PAPER

An Extension of the Dependency Pair Method for Proving Termination of Higher-Order Rewrite Systems

Masahiko SAKAI†, *Regular Member*, Yoshitsugu WATANABE†*, *Nonmember*, *and* Toshiki SAKABE†, *Regular Member*

SUMMARY This paper explores how to extend the dependency pair technique for proving termination of higher-order rewrite systems. We show that the termination property of higher-order rewrite systems can be checked by the non-existence of an infinite R-chain, which is an extension of Arts' and Giesl's result for the first-order case. It is clarified that the subterm property of the quasi-ordering, used for proving termination automatically, is indispensable.

key words: termination, dependency pair, higher-order rewrite system

1. Introduction

Higher-order rewrite rules are used in functional programming, logic programming, and theorem proving. Automatic proving of the termination property is especially required for theorem provers. Several orderings for higher-order terms have been investigated by extending recursive path orderings for proving simple termination of term rewriting systems [7]–[10]. Among them, Jouannaud and Rubio's ordering has a simple definition by using type information. Iwami, Sakai and Toyama [6] extended the improved recursive decomposition ordering to the higher-order case by using Jouannaud's and Rubio's technique.

There is the dependency pair technique [1]–[3] for proving termination of term rewriting systems. It is useful, because it gives us a mechanical support for proving non-simple termination.

This paper extends the dependency pair technique to higher-order rewrite systems [11]. One of difficulties of the extension is the treatment of bound variables. For example, the following terminating higher-order rewrite system,

$$R\colon\ f(d) \to g(\lambda x.f(x)),$$

has an infinite R-chain $\langle f^{\#}(d), f^{\#}(x)\rangle \cdots$, if we do not care about bound variables. Unfortunately, transforming higher-order terms to typed algebraic terms [8] does not work well. For example, both sides of the rule in the following terminating higher-order rewrite system

Manuscript received June 26, 2000.
Manuscript revised February 8, 2001.

†The authors are with the Department of Information Engineering, Nagoya University, Nagoya-shi, 464–8603 Japan.
*Presently, with Building Systems Dept., Mitsubishi Electric Corporation Inazawa Works.

with $f : (\alpha \to \alpha \to \alpha) \to \beta$,

$$R\colon\ f(\lambda xy.x) \to f(\lambda xy.y),$$

are transformed to the typed algebraic term $f(\lambda_{\alpha\to\alpha\to\alpha}(\lambda_{\alpha\to\alpha}(c_\alpha)))$ by replacing lambda bindings by constants $\lambda_{\alpha\to\alpha\to\alpha}$ and $\lambda_{\alpha\to\alpha}$, and bound variables by constants c_α, where all function symbols are regarded as constants. Hence, it has an infinite R-chain.

We show that this problem can be solved by introducing a replacement of bound variables by constants into the definition of the subterm relation. By modifying and extending the definitions of dependency pairs and the R-chains, Arts' and Giesl's theorem is extended to higher-order case.

2. Preliminary Concepts

We assume the readers are familiar with the basic concepts and notations of term rewriting systems [5] and typed lambda calculi [4].

Given a set S of *basic types* (or *sorts*), the set τ_S of types is generated from S by the constructor $\to$ for *functional types*, that is, τ_S is the smallest set such that

$$\tau_S \supseteq S$$
$$\tau_S \supseteq \{\alpha \to \beta \mid \alpha, \beta \in \tau_S\}$$

Types that are not basic are called higher-order types. We use α, β to denote types. For a type β in form of $\alpha_1 \to \cdots \to \alpha_n \to \alpha$ where $n \geq 0$ and α is a basic type, the *output type* of β, denoted by $O(\beta)$, is α.

Let V_α be a set of *variables* of type α and $V = \bigcup_{\alpha\in\tau_S} V_\alpha$. Let C_α be a set of *constants* (or *function symbols*) of type α and $C = \bigcup_{\alpha\in\tau_S} C_\alpha$. We assume $V \cap C = \emptyset$, and $V_\alpha \cap V_\beta = \emptyset$ and $C_\alpha \cap C_\beta = \emptyset$ if $\alpha \neq \beta$. We use V_h to stand for the set of higher-order variables.

Constants are denoted by c, d, e, f and g. We use a to denote constants or variables.

The set T_α of *simply typed λ-terms of type α* is the smallest set satisfying the following:

$$T_\alpha \supseteq V_\alpha \cup C_\alpha$$
$$T_\alpha \supseteq \{(st) \mid s \in T_{\alpha'\to\alpha}, t \in T_{\alpha'}\}$$
$$T_\alpha \supseteq \{(\lambda x.s) \mid x \in V_{\beta'}, s \in T_\beta, \alpha = \beta' \to \beta\}$$

We write $t : \alpha$ to stand for $t \in T_\alpha$. Let $T = \bigcup_{\alpha\in\tau_S} T_\alpha$.

We call a simply typed λ-term t a *term*. We use l, p, q, r, s, t, u and v for terms. We use $FV(t)$ for the set of free variables of t and $BV(t)$ for the set of bound variables of t. Let $Var(t) = FV(t) \cup BV(t)$. We assume for convenience that bound variables in a term are all different, and are disjoint from free variables. We use F, G, H, X, and Y for free variables and x, y and z for bound variables. We use $\equiv$ to denote syntactic equality on terms.

A term containing special constants $\Box_{\alpha_1}, \ldots, \Box_{\alpha_n}$ is called a *context* denoted by $C_{\alpha_1,\ldots,\alpha_n}[\ ,\ldots,\ ]$. We use $C_{\alpha_1,\ldots,\alpha_n}[t_1,\ldots,t_n]$ for the term obtained from $C_{\alpha_1,\ldots,\alpha_n}[\ ,\ldots,\ ]$ by replacing $\Box_{\alpha_1},\ldots,\Box_{\alpha_n}$ with $t_1 : \alpha_1, \ldots, t_n : \alpha_n$ in left to right order. Types are sometimes omitted in case this causes no confusion.

Let $(\cdots(a\,t_1)\cdots t_n)$ be a term of a basic type. Then, it is denoted by $a(t_1,\ldots,t_n)$.

We will borrow from the λ-calculus the notions of α-equivalence, β-reduction and η-reduction. We do not distinguish terms that are α-equivalent. The term $t \equiv (\cdots((a\,t_1)t_2)\cdots t_n)$ is *η-expanded* to $\lambda x.(t\,x)$ if t is not of a basic type. We say t is *η-long β-normal form* (or *normalized*) if it is a normal form with respect to both β-reduction and η-expansion. We use $t{\downarrow}$ for the η-long β-normal form of t. The simply typed λ-calculus is confluent and terminating with respect to β-reduction, and with respect to β-reduction and η-expansion as well.

