_k),\pi(t_1)\theta_\pi,\ldots,\pi(t_n)\theta_\pi)\\
&\quad\geq\ a'(\pi(t'_1),\ldots,\pi(t'_k),\pi(t_1\theta),\ldots,\pi(t_n\theta))\\
&\quad\equiv\ \pi(a'(t'_1,\ldots,t'_k,t_1\theta,\ldots,t_n\theta))\\
&\quad\equiv\ \pi(a(t_1,\ldots,t_n)\theta)
\end{aligned}
$$

Moreover, if no variable occurs at non-leaf position in $t$ then $\pi(t)\theta_\pi \equiv \pi(t\theta)$ can be proved similar to the above proof. □

**Lemma 6.6** Suppose that $>$ be a semi-reduction order with the deletion property. Let $s$ and $t$ be terms such that no variable occurs at non-leaf positions in $s$. If $s \gtrsim_\pi t$ (resp. $s \gtrapprox_\pi t$) then $s\theta \gtrsim_\pi t\theta$ (resp. $s\theta \gtrapprox_\pi t\theta$).

**Proof**. From Lemma 6.5 and the stability of $>$. □

In order to satisfy the supplement property, we introduced the arity condition for pair $(\pi, R)$ of an argument filtering function $\pi$ and a $\mathrm{TRS}_{hv}$ $R$, which is defined as $a \in \mathcal{F}$, $\pi(a) = [i_1,\ldots,i_n]$ and $i_n \leq m$ for all $l \to a(r_1,\ldots,r_m) \in R$.

**Lemma 6.7** Let $R$ be a $\mathrm{TRS}_{hv}$, $\pi$ an argument filtering function, $\theta$ a substitution, $S[\,]$ a suffix context and $>$ a semi-reduction order with the deletion property. Supposing that $(\pi, R)$ has the arity condition. For all $l \to r \in R$, if $l\theta \gtrsim_\pi r\theta$ (resp. $l\theta \gtrapprox_\pi r\theta$) then $S[l\theta] \gtrsim_\pi S[r\theta]$ (resp. $S[l\theta] \gtrapprox_\pi S[r\theta]$).

**Proof**. From the arity condition, $\pi(S[r\theta]) \equiv \pi(r\theta)$. From Lemma 6.4, $S[l\theta] \gtrsim_\pi l\theta$. Hence if $l\theta \gtrsim_\pi r\theta$ then $S[l\theta] \gtrsim_\pi l\theta \gtrsim_\pi r\theta \sim_\pi S[r\theta]$, and if $l\theta \gtrapprox_\pi r\theta$ then $S[l\theta] \gtrsim_\pi l\theta \gtrapprox_\pi r\theta \sim_\pi S[r\theta]$. □

Now we present a method for proving termination of $\mathrm{TRS}_{hv}$s .

**Theorem 6.8** Let $R$ be a $\mathrm{TRS}_{hv}$, $\pi$ an argument filtering function and $>$ be a semi-reduction order with the deletion property. Suppose that
- $l \gtrsim_\pi r$ for all $l \to r \in R$,
- $u^\# \gtrapprox_\pi v^\#$ for all $\langle u^\#, v^\#\rangle \in DP^\#(R)$,
- no variable occurs at non-leaf position in $l$ for all $l \to r \in R$,
- $\gtrsim_\pi$ satisfies the marked condition, and
- $(\pi, R)$ satisfies the arity condition,

then $R$ is terminating.

**Proof**. From Lemma 6.2, $\gtrsim$ has the replacement property. From Lemma 6.3, $\gtrapprox$ is well-founded. From Lemma 6.6 and the assumption, $\gtrsim$ and $\gtrapprox$ are stable for $R \cup DP^\#(R)$. From the arity condition and Lemma 6.7, $\gtrsim$ and $\gtrapprox$ have the supplement property. From Lemma 6.4, $S[t] \gtrsim t$ for any suffix context $S[\,]$. Therefore $R$ is terminating by Theorem 5.7. □

In general, $\gtrsim_\pi$ does not satisfy the marked condition. Hence we need a suitable restriction.

