Termination of Simply-Typed Applicative TRSs

Takahito Aoto[1] and Toshiyuki Yamada[2]

[1] Research Institute of Electrical Communication, Tohoku University, Japan
aoto@nue.riec.tohoku.ac.jp
[2] Faculty of Engineering, Mie University, Japan
toshi@cs.info.mie-u.ac.jp

Abstract. Applicative term rewriting system handles terms having a tree structure with a function symbol or a variable at every leaf node and a unique application symbol at every internal node. Because of this term structure, direct application of standard termination proof methods, such as recursive path orderings, are often ineffective for applicative term rewriting systems. In this paper, we introduce a simply-typed version of applicative term rewriting systems and present a termination proof method for simply-typed applicative term rewriting systems.

1 Introduction

Applicative term rewriting system (TRS) is a popular encoding in term rewriting which can deal with higher-order functions. It is a first-order TRS where terms have a tree structure consisting of a function symbol or a variable at every leaf node and a unique application symbol at every internal node. For example, the higher-order function map, is expressed as the following applicative TRS:

$$\begin{array}{ll} (\mathsf{map}\text{'}F)\text{'}\mathsf{nil} & \to \mathsf{nil} \\ (\mathsf{map}\text{'}F)\text{'}((\mathsf{cons}\text{'}x)\text{'}xs) & \to (\mathsf{cons}\text{'}(F\text{'}x))\text{'}((\mathsf{map}\text{'}F)\text{'}xs) \end{array}$$

where “ ’ ” denotes the infix application symbol.

In an *applicative* term formulation, such as above, all subterms except constants and variables have the same leading symbol. This makes a contrast with the usual *functional* term formulation, such as $\mathsf{map}(F, \mathsf{cons}(x, xs))$. In applicative TRS, direct application of standard termination proof methods, such as recursive path orderings [4] and the dependency pair technique [3], is ineffective. This is because these techniques make use of the difference of leading function symbols.

In this paper, we introduce a simply-typed version of applicative TRSs and present a proof method for their termination. We give a transformation from applicative TRSs to TRSs in functional form for which standard termination proof methods are effectively applied.

2 Simply-Typed Applicative TRSs

For a set B of *basic types*, the set of *simple types* is the smallest set $\mathrm{ST}(B)$ such that (1) $B \subseteq \mathrm{ST}(B)$, and (2) $\tau \to \rho \in \mathrm{ST}(B)$ whenever $\tau, \rho \in \mathrm{ST}(B)$. The set $\mathrm{ST}(B)$ is abbreviated to ST. As usual we assume $\to$ is right-associative and omit unnecessary parentheses. Based on this abbreviation, each simple type τ is uniquely represented by $\tau_1 \to \tau_2 \to \cdots \to \tau_k \to \tau_0$ with $\tau_1, \ldots, \tau_k \in \mathrm{ST}(B)$ and $\tau_0 \in B$, where k is called the *arity* of τ and denoted by $\mathrm{arity}(\tau)$.

Each *function symbol* or *variable* is associated with its type. The sets of function symbols and variables of type τ are denoted by Σ^τ and V^τ. The sets of all function symbols and variables are denoted by Σ and V. The arity of a function symbol $f \in \Sigma^\tau$ is defined as

$\text{arity}(f) = \text{arity}(\tau)$. Apart from members of Σ, we assume that there is a binary infix function symbol " ' " called an *application symbol*. The set $\mathrm{AT}_{\mathrm{ST}}(\Sigma, V)^\tau$ of *simply-typed applicative terms* of type τ is defined as follows: (1) $\Sigma^\tau \cup V^\tau \subseteq \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)^\tau$, and (2) if $s \in \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)^{\tau\to\rho}$ and $t \in \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)^\tau$ then $(s\text{'}t) \in \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)^\rho$. Each simply-typed applicative term has a unique type and thus τ is also referred to as the type of s (denoted by type(s)). The arity of a simply-typed applicative term s is defined by its type. The set of all simply-typed applicative terms is denoted by $\mathrm{AT}_{\mathrm{ST}}(\Sigma, V)$ or $\mathrm{AT}_{\mathrm{ST}}$. The set of variables occurring in a term t is written as $\mathrm{V}(t)$. A term without variables is said to be *ground*. We assume the application symbol " ' " is left-associative and usually omit unnecessary parentheses as well as the outermost one. Based on this abbreviation, each term s is uniquely represented as $a\text{'}s_1\text{'}\cdots\text{'}s_n$ for some $a \in \Sigma \cup V$ and $s_1, \ldots, s_n \in \mathrm{AT}_{\mathrm{ST}}$, where a is called the *head* of the term s and denoted by head(s). A *substitution* is a mapping $\sigma : V \to \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)$ that satisfies the following conditions: (1) $\mathrm{Dom}(\sigma) = \{x \mid \sigma(x) \neq x\}$ is finite, and (2) for each $x \in \mathrm{Dom}(\sigma)$, x and $\sigma(x)$ have the same type. The homomorphic extension of σ to $\mathrm{AT}_{\mathrm{ST}}(\Sigma, V)$ is also denoted by σ. As usual, $\sigma(t)$ is written as $t\sigma$. A substitution is *ground* if $\sigma(x)$ is ground for any $x \in \mathrm{Dom}(\sigma)$.