A substitution σ is a mapping $V \to T$ such that the type of $\sigma(X)$ is the same as the type of X. We define $Dom(\sigma) = \{X \mid X \not\equiv \sigma(X)\}$ and $Var(\sigma) = \bigcup_{X \in Dom(\sigma)} Var(\sigma(X))$. We sometimes use $[X_1 \mapsto t_1, \ldots, X_n \mapsto t_n]$ to denote a substitution σ such that $Dom(\sigma) = \{X_1,\ldots,X_n\}$ and $\sigma(X_i) \equiv t_i$ for all i. The restriction σ_Z of substitution σ for $Z \subseteq V$ is defined as follows:

$$\sigma_Z(X) \equiv \begin{cases} \sigma(X) & \text{if } X \in Z \\ X & \text{if } X \notin Z \end{cases}$$

Any substitution σ is extended to a mapping $\overline{\sigma} : T \to T$ as follows:

$$\begin{aligned} &\overline{\sigma}(X) \equiv \sigma(X) \\ &\overline{\sigma}(c) \equiv c \\ &\overline{\sigma}(s\,t) \equiv (\overline{\sigma}(s)\,\overline{\sigma}(t)) \\ &\overline{\sigma}(\lambda x.t) \equiv \lambda x.(\overline{\sigma_{Dom(\sigma)-\{x\}}}(t)) \text{ if } x \notin Var(\sigma) \end{aligned}$$

Note that α-conversion of t is possibly needed before applying $\overline{\sigma}$ to t in case of $Var(\sigma) \cap BV(t) \neq \emptyset$. Instead of $\overline{\sigma}(t)$, we write $t\overline{\sigma}$ or even $t\sigma$ by identifying σ and $\overline{\sigma}$.

It is known that normalized terms are of the form $\lambda x_1 \cdots x_m.\ a(t_1,\ldots,t_n)$ for some $m, n \geq 0$, $a \in C \cup V$ and terms $t_1,\ldots,t_n$ in η-long β-normal form themselves. Note that $a(t_1,\ldots,t_n)$ is of basic type. For an η-long β-normal form t of the form $a(u_1,\ldots,u_n)$, we write $top(t)$ for a.

A *higher-order rewrite system* (HRS) is a finite set of rewrite rules $R = \{l_i \to r_i : \alpha_i\}$, where l_i and r_i are normalized terms having the same basic type α_i. Given an HRS R, a normalized term s is *reduced* to a term t, written $s \to_R t$ or simply $s \to t$, if $s \equiv C[l\sigma{\downarrow}]$ and $t \equiv C[r\sigma{\downarrow}]$ for some context $C[\ ]$, substitution σ and rule $l \to r \in R$. If $C[\ ] \equiv \Box_\beta$ and $s : \beta$, it is written $s \xrightarrow{\Lambda} t$; otherwise it is written $s \xrightarrow{>\Lambda} t$. Note that t is also normalized if $s \to t$ [8].

We denote by $\xrightarrow{*}$ the reflexive transitive closure of the reduction relation $\to$. If there is an infinite reduction sequence $v \equiv v_0 \to v_1 \to \cdots$ from v, we say v has an infinite reduction sequence; otherwise we say v is terminating. If there exists no v that has an infinite reduction sequence, we say $\to$ is *terminating*. We also say that an HRS R is terminating if $\to_R$ is terminating.

The strict part $\succ$ of a quasi-ordering $\succeq$ is defined as $s \succ t \iff s \succeq t \wedge t \not\succeq s$. We also write $s \sim t$ for $s \succeq t \wedge t \succeq s$. An ordering $\succ$ on T is said to be *well-founded* if it does not admit an infinite sequence $t_1 \succ t_2 \succ \cdots$ of elements $t_1, t_2, \ldots \in T$. A quasi-ordering $\succeq$ is *closed under substitutions* if $s \succeq t \Rightarrow s\sigma{\downarrow} \succeq t\sigma{\downarrow}$ and $s \succ t \Rightarrow s\sigma{\downarrow} \succ t\sigma{\downarrow}$ for all substitutions σ. A quasi-ordering $\succeq$ is *weakly closed under contexts* if $s \succeq t \Rightarrow f(\ldots,s,\ldots) \succeq f(\ldots,t,\ldots)$ for all function symbols f. A quasi-ordering is called a *reduction quasi-ordering* if it is well-founded, closed under substitutions and weakly closed under contexts.

3. Dependency Pairs of HRSs

We extend the notion of dependency pairs [1]–[3] for proving termination of TRSs to higher-order rewrite systems.

It is a problem for developing dependency pair techniques for HRS that bound variables in a term t may become free in a subterm of t in the ordinary subterm definition. Hence, as a new definition we define that a subterm of t is a term obtained from an ordinary subterm of t by replacing each free variable x originated from a bound variable with a fresh constant c_x.

Definition 1: Let s be a normalized term. A term t is a *subterm* of s, denoted by $s \trianglerighteq t$, if

(a) $s \equiv t$, or
(b) $s \equiv \lambda x.s'$ and $s'[x \mapsto c_x] \trianglerighteq t$, or
(c) $s \equiv a(u_1,\ldots,u_n)$ and $u_i \trianglerighteq t$ for some $i \in \{1,\ldots,n\}$.

We say t is a *proper subterm* of s, denoted by $s \triangleright t$, if $s \trianglerighteq t$ and $s \not\equiv t$.

Example 2: The subterms of $f(\lambda x.F(x))$ are $f(\lambda x.F(x))$, $\lambda x.F(x)$, $F(c_x)$ and c_x.

Note that the subterm relation is not closed under substitutions. For example, $(F(d))\sigma{\downarrow} \equiv e \not\trianglerighteq d \equiv d\sigma{\downarrow}$ for $\sigma = [F \mapsto \lambda x.e]$.

Proposition 3:

(a) The subterm relation $\trianglerighteq$ is transitive.
(b) Let R be an HRS. Then, $(\trianglerighteq \circ \to_R) \subseteq (\to_R \circ \trianglerighteq)$, where $\circ$ denotes the composition of relations.

Proof

(a) Obvious.
(b) Letting $s \trianglerighteq t \to_R u$, we will show $s(\to_R \circ \trianglerighteq)u$ by induction on the definition of $\trianglerighteq$. We will only show the interesting case where $s \equiv \lambda x.s'$ and $s'[x \mapsto c_x] \trianglerighteq t$. We have $s'[x \mapsto c_x] \to_R t' \trianglerighteq u$ for some t' by induction hypothesis. Since there is no x occurrence in t', we have $t' \equiv t''[x \mapsto c_x]$ for some t'' having no c_x occurrence in it. Since we obtain $s' \to_R t''$ from $s'[x \mapsto c_x] \to_R t''[x \mapsto c_x]$, we have $\lambda x.s' \to_R \lambda x.t'' \trianglerighteq t''[x \mapsto c_x] \equiv t' \trianglerighteq u$. □

We say f is a defined symbol if $f = top(l)$ for some rule $l \to r$ and let $D = \{top(l) \mid l \to r \in R\}$ and $D^\# = \{f^\# \mid f \in D\}$ where $f^\#$ is a fresh symbol obtained by marking f in D. We define $s^\# \equiv f^\#(t_1, \ldots, t_n)$ if $s \equiv f(t_1, \ldots, t_n)$ and $f \in D$; otherwise $s^\# \equiv s$.