**Theorem 6.9** Let $R$ be a $\mathrm{TRS}_{hv}$, $\trianglerighteq$ be a precedence and $\pi$ an argument filtering function. Suppose that
- $\pi(l) \geq_{rpo} \pi(r)$ for all $l \to r \in R$,
- $\pi(u^\#) >_{rpo} \pi(v^\#)$ for all $\langle u^\#, v^\#\rangle \in DP^\#(R)$,
- no variable occurs at non-leaf position in $l$ for all $l \to r \in R$,
- $F \trianglerighteq F^\#$ and $\pi(F) \supseteq \pi(F^\#)$ for all $F \in DF(R)$, and
- $(\pi, R)$ satisfies the arity condition,

then $R$ is terminating.

**Proof**. We define $\gtrsim$ by the argument filtering based on $>_{rpo}$, that is, $s \gtrsim t \iff \pi(s) \geq_{rpo} \pi(t)$. Since $F \trianglerighteq F^\#$ and $\pi(F) \supseteq \pi(F^\#)$, $\gtrsim$ satisfies the marked condition. Therefore $R$ is terminating by Theorem 6.8. □

**Example 6.10** Here we prove the termination of $R_{Map}$ defined as follows:

$$\begin{cases} Map(f,[]) & \to & [] \\ Map(f,x::xs) & \to & f(x)::Map(f,xs) \end{cases}$$

Let $Map \sim Map^{\#} \rhd :: \rhd []$, $\pi(Map) = \pi(Map^{\#}) = \pi(::) = [1,2]$ and $\pi([]) = []$. Then for all rules

$$\pi(Map(f,[])) \quad \geq_{rpo} \quad \pi([])$$
$$\pi(Map(f,x::xs)) \geq_{rpo} \pi(f(x)::Map(f,xs)),$$

and for all dependency pairs

$$\pi(Map^{\#}(f,x::xs)) \quad \geq_{rpo} \quad \pi(Map^{\#}(f,xs))$$
$$\pi(Map^{\#}(f,x::xs)) \quad \geq_{rpo} \quad \pi(f(x)).$$

Therefore $R_{Map}$ is terminating by Theorem 6.9.

We also show the termination of the following non-simply terminating $\mathrm{TRS}_{hv}$ $R$.

**Example 6.11** Let $R$ be the $\mathrm{TRS}_{hv}$ defined as follows:

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

Then there exist three dependency pairs

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

Suppose that $F \sim F^{\#} \rhd G$, $\pi(F^{\#}) = \pi(F) = [1,2]$ and $\pi(G) = []$. Then for the rule

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

and for all dependency pairs

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

Therefore $R$ is terminating by Theorem 6.9.

When we try to prove the termination of non-simply terminating $\mathrm{TRS}_{hv}$s , the argument filtering method is very effective. On the other hand, in the framework of HRSs introduced by Nipkow, the argument filtering method cannot be designed essentially. This fact proves the usefulness of our $\mathrm{TRS}_{hv}$s .

## 7. Simply-Typed TRSs

We have discussed with methods for proving termination of $\mathrm{TRS}_{hv}$s . A lot of unwilling restriction are caused by the supplement property. In this section, we solve such restrictions by introducing simply typed systems.

**Definition 7.1** A set of basic-types is denoted by $\mathcal{B}$. The set $\mathcal{T}$ of simply-types is generated from $\mathcal{B}$ by the constructor $\to$ as follows:

$$\mathcal{T} ::= \mathcal{B} \mid (\mathcal{T} \to \mathcal{T})$$

A type attachment $\tau$ is a function from $\mathcal{F} \cup \mathcal{V}$ to $\mathcal{T}$. A term $a(t_1,\ldots,t_n)$ has type $\beta$ if $\tau(a) = (\alpha_1 \to (\cdots \to (\alpha_n \to \beta)\cdots))$ and each $t_i$ has type $\alpha_i$. A term is said to be a simply-typed term if it has a simply-typed type.

**Definition 7.2** A $\mathrm{TRS}_{hv}$ is said to be a simply-typed TRS if $l$ and $r$ have the same basic type for all $l \to r \in R$. A reduction relation $\underset{R}{\to}$ in simply-typed TRS is defined over simply-typed terms.