A pair of simply-typed applicative terms written as $l \to r$ is a *simply-typed applicative rewrite rule* when (1) l and r have the same type, (2) $\text{head}(l) \in \Sigma$, and (3) $V(r) \subseteq V(l)$. A triple $\langle B, \Sigma, R\rangle$ consisting of a set B of basic types, a set Σ of function symbols, and a set R of rewrite rules is called a *simply-typed applicative TRS* (SATRS, for short). The *rewrite relation* $\to_{\mathcal{R}}$ induced by an SATRS $\mathcal{R} = \langle B, \Sigma, R\rangle$ is the smallest relation over $\mathrm{AT}_{\mathrm{ST}}(\Sigma, V)$ satisfying the following conditions: (1) $l\sigma \to_{\mathcal{R}} r\sigma$ for all $l \to r \in R$ and for all substitutions σ, and (2) if $s \to_{\mathcal{R}} t$ then $(u_1\text{'}s) \to_{\mathcal{R}} (u_1\text{'}t)$ and $(s\text{'}u_2) \to_{\mathcal{R}} (t\text{'}u_2)$ for any u_1 and u_2. An SATRS $\mathcal{R}$ is *terminating* if there is no infinite reduction sequences $s_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} \cdots$.

Example 1 (simply-typed applicative rewriting). Let $\mathcal{R} = \langle B, \Sigma, R\rangle$ be an SATRS where $B = \{\mathrm{N}, \mathrm{L}\}$, $\Sigma = \{\mathsf{0}^{\mathrm{N}}, \mathsf{s}^{\mathrm{N}\to\mathrm{N}}, \mathsf{id}^{\mathrm{N}\to\mathrm{N}}, \mathsf{add}^{\mathrm{N}\to\mathrm{N}\to\mathrm{N}}, \mathsf{nil}^{\mathrm{L}}, \mathsf{cons}^{\mathrm{N}\to\mathrm{L}\to\mathrm{L}}, \mathsf{map}^{(\mathrm{N}\to\mathrm{N})\to\mathrm{L}\to\mathrm{L}}\}$, and

$$R = \begin{cases} \mathsf{id}\text{'}x \to x & \\ \mathsf{add}\text{'}\mathsf{0} \to \mathsf{id} & \mathsf{add}\text{'}(\mathsf{s}\text{'}x)\text{'}y \to \mathsf{s}\text{'}(\mathsf{add}\text{'}x\text{'}y) \\ \mathsf{map}\text{'}F\text{'}\mathsf{nil} \to \mathsf{nil} & \mathsf{map}\text{'}F\text{'}(\mathsf{cons}\text{'}x\text{'}xs) \to \mathsf{cons}\text{'}(F\text{'}x)\text{'}(\mathsf{map}\text{'}F\text{'}xs) \end{cases}$$