In case of TRSs, a dependency pair of a rule $l \to r$ is $\langle l^\#, t^\# \rangle$, where t is a subterm of r such that $top(t) \in D$. In case of HRSs, we also have to take a subterm whose top symbol is a higher-order free variable, because a defined symbol that causes infinite reduction may appear by a substitution, even if it substitutes terminating terms for variables. Unfortunately, we need the subterm relation in the definition of the R-chains and it is indispensable as shown by Example 9. Since all subterms are taken care of by the definition of the R-chains, any subterm located under a higher-order variable is unnecessary for a second component of dependency pairs.

Consider subterms s (of t) such that s is not located under a higher-order variable in t. The set of subterms of a normalized term t that are candidates for the second component in dependency pairs is formally defined as follows:

$$Cand(t) = \begin{cases} \{t\} \cup \bigcup_i Cand(t_i) & \text{if } t \equiv f(t_1, \ldots, t_n), f \in D \\ \bigcup_i Cand(t_i) & \text{if } t \equiv f(t_1, \ldots, t_n), f \in C - D \\ \{t\} & \text{if } t \equiv F(t_1, \ldots, t_n), F \in V_h \\ Cand(t'[x \mapsto c_x]) & \text{if } t \equiv \lambda x.t' \\ \emptyset & \text{otherwise.} \end{cases}$$

Example 4: Let $f \in D$ and $F \in V_h$. Then,

$$Cand(f(\lambda x.F(x))) = \{f(\lambda x.F(x)),\ F(c_x)\}.$$

Proposition 5:

(a) $s \in Cand(t)$ implies $t \trianglerighteq s$.
(b) $s \in Cand(t)$ implies $t\sigma\downarrow \trianglerighteq s\sigma\downarrow$ for any substitution σ such that $Var(\sigma) \cap BV(t) = \emptyset$.

Proof

(a) Obvious.
(b) We will prove it by induction on the definition of $Cand(t)$. From the assumption $s \in Cand(t)$, it is enough to consider the following three cases that $s \equiv t$, $t \equiv f(t_1, \ldots, t_n) \wedge s \in Cand(t_i)$ and $t \equiv \lambda x.t' \wedge s \in Cand(t'[x \mapsto c_x])$. In case of $s \equiv t$, it is trivial. In case of $t \equiv f(t_1, \ldots, t_n)$ and $s \in Cand(t_i)$ for some t_i, we have $t_i\sigma\downarrow \trianglerighteq s\sigma\downarrow$ by induction hypothesis. Since $t\sigma\downarrow \equiv f(t_1\sigma\downarrow, \ldots, t_n\sigma\downarrow)$, the claim holds. Consider the case $t \equiv \lambda x.t'$ and $s \in Cand(t'[x \mapsto c_x])$. We have $t\sigma\downarrow \equiv (\lambda x.t')\sigma\downarrow \equiv \lambda x.(t'\sigma_{Dom(\sigma)-\{x\}})\downarrow \equiv \lambda x.(t'\sigma_{Dom(\sigma)-\{x\}}\downarrow) \trianglerighteq (t'\sigma_{Dom(\sigma)-\{x\}}\downarrow)[x \mapsto c_x] \equiv (t'[x \mapsto c_x])\sigma_{Dom(\sigma)-\{x\}}\downarrow$. We also have $t'[x \mapsto c_x]\sigma_{Dom(\sigma)-\{x\}}\downarrow \trianglerighteq s\sigma_{Dom(\sigma)-\{x\}}\downarrow$ by induction hypothesis. Since there is no x occurrence in s, $s\sigma_{Dom(\sigma)-\{x\}}\downarrow \equiv s\sigma\downarrow$, which concludes the proof. □

Now we will define dependency pairs.

Definition 6: The set $DP_{l\to r}$ of *dependency pairs* of a rule $l \to r$ is defined as follows:

$$DP_{l\to r} = \{\langle l^\#, t^\# \rangle \mid t \in Cand(r)\}$$

DP_R denotes the collection of all dependency pairs of rules in HRS R.

Considering a dependency pair, it is allowed to rename bound variables and corresponding fresh constants introduced by $Cand$, since renaming bound variables in each rule $l \to r$ takes no effect in HRSs.

Example 7: Consider the following HRS:

$$R_1 = \begin{cases} map(\lambda x.F(x), nil) \to nil, \\ map(\lambda x.F(x), cons(X, XS)) \\ \qquad \to cons(F(X), map(\lambda x.F(x), XS)) \end{cases}$$

Then, the dependency pairs are

$\langle map^\#(\lambda x.F(x), cons(X, XS)),\ F(X)\rangle$,
$\langle map^\#(\lambda x.F(x), cons(X, XS)),\ map^\#(\lambda x.F(x), XS)\rangle$,
$\langle map^\#(\lambda x.F(x), cons(X, XS)),\ F(c_x)\rangle$.

We need to change the definition of the R-chains for higher-order variables as follows.

Definition 8: Let $\langle s_1, t_1\rangle \cdots \langle s_n, t_n\rangle$ be a (possibly infinite) sequence of dependency pairs for an HRS R. It is called an *R-chain* if there exist substitutions $\sigma_1, \ldots, \sigma_n$ such that $Var(\sigma_i)$ and $BV(s_i) \cup BV(t_i)$ are disjoint for all $i = 1, \ldots, n$ and one of the following conditions holds for all $i = 1, \ldots, n-1$:

(a) $top(t_i) \in D^\#$ and $t_i\sigma_i\downarrow \xrightarrow{*} s_{i+1}\sigma_{i+1}\downarrow$
(b) $top(t_i) \in V_h$, $t_i\sigma_i\downarrow \trianglerighteq s$ and $s^\# \xrightarrow{*} s_{i+1}\sigma_{i+1}\downarrow$ for some s (See Fig. 1).