The restriction by simply-typed system is very useful. We can eliminate suffix contexts in the definition of the reduction relation, because we can naturally restrict that each rule has a basic type. Hence semi-reduction orders can directly prove termination.

**Theorem 7.3** Let $R$ be a simply-typed TRS. Then $R$ is terminating iff there exists a semi-reduction order $>$ satisfying $l > r$ for all $l \to r \in R$.

**Proof**. Let $s \underset{R}{\to} t$. Then there exist a context $C[\,]$, a substitution $\theta$ and a rule $l \to r \in R$ such that $s \equiv C[S[l\theta]]$ and $t \equiv C[S[r\theta]]$. Thanks to the restriction by simply-typed, $S[\,] \equiv \Box$. It follows that if $l > r$ for all $l \to r \in R$ then $\underset{R}{\to} \subseteq >$. Hence the well-foundedness of $>$ ensures that $R$ is terminating. On the other hand, $R$ is terminating iff $\overset{+}{\underset{R}{\to}}$ is a semi-reduction order, which obviously satisfies $l \overset{+}{\underset{R}{\to}} r$ for all $l \to r \in R$. □

In simply-typed TRSs, for any simply-typed terms $s$ and $t$, if $s \to t$ then both terms have the same types. For any simply-typed term $s$ and $\theta$, if $s\theta$ has a simply-typed then $s$ and $s\theta$ has the same type. Moreover we suppose that the left-hand side of any rules has a basic type. Hence we can define dependency pairs in simply-typed TRSs as $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}(R)$ such that $v^{\#}$ has a basic type and itself is not a variable. We denote by $DP^{\#}_s(R)$ the set of dependency pairs of simply-typed TRS $R$. We can also define dependency chains in simply-typed systems as $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \cdots$ such that there exist $\theta_i$ and $(v_i\theta_i)^\# \xrightarrow{*} (u_{i+1}\theta_{i+1})^\#$. From Theorem 5.5, we obtain the following corollary:

**Corollary 7.4** A simply-typed TRS $R$ is not terminating iff there exists an infinite dependency chains in simply-typed systems.

Next we should define reduction pairs in simply-typed systems.

**Definition 7.5** A pair $(\gtrsim, >)$ of binary relations on terms is said to be a reduction pair for $R$ in simply-typed systems if it satisfies the following conditions:

- $\gtrsim$ has the replacement property and the stability for $R$.
- $>$ is well-founded and stable for $DP^\#(R)$,
- $\gtrsim \cdot > \;\subseteq\; >$ or $> \cdot \gtrsim \subseteq\; >$.
- $\gtrsim$ satisfies the marked condition,

Specially, a quasi-order $\gtrsim$ is said to be a weak reduction order in simply-typed systems if $(\gtrsim, \gnsim)$ is a reduction pair in simply-typed systems.

The above definition is only to remove last two conditions for the supplement property in Definition 5.6. Hence we obtain the following corollary from Theorem 5.7.

**Corollary 7.6** Let $R$ be a simply-typed TRS and $(\gtrsim, >)$ a reduction pair for $R$ in simply typed systems. If $l \gtrsim r$ for all $l \to r \in R$ and $u^\# > v^\#$ for all $\langle u^\#, v^\# \rangle \in DP_s^\#(R)$ then $R$ is terminating.

Finally, since reduction pairs in simply typed systems does not require the supplement property, from Theorems 6.8 and 6.9, we obtain two following corollaries.

**Corollary 7.7** Let $R$ be a simply typed TRS, $\pi$ an argument filtering function and $>$ be a semi-reduction order. Suppose that

- $l \gtrsim_\pi r$ for all $l \to r \in R$,
- $u^\# \gnsim_\pi v^\#$ for all $\langle u^\#, v^\# \rangle \in DP^\#(R)$,
- no variable occurs at non-leaf position in $l$ for all $l \to r \in R$, and
- $\gtrsim_\pi$ satisfies the marked condition,

then $R$ is terminating.