Here is a rewrite sequence of $\mathcal{R}$:

$$\begin{aligned} \mathsf{map}\text{'}(\mathsf{add}\text{'}\mathsf{0})\text{'}(\mathsf{cons}\text{'}(\mathsf{s}\text{'}\mathsf{0})\text{'}\mathsf{nil}) &\to_{\mathcal{R}} \mathsf{map}\text{'}\mathsf{id}\text{'}(\mathsf{cons}\text{'}(\mathsf{s}\text{'}\mathsf{0})\text{'}\mathsf{nil}) \to_{\mathcal{R}} \mathsf{cons}\text{'}(\mathsf{id}\text{'}(\mathsf{s}\text{'}\mathsf{0}))\text{'}(\mathsf{map}\text{'}\mathsf{id}\text{'}\mathsf{nil}) \\ &\to_{\mathcal{R}} \mathsf{cons}\text{'}(\mathsf{s}\text{'}\mathsf{0})\text{'}(\mathsf{map}\text{'}\mathsf{id}\text{'}\mathsf{nil}) \to_{\mathcal{R}} \mathsf{cons}\text{'}(\mathsf{s}\text{'}\mathsf{0})\text{'}\mathsf{nil} \end{aligned}$$

3 Removing Head Variables by Cover Instantiation

An applicative term can be translated into the corresponding term in functional form if there is no variable at any head position. In this section, we introduce a transformation on SATRSs for removing head variables.

Each rewrite rule can be transformed into an equivalent set of instantiated rules if every variable is instantiated by appropriate terms which cover all instances. Such instantiation can be used to remove head variables in an SATRS.

Definition 1 (cover instantiation). Let $l \to r$ be a rewrite rule and x^τ a variable in l. A set S of terms of type τ is a *coverset* with respect to $l \to r$ and x if (1) every ground term t^τ can be expressed as the form $t = s\sigma$ for some $s \in S$ and substitution σ, and

(2) $\mathrm{V}(s) \cap (\mathrm{V}(l) \setminus \{x\}) = \emptyset$ for every $s \in S$. A mapping $\psi : V \to \mathcal{P}(\mathrm{AT}_{\mathrm{ST}})$ is a *covermap* with respect to $l \to r$ if (1) its domain includes $\mathrm{V}(l)$, (2) $\psi(x)$ is a coverset with respect to $l \to r$ and x for every $x \in \mathrm{V}(l)$, and (3) $\psi(x)$ and $\psi(y)$ do not share the same variable for different variables x and y. Let ψ be a covermap with respect to $l \to r$. This covermap is used to transform the rewrite rule into a set of its instances as follows:

$$\mathrm{Inst}(l \to r, \psi) = \{l\sigma \to r\sigma \mid \forall x \in \mathrm{V}(l)\ \ \sigma(x) \in \psi(x)\}$$

Let $\mathcal{R} = \langle B, \Sigma, R\rangle$ be an SATRS. An indexed set of mappings $\Psi = \{\psi_{l\to r} \mid l \to r \in R\}$ is a *covermap system* with respect to R if each $\psi_{l\to r}$ is a covermap with respect to $l \to r$. A covermap system is used to transform R into a set of instantiated rewrite rules as follows:

$$\mathrm{Inst}(R, \Psi) = \bigcup_{l\to r \in R} \mathrm{Inst}(l \to r, \psi_{l\to r})$$

Definition 2 (head variable). The set of *head variables* in a term t is defined as follows:

$$\begin{array}{lll} \mathrm{HV}(a) & = \emptyset & \text{if } a \in \Sigma \cup V \\ \mathrm{HV}(f\text{'}t) & = \mathrm{HV}(t) & \text{if } f \in \Sigma \\ \mathrm{HV}(x\text{'}t) & = \{x\} \cup \mathrm{HV}(t) & \text{if } x \in V \\ \mathrm{HV}(s\text{'}t) & = \mathrm{HV}(s) \cup \mathrm{HV}(t) & \text{otherwise} \end{array}$$

Lemma 1 (removing head variables). *Let $\mathcal{R} = \langle B,\ \Sigma, R\rangle$ be an SATRS. Suppose a covermap system $\Psi = \{\psi_{l\to r} \mid l \to r \in R\}$ satisfies the properties that, for all rewrite rules $l \to r \in R$, (1) $\mathrm{HV}(\psi_{l\to r}(x)) = \emptyset$ for any $x \in \mathrm{V}(l)$, and (2) $V \cap \psi_{l\to r}(x) = \emptyset$ for any $x \in \mathrm{HV}(l) \cup \mathrm{HV}(r)$. Define $R' = \mathrm{Inst}(R, \Psi)$ and $\mathcal{R}' = \langle B, \Sigma, R'\rangle$. Then, $s \to_{\mathcal{R}} t$ if and only if $s \to_{\mathcal{R}'} t$ for all ground terms s and t. Moreover, none of rewrite rules in R' contain a head variable.*

Because instantiated rewrite rules can simulate only ground rewrite steps, existence of a ground term of types in concern is required.