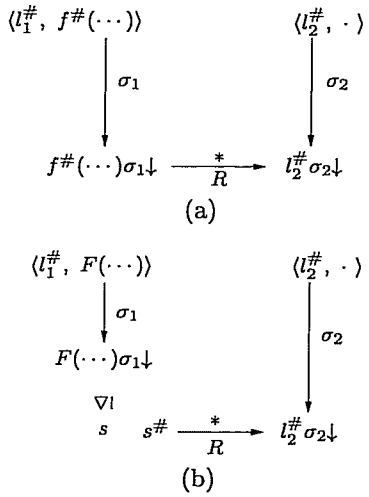


Fig. 1 Definition of R-chain.

Note that we use a substitution σ_i for each dependency pair $\langle s_i, t_i\rangle$ in the definition of the R-chains, although the original definition uses only one substitution. The reason is only for presentation convenience.

The following example shows that the subterm condition in Definition 8 (b) is indispensable.

Example 9: Consider the following HRS with $f, h \in C_{\alpha\to\alpha}$, $g \in C_{(\alpha\to\alpha)\to\alpha}$, $F \in V_{\alpha\to\alpha}$ and basic type α:

$$R_2 = \{\ f(g(\lambda x.F(x))) \to F(g(\lambda x.h(F(x))))\ \}$$

The only dependency pair $\langle s, t\rangle$ is

$$\langle f^\#(g(\lambda x.F(x))), F(g(\lambda x.h(F(x))))\rangle.$$

We have an infinite reduction sequence:

$$\begin{aligned}
&f(g(\lambda x.f(x)))\\
&\to_{R_2} f(g(\lambda x.h(f(x))))\\
&\to_{R_2} h(f(g(\lambda x.h(h(f(x))))))\\
&\to_{R_2} h(h(h(f(g(\lambda x.h(h(h(f(x)))))))))\\
&\ \ \vdots
\end{aligned}$$

and an R_2-chain $\langle s, t\rangle\langle s, t\rangle\cdots$ for $F\sigma_1 \equiv \lambda y.f(y)$, $F\sigma_2 \equiv \lambda y.h(f(y))$, $F\sigma_3 \equiv \lambda y.h(h(f(y)))$, $\cdots$.

However, we have no infinite R_2-chain, if we replace Condition (b) by

$$top(t_i) \in V_h \text{ and } (t_i\sigma\downarrow)^\# \xrightarrow{*} s_{i+1}\sigma_{i+1}\downarrow$$

in Definition 8 by dropping the subterm condition. This situation does not change even if we do not ignore subterms under higher-order variables in the definition of dependency pairs.

Before we show the relation between the R-chains and the termination of HRSs, we present a technical lemma to give clear proofs.

Definition 10: A term u in η-long β-normal form is said to be *essential* if

(a) u has an infinite reduction sequence, and
(b) any proper subterm of u has no infinite reduction sequence.

Note that the type of an essential term is basic since we assume the type of both sides of rewrite rules are basic. We also note that a term has at least one essential subterm if it has an infinite reduction sequence.

We say a substitution σ is *terminating*, if $F\sigma\downarrow$ is terminating for all variables F.

Lemma 11: Let r and u be in η-long β-normal form and σ be a substitution such that $\sigma_{Var(r)}$ is terminating and $Var(\sigma_{Var(r)})$ and $BV(r)$ are disjoint. If u is essential and $r\sigma\downarrow \trianglerighteq u$, then the following (a) or (b) holds for some $q \in Cand(r)$:

(a) $top(q) = top(u) \in D$ and $q\sigma\downarrow \equiv u$,
(b) $top(q)$ is a higher-order variable and $q\sigma\downarrow \trianglerighteq u$.

Proof We prove the lemma by induction on the definition of $\trianglerighteq$.

In case of $top(r) = F \in V$ we have $r \not\equiv F$. (Because, otherwise it follows from Condition (a) of essentiality of u and $F\sigma\downarrow \equiv r\sigma\downarrow \trianglerighteq u$ that $F\sigma\downarrow$ has an infinite reduction sequence, which contradicts that σ is terminating.) Thus, we have $F \in V_h$ and (b) follows by taking $q \equiv r \in Cand(r)$.

Consider the case of $r \equiv f(r_1, \ldots, r_n)$. Since $r\sigma\downarrow \equiv f(r_1\sigma\downarrow, \ldots, r_n\sigma\downarrow) \trianglerighteq u$, we have two subcases. If $r\sigma\downarrow \equiv u$, it follows from the essentiality of u that $f = top(u) \in D$. Hence, (a) holds by taking $q \equiv r \in Cand(r)$. Otherwise, we have $r_i\sigma\downarrow \trianglerighteq u$ for some i. Hence, we have (a) or (b) for some $q \in Cand(r_i)$ by the induction hypothesis. The lemma follows since we have $q \in Cand(r)$ by the definition of $Cand(r)$.

In case of $r \equiv \lambda x.s$, we have $r\sigma\downarrow \equiv \lambda x.(s\sigma_{Dom(\sigma)-\{x\}}\downarrow) \trianglerighteq u$. From the essentiality of u, we have $(s\sigma_{Dom(\sigma)-\{x\}}\downarrow)[x \mapsto c_x] \trianglerighteq u$. Since $(s\sigma_{Dom(\sigma)-\{x\}}\downarrow)[x \mapsto c_x] \equiv (s[x \mapsto c_x])\,\sigma_{Dom(\sigma)-\{x\}}\downarrow$, we have (a) or (b) for some $q \in Cand(s[x \mapsto c_x])$ by the induction hypothesis. The lemma follows since $q \in Cand(\lambda x.s)$. □

Theorem 12: If an HRS R has no infinite R-chain, it is terminating.

Proof Consider an infinite reduction sequence from a term v_0. Then, we will recursively define terms $v_1, v_2, \ldots$ and $u_1, u_2, \ldots$ each of which has an infinite reduction sequence. Let u_i be an essential subterm of v_{i-1} for all $i \geq 1$. The term v_i is defined from u_i by an infinite reduction sequence $u_i \xrightarrow{>\Lambda}{}^* w_i \xrightarrow{\Lambda} v_i \to \cdots$. The existence of v_i is assured by the essentiality of u_i.

Since $w_i \xrightarrow{\Lambda} v_i$ and w_i is essential, we have $w_i \equiv l_i\sigma_i\downarrow \to r_i\sigma_i\downarrow \equiv v_i$ for some rule $l_i \to r_i \in R$ and terminating substitution σ_i, where we can assume $Var(\sigma_i)$

and $BV(l_i) \cup BV(r_i)$ are disjoint without loss of generality. Since u_{i+1} is an essential subterm of v_i and σ_i is terminating, we have either (a) or (b) for some $q_i \in Cand(r_i)$ by Lemma 11:

(a) $top(q_i) = top(u_{i+1}) \in D$ and $q_i\sigma_i\downarrow \equiv u_{i+1}$,
(b) $top(q_i) \in V_h$ and $q_i\sigma_i\downarrow \trianglerighteq u_{i+1}$.