**Corollary 7.8** Let $R$ be a simply-typed TRS, $\trianglerighteq$ be a precedence and $\pi$ an argument filtering function. Suppose that

- $\pi(l) \geq_{rpo} \pi(r)$ for all $l \to r \in R$,
- $\pi(u^\#) >_{rpo} \pi(v^\#)$ for all $\langle u^\#, v^\# \rangle \in DP^\#(R)$,
- no variable occurs at non-leaf position in $l$ for all $l \to r \in R$, and
- $F \trianglerighteq F^\#$ and $\pi(F) \supseteq \pi(F^\#)$ for all $F \in DF(R)$,

then $R$ is terminating.

At the end, in order to show the usefulness of our methods, we show the termination of the following non-simply terminating simply-typed TRS $R$ by Corollary 7.8.

**Example 7.9** Let $R$ be a simply-typed TRS defined as follows:

$$\begin{aligned}
If(T,x,y) &\to x\\
If(F,x,y) &\to y\\
Sub(x,0) &\to x\\
Sub(S(x),S(y)) &\to Sub(x,y)\\
Gtr(S(x),0) &\to T\\
Gtr(0,y) &\to F\\
Gtr(S(x),S(y)) &\to Gtr(x,y)\\
D(x,0) &\to T\\
D(S(x),S(y)) &\\
&\to If(Gtr(x,y),F,D(S(x),Sub(y,x)))\\
Len([\,]) &\to 0\\
Len(x :: xs) &\to S(Len(xs))\\
Flt(p,[\,]) &\to [\,]\\
Flt(p,x :: xs) &\\
&\to If(p(x), x :: Flt(p,xs), Flt(p,xs))
\end{aligned}$$

We suppose that $\mathcal{B} = \{Nat, NatList, Bool\}$, $Flt$ has the type $((Nat \to Bool) \to (NatList \to NatList))$, and other symbols have usual types. In this simply-typed TRS, $Sub$ defines usual subtraction on natural numbers, $Gtr$ defines usual order $>$ on natural numbers, $Len(Xs)$ calculates the length of list $Xs$, and $D(n,m)$ defines the predicate such that $n$ is a divisor of $m$. Moreover, the $Flt$-function ($Filter$-function) is one of the most standard higher-order function in functional programs. For example, the output of $Len(Flt(D(X),Xs))$ by $R$ is the number of multiples of $X$ in list $Xs$. Here, the dependency pairs $DP_s^\#(R)$ are defined as follows:

$$\begin{aligned}
&\langle Sub^\#(S(x),S(y)), Sub^\#(x,y)\rangle\\
&\langle Gtr^\#(S(x),S(y)), Gtr^\#(x,y)\rangle\\
&\langle D^\#(S(x),S(y)),\\
&\qquad If^\#(Gtr(x,y),F,D(S(x),Sub(y,x)))\rangle
\end{aligned}$$

$\langle D^\#(S(x), S(y)), Gtr^\#(x, y)\rangle$
$\langle D^\#(S(x), S(y)), D^\#(S(x), Sub(y, x))\rangle$
$\langle D^\#(S(x), S(y)), Sub^\#(y, x)\rangle$
$\langle Len^\#(x :: xs), Len^\#(xs)\rangle$
$\langle Flt^\#(p, x :: xs),$
$\quad If^\#(p(x), x :: Flt(p, xs), Flt(p, xs))\rangle$
$\langle Flt^\#(p, x :: xs), p(x)\rangle$
$\langle Flt^\#(p, x :: xs), Flt^\#(p, xs)\rangle$

We suppose that $G^\#$ is identified to $G$ for all $G \in \mathcal{F}$. We define the argument filtering function $\pi$ by $\pi(T) = \pi(F) = \pi(0) = \pi([\,]) = [],\ \pi(Sub) = \pi(S) = \pi(Len) = [1]$, $\pi(Gtr) = \pi(D) = \pi(::) = \pi(Flt) = [1, 2]$ and $\pi(If) = [1, 2, 3]$. We define the precedence $\trianglerighteq$ by $D \rhd Len \rhd S \rhd Sub \rhd 0$, $D \rhd If \rhd Gtr \rhd T \rhd F$, and $Flt \rhd G$ for all $G \in \mathcal{F}\backslash\{Flt\}$. Then it is routine to check that $\pi(l) \geq_{rpo} \pi(r)$ for all $l \to r \in R$ and $\pi(u^\#) >_{rpo} \pi(v^\#)$ for all $\langle u^\#, v^\#\rangle \in DP^\#(R)$. Therefore $R$ is terminating by Corollary 7.8.