Definition 3 (ground term existence condition). Let $\mathcal{R} = \langle B, \Sigma, R\rangle$ be an SATRS. We say $\mathcal{R}$ satisfies the *ground term existence condition* (GTEC) if the following conditions are satisfied: (1) for every variable x appearing in R, there exists a ground term with the same type as x, and (2) for every function symbol $f \in \Sigma$ of type $\tau_1 \to \cdots \to \tau_n \to \rho$ ($\rho \in B$), there exists ground terms u_i of type τ_i for any $i = 1, \ldots, n$.

Lemma 2 (termination by cover instantiation). *Let $\mathcal{R} = \langle B, \Sigma, R\rangle$ be an SATRS satisfying GTEC and Ψ be a coverset system with respect to R. Define $\mathcal{R}' = \langle B, \Sigma, \mathrm{Inst}(R, \Psi)\rangle$. If $\mathcal{R}'$ is terminating then $\mathcal{R}$ is terminating.*

Based on the observations so far, we define a transformation on SATRSs for removing head variables which preserves non-termination property.

Definition 4 (transformation HVI). Let $\mathcal{R} = \langle B, \Sigma, R\rangle$ be an SATRS. The covermap system for *head variable instantiation* $\Psi_{\mathrm{HVI}} = \{\psi_{l\to r}\}_{l\to r\in R}$ is defined as

$$\psi_{l\to r}(x) = \begin{cases} \{f\text{'}x_1\text{'}\cdots\text{'}x_n \mid f \in \Sigma,\ \mathrm{type}(f) = \tau_1 \to \cdots \to \tau_n \to \mathrm{type}(x)\} & \text{if } x \in \mathrm{HV}(l) \cup \mathrm{HV}(r) \\ \{x\} & \text{otherwise} \end{cases}$$

for all $x \in \mathrm{V}(l)$ where $x_1, \cdots, x_n$ are pairwise distinct fresh variables. The transformation HVI is defined as:

$$\mathrm{HVI}(\mathcal{R}) = \langle B, \Sigma, \mathrm{Inst}(R, \Psi_{\mathrm{HVI}})\rangle$$

Example 2 (transformation HVI). Let $\mathcal{R} = \langle B,\, \Sigma, R\,\rangle$ be the SATRS given in Example 1. Then the transformed set of rewrite rules of $\mathrm{HVI}(\mathcal{R})$ are

$$\begin{cases} \mathsf{id}\text{'}x & \to x \\ \mathsf{add}\text{'}\mathsf{0} & \to \mathsf{id} \\ \mathsf{add}\text{'}(\mathsf{s}\text{'}x)\text{'}y & \to \mathsf{s}\text{'}(\mathsf{add}\text{'}x\text{'}y) \\ \mathsf{map}\text{'}F\text{'}\mathsf{nil} & \to \mathsf{nil} \\ \mathsf{map}\text{'}\mathsf{s}\text{'}(\mathsf{cons}\text{'}x\text{'}xs) & \to \mathsf{cons}\text{'}(\mathsf{s}\text{'}x)\text{'}(\mathsf{map}\text{'}\mathsf{s}\text{'}xs) \\ \mathsf{map}\text{'}\mathsf{id}\text{'}(\mathsf{cons}\text{'}x\text{'}xs) & \to \mathsf{cons}\text{'}(\mathsf{id}\text{'}x)\text{'}(\mathsf{map}\text{'}\mathsf{id}\text{'}xs) \\ \mathsf{map}\text{'}(\mathsf{add}\text{'}y)\text{'}(\mathsf{cons}\text{'}x\text{'}xs) & \to \mathsf{cons}\text{'}(\mathsf{add}\text{'}y\text{'}x)\text{'}(\mathsf{map}\text{'}(\mathsf{add}\text{'}y)\text{'}xs) \end{cases}$$

Lemma 3 (properties of transformation HVI). *Let $\mathcal{R} = \langle B, \Sigma, R\,\rangle$ be an SATRS satisfying GTEC. If* $\mathrm{HVI}(\mathcal{R})$ *is terminating then $\mathcal{R}$ is terminating. Moreover, all rewrite rules in* $\mathrm{HVI}(\mathcal{R})$ *do not contain any head variables in both hand sides.*

4 Translation to Functional Form

In this section, we introduce a translation from an SATRS to a TRS.