We have a dependency pair $\langle l_i^\#, q_i^\# \rangle$ in case (a), and a dependency pair $\langle l_i^\#, q_i \rangle$ in case (b). We write $\langle s_i, t_i \rangle$ for the dependency pair.

Now we show that the infinite sequence $\langle s_1, t_1 \rangle$ $\langle s_2, t_2 \rangle \cdots$ is an R-chain. Let i be a positive integer. In case of $t_i \equiv q_i^\#$ by (a), we have $t_i\sigma_i\downarrow \equiv q_i^\#\sigma_i\downarrow \equiv u_{i+1}^\# \xrightarrow{>\Lambda}{}^* w_{i+1}^\# \equiv (l_{i+1}\sigma_{i+1}\downarrow)^\#$. Since $top(w_i) \in D$, we have $(l_{i+1}\sigma_{i+1}\downarrow)^\# \equiv l_{i+1}^\#\sigma_{i+1}\downarrow \equiv s_{i+1}\sigma_{i+1}\downarrow$. In case of $t_i \equiv q_i$ by (b), we have $q_i\sigma\downarrow \trianglerighteq u_{i+1}$. Since u_{i+1} is essential, we have $top(u_{i+1}) \in D$. Hence, $u_{i+1}^\# \xrightarrow{>\Lambda}{}^* s_{i+1}\sigma_{i+1}\downarrow$ follows as in the previous case. □

Next, we will show the reverse of Theorem 12.

Theorem 13: If an HRS R is terminating, it has no infinite R-chain.

Proof Assume that there exists an infinite R-chain $\langle l_1^\#, t_1 \rangle \langle l_2^\#, t_2 \rangle \cdots$. Then, we have σ_i satisfying Condition (a) or (b) in Definition 8 for any positive integer i. We also have a rule $l_i \to r_i$ corresponding to $\langle l_i^\#, t_i \rangle$, where $Var(\sigma_i)$ and $BV(r_i)$ are assumed to be disjoint without loss of generality. In case of $t_i \equiv q_i^\#$ for some $q_i \in Cand(r_i)$ such that $top(q_i) \in D$, we have $r_i\sigma_i\downarrow \trianglerighteq q_i\sigma_i\downarrow$ by (a) and (b) of Proposition 5. Since $(q_i\sigma_i\downarrow)^\# \equiv t_i\sigma_i\downarrow \xrightarrow{*} l_{i+1}^\#\sigma_{i+1}\downarrow$ and $top(q_i) = top(l_i) \in D$, we have $q_i\sigma_i \downarrow \xrightarrow{*} l_{i+1}\sigma_{i+1} \downarrow \to r_{i+1}\sigma_{i+1} \downarrow$. Hence, $r_i\sigma_i\downarrow (\trianglerighteq \circ \to^+) r_{i+1}\sigma_{i+1}\downarrow$. In case of $top(t_i) \in V_h$, we have $t_i\sigma_i\downarrow \trianglerighteq s$ and $s^\# \xrightarrow{*} l_{i+1}^\#\sigma_{i+1}\downarrow$ for some s. Hence, $s \xrightarrow{*} l_{i+1}\sigma_{i+1}\downarrow$. Since $t_i \in Cand(r_i)$ from Definition 6, we have $r_i\sigma\downarrow \trianglerighteq t_i\sigma_i\downarrow \trianglerighteq s \xrightarrow{*} l_{i+1}\sigma_{i+1}\downarrow \to r_{i+1}\sigma_{i+1}\downarrow$ by (a) and (b) of Proposition 5. Thus, we have a sequence

$$r_1\sigma_1\downarrow (\trianglerighteq \circ \to^+) r_2\sigma_2\downarrow (\trianglerighteq \circ \to^+) \cdots .$$

Since R is terminating, we have the maximum length k of reduction sequences from $r_1\sigma_1\downarrow$. However, we can construct a reduction sequence from $r_1\sigma_1\downarrow$ longer than k from Proposition 3 (b), which is a contradiction. □

The following example shows why we do not construct dependency pairs from a subterm of a right-hand side located under a higher-order variable.

Example 14: Consider a terminating HRS

$$R_3 = \{\ f(\lambda x.F(x)) \to F(f(\lambda x.d))\ \}.$$

The only dependency pair is $\langle f^\#(\lambda x.F(x)),\ F(f(\lambda x.d)) \rangle$, and there is no infinite R-chain. However, if we include $\langle f^\#(\lambda x.F(x)), f^\#(\lambda x.d) \rangle$ in the set of dependency pairs, then it gives rise to an infinite R_3-chain since $f^\#(\lambda x.F(x))\sigma\downarrow \equiv f^\#(\lambda x.d) \equiv f^\#(\lambda x.d)\sigma\downarrow$ for $\sigma = [F \mapsto \lambda x.d]$.

4. Proving Termination

We can apply the method similarly to the first-order case for proving termination of HRSs. However, we have to find a quasi-ordering $\succeq$ such that not only it is a reduction quasi-ordering but it must also satisfy the subterm property and $f \succeq f^\#$ for all defined function symbol f, where we say a quasi-ordering $\succeq$ has the *subterm property* if $s \succeq t$ for any subterm t of s.

Theorem 15: Let R be an HRS. If there exists a reduction quasi-ordering $\succeq$ such that

(a) $\succeq$ has the subterm property,
(b) $f(t_1, \ldots, t_n) \succeq f^\#(t_1, \ldots, t_n)$ for all function symbols $f \in D$ and terms t_i,
(c) $l \succeq r$ for all rules $l \to r \in R$, and
(d) $s \succ t$ for all dependency pairs $\langle s, t \rangle$,

then R is terminating.

Proof Assume R is not terminating, then we have an infinite R-chain $\langle s_1, t_1 \rangle$ $\langle s_2, t_2 \rangle$ $\langle s_3, t_3 \rangle \cdots$ from Theorem 12. Then there exist substitutions σ_1, σ_2, ... such that for all i one of the following conditions holds:

(1) $top(t_i) \in D^\#$ and $t_i\sigma_i\downarrow \xrightarrow{*} s_{i+1}\sigma_{i+1}\downarrow$,
(2) $top(t_i) \in V_h$, $t_i\sigma_i\downarrow \trianglerighteq s$ and $s^\# \xrightarrow{*} s_{i+1}\sigma_{i+1}\downarrow$ for some s.