Note that the above simply-typed TRS $R$ is not simply terminating, that is, any simplification orders cannot prove the termination. When we try to prove the termination of non-simply terminating simply-typed TRS, the argument filtering method is very effective. On the other hand, in the framework of HRSs introduced by Nipkow, the argument filtering method cannot be designed essentially. This fact proves the usefulness of our systems.

**Acknowledgments**

We would like to thank Masahiko Sakai for their useful comments and discussion, Yoshihito Toyama, Toshiyuki Yamada, and Munehiro Iwami for their useful discussion.

## References

1) Arts, T.: Automatically Proving Termination and Innermost Normalization of Term Rewriting Systems, PhD thesis, Univ. of Utrecht (1997).
2) Arts, T., Giesl, J.: Automatically Proving Termination Where Simplification Orderings Fail, LNCS 1214 (TAPSOFT'97), pp.261–272 (1997).
3) Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs, Theoretical Computer Science, Vol. 236, pp.133–178 (2000).
4) Baader, F., Nipkow,T.: Term Rewriting and All That, Cambridge University Press (1998).
5) Dershowitz, N.: A Note on Simplification Orderings, Information Processing Letters, 9(5), pp.212–215 (1979).
6) Dershowitz, N.: Orderings for Term-rewriting Systems, Theoretical Computer Science, Vol. 17, pp.279–301 (1982).
7) Giesl, J., Ohlebusch, E.: Pushing the Frontiers of Combining Rewrite Systems Farther Outwards, In Proc. of the 2nd Int. Workshop on Frontiers of Combining Systems (FroCoS '98), Amsterdam, The Netherlands, Studies in Logic and Computation 7, pp.141–160, Research Studies Press, John Wiley & Sons (2000).
8) Jouannaud, J.-P., Rubio, A.: Rewrite Orderings for Higher-Order Terms in $\eta$-Long $\beta$-Normal Form and the Recursive Path Ordering, Theoretical Computer Science 208(1–2), pp.33–58 (1998).
9) Jouannaud, J.-P., Rubio, A.: The Higher-Order Recursive Path Ordering, In IEEE Symposium on Logic in Computer Science, Trento, Italy, (1999).
10) Hudak, P., Jones, S.P., Walder, P.: Report on the Programming Language Haskell: A Non-Strict, Purely Functional Language, ACM SIGPLAN Notices 27(5) (1997).
11) Iwami, M., Toyama, Y.: Simplification Ordering for Higher-Order Rewrite Systems, IPSJ Trans. on Programming, vol.40, No.SIG 4 (PRO 3), pp.1-10 (1999).
12) Iwami, M.: Termination of Higher-Order Rewrite Systems, Ph.D. thesis, Japan Advanced Institute of Science and Technology (1999).
13) Milner, R., Tofte, M., Harper, R.: The Definition of Standard ML, MIT Press, 1990.
14) Nipkow, T.: Higher-Order Critical Pairs, In Proc. of 6th IEEE Symp. Logic in Computer Science, pp.342–349, IEEE Computer Society Press (1991).
15) Nipkow, T.: Orthogonal Higher-Order Rewrite Systems are Confluent, In Proc. of Int. Conf. on Typed Lambda Calculi and Applications, LNCS 664, pp.306–317 (1993).
16) Paulson, L.C.: ML for the Working Programmer, Cambridge University Press (1991).
17) Watanabe, Y., Kawaguti, N., Sakai, M., Sakabe, T., Inagaki, Y.: Proving Termination of Higher Order Rewriting Systems based on Dependency Pairs, Tech. Rep. of IEICE, SS97–85 (1998–03), pp.63–70 (1998), (in Japanese).