Definition 5 (translation FF). Let $\mathcal{R} = \langle B, \Sigma, R\,\rangle$ be an SATRS where all rules contain no head variables. The set of function symbols in functional form is defined as

$$\mathrm{FF}(\Sigma) = \{f_k \mid f \in \Sigma, 0 \le k \le \mathrm{arity}(f)\}$$

where the arity of f_k is k. For each applicative term $s \in \mathrm{AT}_{\mathrm{ST}}(\Sigma, V)$ without head variables, its functional form $\mathrm{FF}(s) \in \mathrm{T}(\mathrm{FF}(\Sigma), V)$ is defined as follows:

$$\mathrm{FF}(a\text{'}t_1\text{'}\cdots\text{'}t_n) = \begin{cases} a_n(\mathrm{FF}(t_1), \ldots, \mathrm{FF}(t_n)) & \text{if } a \in \Sigma \\ a & \text{if } a \in V \end{cases}$$

Note here that $a \in V$ implies $n = 0$ by the absence of head variable. The set of rules and the rewrite system are translated as follows:

$$\begin{aligned} &\mathrm{FF}(R) = \{\ \mathrm{FF}(l\text{'}x_1\text{'}\cdots\text{'}x_k) \to \mathrm{FF}(r\text{'}x_1\text{'}\cdots\text{'}x_k) \mid l \to r \in R,\ 0 \le k \le \mathrm{arity}(l)\ \} \\ &\mathrm{FF}(\mathcal{R}) = \langle \mathrm{FF}(\Sigma), \mathrm{FF}(R) \rangle \end{aligned}$$

where $x_1, \cdots, x_n$ are pairwise distinct fresh variables.

Example 3 (translation FF*).* Let B, Σ, and R be the ones in Example 1. Then, we have $\mathrm{FF}(\mathsf{map}\text{'}(\mathsf{add}\text{'}\mathsf{0})\text{'}(\mathsf{cons}\text{'}(\mathsf{s}\text{'}\mathsf{0})\text{'}\mathsf{nil})) = \mathsf{map}_2(\mathsf{add}_1(\mathsf{0}_0), (\mathsf{cons}_2(\mathsf{s}_1(\mathsf{0}_0), \mathsf{nil}_0)))$. The set $\mathrm{FF}(\mathrm{HVI}(R))$ is

$$\begin{cases} \mathsf{id}_1(x) & \to x \\ \mathsf{add}_1(\mathsf{0}_0) & \to \mathsf{id}_0 \\ \mathsf{add}_2(\mathsf{0}_0, x) & \to \mathsf{id}_1(x) \\ \mathsf{add}_2(\mathsf{s}_1(x), y) & \to \mathsf{s}_1(\mathsf{add}_2(x, y)) \\ \mathsf{map}_2(F, \mathsf{nil}_0) & \to \mathsf{nil}_0 \\ \mathsf{map}_2(\mathsf{s}_0, \mathsf{cons}_2(x, xs)) & \to \mathsf{cons}_2(\mathsf{s}_1(x), \mathsf{map}_2(\mathsf{s}_0, xs)) \\ \mathsf{map}_2(\mathsf{id}_0, \mathsf{cons}_2(x, xs)) & \to \mathsf{cons}_2(\mathsf{id}_1(x), \mathsf{map}_2(\mathsf{id}_1, xs)) \\ \mathsf{map}_2(\mathsf{add}_1(y), \mathsf{cons}_2(x, xs)) & \to \mathsf{cons}_2(\mathsf{add}_2(y, x), \mathsf{map}_2(\mathsf{add}_1(y), xs)) \end{cases}$$

Lemma 4 (simulation of rewriting). *Let $\mathcal{R} = \langle B,\ \Sigma, R\rangle$ be an SATRS where every rewrite rule contains no head variables. Let s and t be ground simply-typed applicative terms. Then $s \to_{\mathcal{R}} t$ implies* $\mathrm{FF}(s) \to_{\mathrm{FF}(\mathcal{R})} \mathrm{FF}(t)$.