For i satisfying (1), we have $t_i\sigma_i\downarrow \succeq s_{i+1}\sigma_{i+1}\downarrow$ from Premise (c) and the closedness of $\succeq$ under substitutions. For i satisfying (2), we have $t_i\sigma_i\downarrow \succeq s$ from the subterm property, $s \succeq s^\#$ from (b), and $s^\# \succeq s_{i+1}\sigma_{i+1}\downarrow$ from Premise (c) and closedness of $\succeq$. It follows from $t_i\sigma_i\downarrow \succeq s_{i+1}\sigma_{i+1}\downarrow$ and $s_i \succ t_i$ for all i that we have an infinite sequence $s_1\sigma_1\downarrow \succ s_2\sigma_2\downarrow \succ \cdots$, which is a contradiction. □

The higher-order orderings [6]–[10] do not have the subterm property with respect to this paper's definition of subterms. Thus, we show that they have the subterm property if all constants c_x introduced by $Cand$ have less precedence than the corresponding typed algebraic term $\bot_\alpha$.

The *typed algebraic terms* [8] are first-order terms over the following variables V^A and constants C^A:

$$\begin{aligned} V^A &= \{X_\alpha \mid X \in V_\alpha, \alpha \in \tau_S\} \\ C^A &= \{f_{O(\alpha)} \mid f \in C_\alpha, \alpha \in \tau_S\} \\ &\quad \cup \{\bot_\alpha \mid |V_\alpha \neq \emptyset, \alpha \in \tau_S\} \\ &\quad \cup \{\lambda_{\alpha \to O(\beta)} \mid V_\alpha \neq \emptyset, C_\beta \neq \emptyset, \alpha, \beta \in \tau_S\}. \end{aligned}$$

Definition 16: The interpretation that maps normalized term to typed algebraic terms is defined as follows:

$$\|a(s_1,\ldots,s_n)\| = a_{O(\alpha)}(\|s_1\|,\ldots,\|s_n\|) \quad \text{where } n \geq 0 \text{ and } a \in V_\alpha \cup C_\alpha,$$
$$\|\lambda x.s\| = \lambda_{\alpha\to\beta}(\|s\|[x_{O(\alpha)} \mapsto \bot_{O(\alpha)}]) \quad \text{where } x \in V_\alpha \text{ and } s \in T_\beta.$$

We easily obtain the following proposition.

Proposition 17: Let c be a constant of type α. Then,

$$\|s\|[x_{O(\alpha)} \mapsto c_{O(\alpha)}] \equiv \|s[x \mapsto c]\|.$$

Lemma 18: Let $\sqsupseteq$ be a reduction quasi-ordering on typed algebraic terms having the subterm property. If $\bot_{O(\alpha)} \sqsupseteq c_{xO(\alpha)}$ for any $x \in V_\alpha$ and $\alpha \in \tau_S$, the quasi-order $\succeq$ on higher-order terms defined by

$$s \succeq t \iff \|s\| \sqsupseteq \|t\|$$

has the subterm property, that is, $s \trianglerighteq t$ implies $s \succeq t$.

Proof We show that $s \trianglerighteq t$ implies $s \succeq t$ by induction on the definition of $\trianglerighteq$. In case of $s \equiv t$, it is trivial. Consider the case that $s \equiv \lambda x.s'$ and $s'[x \mapsto c_x] \trianglerighteq t$, where x and s' are of type α and β, respectively. Then, we have $\|s\| \equiv \|\lambda x.s'\| \equiv \lambda_{\alpha\to\beta}(\|s'\|[x_{O(\alpha)} \mapsto \bot_{O(\alpha)}]) \sqsupseteq \|s'\|[x_{O(\alpha)} \mapsto \bot_{O(\alpha)}] \sqsupseteq \|s'\|[x_{O(\alpha)} \mapsto c_{x\,O(\alpha)}] \equiv \|s'[x \mapsto c_x]\|$ from the subterm property and the weakly-closedness of $\sqsupseteq$, the premise $\bot_{O(\alpha)} \sqsupseteq c_{x(\alpha)}$ and Proposition 17. Since we also have $\|s'[x \mapsto c_x]\| \trianglerighteq \|t\|$ from $s'[x \mapsto c_x] \trianglerighteq t$ by induction hypothesis, $s \succeq t$ follows. Consider the case that $s \equiv a(s_1,\ldots,s_n)$ and $s_i \trianglerighteq t$ for some i, where a is of type α. Then, we have $\|s\| \equiv a_{O(\alpha)}(\|s_1\|,\ldots,\|s_n\|) \sqsupseteq \|s_i\|$ by the subterm property of $\sqsupseteq$. Since we also have $\|s_i\| \sqsupseteq \|t\|$ from $s_i \trianglerighteq t$ by induction hypothesis, $s \succeq t$ follows. □

In the rest of this section, we briefly explain the higher-order recursive path ordering $\succeq_{horpo}$ [8], and show examples whose termination cannot be proved by $\succeq_{horpo}$ alone, but can be proved by combination of the dependency pair method and $\succeq_{horpo}$.

In order to use $\succeq_{horpo}$ for the termination proof of an HRS, we have to prepare

- an appropriate well-founded quasi-ordering $\succeq_\tau$[†] on types, and
- an appropriate quasi-ordering $\succeq_C$ on the constants of the typed algebraic terms.

The *higher-order recursive path ordering* $\succeq_{horpo}$[††] is defined by $s \succeq_{horpo} t \iff \|s\| \sqsupseteq \|t\|$ where $\sqsupseteq$ is the lexical combination of $\succeq_\tau$ and $\succeq_{rec}$, where $\succeq_{rec}$ is defined as follows:

$$s \equiv a(s_1,\ldots,s_n) \succeq_{rec} b(t_1,\ldots,t_m) \equiv t$$
$$\iff 1)\ a \notin V^A \text{ and } s_i \succeq_{rec} t \text{ for some } i,$$
$$2)\ a \succ_C b \text{ and } s \succ t_i \text{ for all } i, \text{ or}$$
$$3)\ a \sim_C b \text{ and } \{s_1,\ldots,s_n\} \succeq_{mul} \{t_1,\ldots,t_m\},$$

where $\succeq_{mul}$ is the multi-set extension of $\succeq_{rec}$.

Example 19: Consider the following HRS R_4 with $f, h \in C_{\alpha\to(\alpha\to\alpha)\to\alpha}$, $i \in C_{\alpha\to\alpha}$ and $X \in V_\alpha$:

$$R_4 = \begin{cases} f(X,\lambda x.x) \to h(X,\lambda x.x), \\ h(f(X,\lambda x.x),\lambda x.x) \to f(i(X),\lambda x.x) \end{cases}$$

In order to demonstrate its termination by $\succeq_{horpo}$, we need to show that

$$f_\alpha(X_\alpha,\lambda_{\alpha\to\alpha}(\bot_\alpha)) \sqsupseteq h_\alpha(X_\alpha,\lambda_{\alpha\to\alpha}(\bot_\alpha)), \text{ and}$$
$$h_\alpha(f_\alpha(X_\alpha,\lambda_{\alpha\to\alpha}(\bot_\alpha)),\lambda_{\alpha\to\alpha}(\bot_\alpha)) \sqsupseteq f_\alpha(i_\alpha(X_\alpha),\lambda_{\alpha\to\alpha}(\bot_\alpha)).$$

It is impossible; we have $\alpha \sim_\tau (\alpha\to\alpha)$ from the consistency of $\succeq_\tau$ and have $f_\alpha \succ_C h_\alpha$ from the former rule, and then we fail to show either of

$$f_\alpha(X_\alpha,\lambda_{\alpha\to\alpha}(\bot_\alpha)) \sqsupseteq f_\alpha(i_\alpha(X_\alpha),\lambda_{\alpha\to\alpha}(\bot_\alpha)), \text{ and}$$
$$\lambda_{\alpha\to\alpha}(\bot_\alpha)) \sqsupseteq f_\alpha(i_\alpha(X_\alpha),\lambda_{\alpha\to\alpha}(\bot_\alpha)).$$

On the other hand, the dependency pairs of R_4 are

$$f^\#(X,\lambda x.x) \to h^\#(X,\lambda x.x),$$
$$h^\#(f(X,\lambda x.x),\lambda x.x) \to f^\#(i(X),\lambda x.x).$$

Hence, we can show that this HRS is terminating by our method with $f_\alpha \sim_C h_\alpha \succ_C f^\#_\alpha \succ_C h^\#_\alpha \succ_C i_\alpha \succ_C \bot_\alpha \succ_C c_{x\alpha}$.

Example 20: Consider the following HRS R_5 with $altmap \in C_{(int\to int)\to list\to list}$, $pls, mlt \in C_{int\to int\to int}$, $nil \in C_{list}$, $cons \in C_{int\to list\to list}$, $F \in V_{int\to int}$, $XS \in V_{list}$, and $X, Y \in V_{int}$:

$$R_5 = \begin{cases} altmap(\lambda z.pls(z,Y),cons(X,XS)) \\ \to cons(pls(X,Y),altmap(\lambda z.mlt(z,Y),XS)), \\ altmap(\lambda z.mlt(z,Y),cons(X,XS)) \\ \to cons(mlt(X,Y),altmap(\lambda z.pls(z,Y),XS)), \\ altmap(\lambda z.F(z),nil) \to nil \\ pls(X,Y) \to X + Y \\ mlt(X,Y) \to X * Y \\ pls(2,2) \to mlt(2,2) \end{cases}$$

The termination of R_5 is not provable by $\succeq_{horpo}$ alone, but provable by our approach. Since $pls_{int} \sim_C mlt_{int}$ from the first and the second rules, we have $pls(2,2) \not\succ_{horpo} mlt(2,2)$. On the other hand, we have $pls(2,2) \succeq_{horpo} mlt(2,2)$ and also $pls^\#(2,2) \succ_{horpo} mlt^\#(2,2)$ by setting $pls^\# \succ_C mlt^\#$.

We can show the following theorem easily.

Theorem 21: Let R be an HRS and $\succeq$ be a reduction quasi-ordering on higher-order terms having subterm property. If $l \succ r$ for any rule in R, then the conditions (a)–(d) of Theorem 15 are satisfied by $\succeq$. Hence, the termination of R is provable by the dependency pair method.

[†] $\succeq_\tau$ should be consistent with the type structure (see Sect. 5 in Ref. [8]).

[††] This paper's definition of $\succeq_{horpo}$ is a simplified version. The original definition [8] allows the statuses *Mul* and *Lex* with function symbols.

From this theorem and Lemma 18, for every HRS where the termination is provable by an ordering like $\succeq_{horpo}$, its termination is also provable by Theorem 15.

5. Dependency Graph

The dependency graph method can be applied to enhance Theorem 15 in a similar way to [1]–[3].

The definition of the dependency graph is essentially the same as the original one.

Definition 22: Let DP be a set of dependency pairs, and $\rightharpoonup$ be a relation on higher-order terms. The dependency graph $G(DP, \rightharpoonup)$ is the directed graph of which nodes are elements of DP and there is an arc from $\langle s, t\rangle$ to $\langle s', t'\rangle$ if and only if there exist substitutions σ and σ' such that

(a) $top(t) \in D^{\#}$ and $t\sigma\downarrow \rightharpoonup s'\sigma'\downarrow$, or
(b) $top(t) \in V_h$, $t\sigma\downarrow \trianglerighteq u$ and $u^{\#} \rightharpoonup s'\sigma'\downarrow$ for some u.

Example 23: Consider the following HRS R_6:

$$R_6 = R_5 \cup \{mlt(0,0) \to pls(0,0)\}$$

The dependency graph $G(DP_{R_6}, \xrightarrow{*}_{R_6})$ is shown in Fig. 2.

Theorem 24: Let R be an HRS and $\rightharpoonup$ be a relation such that $\rightharpoonup \supseteq \xrightarrow{*}_R$. If there exists a reduction quasi-ordering $\succeq$ on higher-order terms such that

(a) $\succeq$ has the subterm property,
(b) $f(t_1, \ldots, t_n) \succeq f^{\#}(t_1, \ldots, t_n)$ for all function symbols $f \in D$ and terms t_i,
(c) $l \succeq r$ for all rules $l \to r \in R$,
(d) $s \succeq t$ for all dependency pairs $\langle s, t\rangle$ on a cycle in $G(DP_R, \rightharpoonup)$, and

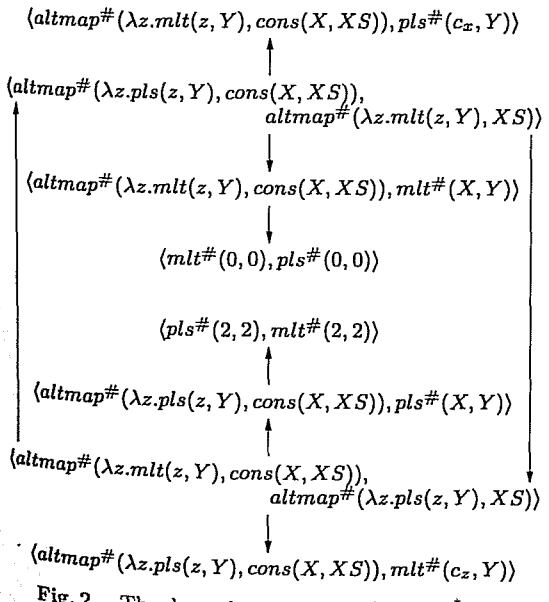


Fig. 2 The dependency graph $G(DP_{R_6}, \xrightarrow{*}_{R_6})$.

(e) $s \succ t$ for at leat one dependency pair $\langle s, t\rangle$ on every cycle in $G(DP_R, \rightharpoonup)$

then R is terminating.