Theorem 1 (soundness of $\mathrm{FF} \circ \mathrm{HVI}$**).** *Let $\mathcal{R} = \langle B,\ \Sigma, R\rangle$ be an SATRS satisfying GTEC. If* $\mathrm{FF}(\mathrm{HVI}(\mathcal{R}))$ *is terminating then $\mathcal{R}$ is terminating.*

Example 4 (termination proof I). Let $\mathcal{R}$ be the SATRS in Example 1. Then $\mathrm{FF}(\mathrm{HVI}(\mathcal{R}))$ is the one in Example 3. Its termination is shown by the recursive path ordering with the precedence $\mathsf{add}_1 > \mathsf{id}_0$; $\mathsf{map}_2 > \mathsf{add}_2, \mathsf{cons}_2 > \mathsf{id}_1, \mathsf{s}_1$. Therefore, by Theorem 1, $\mathcal{R}$ is terminating.

Example 5 (termination proof II). Let $\mathcal{R} = \langle B,\ \Sigma, R\rangle$ be an SATRS where $B = \{\mathrm{N}\}$, $\Sigma = \{0^{\mathrm{N}},\ \mathsf{s}^{\mathrm{N}\to\mathrm{N}},\ \mathsf{add}^{\mathrm{N}\to\mathrm{N}\to\mathrm{N}},\ \mathsf{mult}^{\mathrm{N}\to\mathrm{N}\to\mathrm{N}},\ \mathsf{rec}^{(\mathrm{N}\to\mathrm{N}\to\mathrm{N})\to\mathrm{N}\to\mathrm{N}\to\mathrm{N}},\ \mathsf{fact}^{\mathrm{N}\to\mathrm{N}}\}$ and

$$R = \begin{cases} \mathsf{add}\text{'}0\text{'}y \to y & \mathsf{add}\text{'}(\mathsf{s}\text{'}x)\text{'}y \to \mathsf{s}\text{'}(\mathsf{add}\text{'}x\text{'}y) \\ \mathsf{mult}\text{'}0\text{'}y \to 0 & \mathsf{mult}\text{'}(\mathsf{s}\text{'}x)\text{'}y \to \mathsf{add}\text{'}(\mathsf{mult}\text{'}x\text{'}y)\text{'}y \\ \mathsf{rec}\text{'}F\text{'}x\text{'}0 \to x & \mathsf{rec}\text{'}F\text{'}x\text{'}(\mathsf{s}\text{'}y) \to F\text{'}(\mathsf{s}\text{'}y)\text{'}(\mathsf{rec}\text{'}F\text{'}x\text{'}y) \\ \mathsf{fact} \to \mathsf{rec}\text{'}\mathsf{mult}\text{'}(\mathsf{s}\text{'}0) & \end{cases}$$

Termination of $\mathrm{FF}(\mathrm{HVI}(\mathcal{R}))$ is shown by using the lexicographic path ordering [5]. We note that termination of this SATRS can not be shown by other termination proof methods for lambda-free higher-order TRSs such as [6], [7] or (the product-free fragments of) [1], [2], [8], [9].

Acknowledgments

The authors thank Mario Rodríguez-Artalejo for his comments, by which our translation FF is inspired.

References

1. T. Aoto and T. Yamada. Proving termination of simply typed term rewriting systems automatically. *IPSJ Transactions on Programming*, 44(SIG 4 PRO 17):67–77, 2003. In Japanese.
2. T. Aoto and T. Yamada. Termination of simply typed term rewriting systems by translation and labelling. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications (RTA'03)*, volume 2706 of *Lecture Notes in Computer Science*, pages 380–394, 2003.
3. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
4. N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
5. S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, University of Illinois, 1980.
6. K. Kusakari. On proving termination of term rewriting systems with higher-order variables. *IPSJ Transactions on Programming*, 42(SIG 7 PRO 11):35–45, 2001.
7. K. Kusakari. Higher-order path orders based on computability. *IEICE Transactions on Information and Systems*, E87–D(2):352–359, 2003.
8. M. Lifantsev and L. Bachmair. An lpo-based termination ordering for higher-order terms without λ-abstraction. In *Proceedings of the 11th International Conference Theorem Proving in Higher Order Logics (TPHOLs'98)*, volume 1479 of *Lecture Notes in Computer Science*, pages 277–293. Springer-Verlag, 1998.
9. Y. Toyama. Termination of S-expression rewriting systems: Lexicographic path ordering for higher-order terms. In *Proceedings of the 15th International Conference on Rewriting Techniques and Applications (RTA'04)*, to appear.