Proof The proof is done similarly to the proof of Theorem 15. Assume that there is an infinite R-chain $\langle s_1, t_1\rangle\ \langle s_2, t_2\rangle\ \langle s_3, t_3\rangle \cdots$. Then, it corresponds to an infinite path in the dependency graph. We can show that $s_i\sigma_i\downarrow \succeq s_{i+1}\sigma_{i+1}\downarrow \succeq s_{i+2}\sigma_{i+2}\downarrow \succeq \cdots$ for some i from the conditions (a)–(d). Moreover, it is shown from the condition (e) that there are infinitely many ks such that $s_k\sigma_k\downarrow \succ s_{k+1}\sigma_{k+1}\downarrow$, which is a contradiction. □

Example 25: Consider the HRS R_6 in Example 23. It is impossible to show the termination of R_6 by using Theorem 15; we need $pls^{\#}_{int} \succ_C mlt^{\#}_{int}$ for $pls^{\#}(2,2) \succ_{horpo} mlt^{\#}(2,2)$, hence $mlt^{\#}(0,0) \not\succ_{horpo} pls^{\#}(0,0)$. However, we do not have to consider these dependency pairs according to Theorem 24. Therefore, we can show the termination of R_6 by dependency graph approach.

Remark that it is difficult to remove arcs from a node of the form $\langle s, F(t_1, \ldots, t_n)\rangle$ due to Definition 22 (b).

Note that it is obvious that Theorem 24 is strictly stronger than Theorem 15. From this fact and Theorem 21, it follows that applying our approach as a preprocessing step can never reduce the power.

6. Conclusion

By extending the dependency pair approach to the higher-order setting, one can benefit from the following features of dependency pairs:

- One can make a difference between usual function symbols and marked function symbols. This is the reason why Example 19 and 20 works.
- One can strip off constructor context around defined symbols when building dependency pairs.
- The dependency graph refinement can be helpful in the higher-order case as well to determine that the application of certain reduction steps never leads to an infinite reduction.
- Applying our approach as a preprocessing step for ordinary termination method can never reduce the power of the method.

However, unfortunately, the following benefits which are crucial for the power of dependency pairs in the first-order case do not hold for our extension any more:

- One can no longer perform argument filtering to eliminate certain arguments of function symbols since we require the subterm property.
- Thus, one can also no longer benefit substantially from the fact that one only needs a weakly monotonic ordering.

In particular, this also means that in the absence of higher-order terms, our method unfortunately does not reduce to the original dependency pair approach. An additional drawback is that if one has higher-order variables, then the termination proof is not simplified considerably by using our approach (this is due to the new dependency pairs built with higher-order variables).

Acknowledgment

We thank anonymous referees for their useful suggestions. We also thank Prof. Munehiro Iwami, Prof. Keiichiro Kusakari, Prof. Nobuo Kawaguchi, and members of TRS meeting JAPAN for discussion. This work is partly supported by Grants from Ministry of Education, Science and Culture of Japan #11680352.

References

[1] T. Arts and J. Giesl, "Automatically proving termination where simplification orderings fail," Proc. 22nd International Colloquium on Trees in Algebra and Programming, CAAP '97, in LNCS, vol.1214, pp.261–272, Springer-Verlag, 1997.

[2] T. Arts, "Automatically proving termination and innermost normalization of term rewriting systems," Ph.D. thesis, Utrecht University, The Netherlands, 1997.

[3] T. Arts and J. Giesl, "Termination of term rewriting using dependency pairs," Theoretical Computer Science, vol.236, pp.133–178, 2000.

[4] H. Barendregt, "Lambda calculi with types," in Handbook of Logic in Computer Science, eds. Abramsky et al., Oxford University Press, 1993.

[5] F. Baader and T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

[6] M. Iwami, M. Sakai, and Y. Toyama, "An improved recursive decomposition ordering for higher-order rewrite systems," IEICE Trans. Inf. & Syst., vol.E81-D, no.9, pp.988–996, 1998.

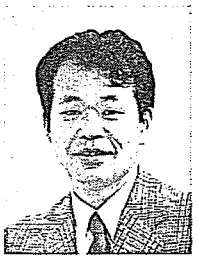
[7] M. Iwami and Y. Toyama, "Simplification ordering for higher-order rewrite systems," Technical Report, IS-RR-98-0024F, JAIST, 1998.

[8] J.-P. Jouannaud and A. Rubio, "Rewrite orderings for higher-order terms in η-long β-normal form and the recursive path ordering," Theoretical Computer Science, vol.208, pp.33–58, 1998.

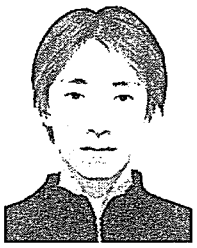
[9] O. Lysne and J. Piris, "A termination ordering for higher order rewrite systems," Proc. 6th International Conference on Rewriting Techniques and Applications, RTA '95, in LNCS, vol.914, pp.26–40, Springer-Verlag, 1995.

[10] C. Loría-Sáenz and J. Steinbach, "Termination of combined (rewrite and λ-calculus) systems," Proc. 3rd International Workshop on Conditional Term Rewriting Systems, CTRS '92, in LNCS, vol.656, pp.143–147, Springer-Verlag, 1993.

[11] T. Nipkow, "Higher-order critical pairs," Proc. 6th Annual IEEE Symposium on Logic in Computer Science, pp.342–349, 1991.



Masahiko Sakai completed the graduate course of Nagoya University in 1989 and became Assistant Professor, where he obtained a D.E. degree in 1992. From April 1993 to March 1997, he was Associate Professor at JAIST, Hokuriku. In 1996 he stayed at SUNY at Stony Brook for six months as Visiting Research Professor. Since April 1997, he has been Associate Professor at Nagoya University. He is interested in term rewriting systems, verification of specifications and software generation. He received the Best Paper Award from IEICE in 1992. He is member of JSSST.



Yoshitsugu Watanabe received the B.E. and M.S. degrees in Information Engineering from Nagoya University in 1996 and 1998, respectively. He engaged in research on term rewriting systems. Currently, he works at Mitsubishi Electric Co. Inazawa Works.

Toshiki Sakabe received B.E., M.E. and D.E. degrees from Nagoya University in 1972, 1974 and 1978, respectively. He was a research associate at Nagoya University during 1977–1985, and an associate professor at Mie University and Nagoya University during 1985–1987 and 1987–1993, respectively. He has been a professor at the Department of Information Engineering of Nagoya University since 1993. His research interests are in the field of theoretical foundations of software including algebraic specifications, rewriting computation, object-oriented computation and so on. He is a member of IPSJ, JSAI, JSSST and EATCS.