単純型付項書換え系に関する考察

Note on Simply Typed Term Rewriting Systems

岩見 宗弘 †
Munehiro IWAMI

† 島根大学 総合理工学部
Faculty of Science and Engineering, Shimane University
munehiro@cis.shimane-u.ac.jp

Simply typed term rewriting systems is a framework of term rewriting allowing higher-order functions. In contrast to the usual higher-order rewrite systems, simply typed term rewriting systems without bound variables.

In this paper, we study the confluence of simply typed term rewriting systems. Also we consider the modularity of simply typed term rewriting systems.

1 Introduction

Higher-order rewriting is a natural extension of first-order term rewriting to reason with higher-order equations. An important application of higher-order rewrite systems is to model the basic mechanisms of higher-order functional programming languages and higher-order theorem provers.

Recently, several higher-order rewrite systems without bound variables were defined. Their rewrite systems are *not* complicated as previous higher-order rewrite systems defined by Nipkow [6]. Simply typed term rewriting systems proposed by Yamada [13] is a framework of term rewriting systems allowing higher-order functions. In contract to the usual higher-order rewrite systems, simply typed term rewriting systems without bound variables. Also, term rewriting systems with higher-order variables proposed by Kusakari [4], which are defined without a meta-language like the λ-calculus. Furthermore, higher-order rewrite systems without λ-abstraction, expressed as S-expression rewriting systems proposed by Toyama [12].

In this paper, we consider simply typed term rewriting systems proposed by Yamada. First, we study the confluence of simply typed term rewriting systems. Next, we consider the modularity of simply typed term rewriting systems.

In section 2, we give the definition of simply typed term rewriting systems. We study the confluence of simply typed term rewriting systems in section 3. We consider the modularity of simply typed term rewriting systems in section 4. Finally, we mention the related works of translation.

2 Preliminaries

We mainly follow basic definitions in the literature [1, 13]. In this section, we introduce the basic notions of simply typed term rewriting systems. We assume the reader to be familiar with (first-order) term rewriting systems (TRS, for short) [2, 7, 9].

For a set B of *basic types*, the set of *simple types* is the smallest set $ST(B)$ such that (1) $B \subseteq ST(B)$ and (2) $\tau_1 \times \ldots \times \tau_n \to \tau_0 \in ST(B)$ whenever $\tau_0, \tau_1, \ldots, \tau_n \in ST(B)$. Simple type is abbreviated as *type* and $ST(B)$ is written as ST.

Each *constant* or *variable* is associated with its type; the set of constants and variables of type τ are denoted by C^τ and V^τ, respectively. C and V denote the sets of all constants and variables, respectively: $C = \cup_{\tau \in ST} C^\tau$ and $V = \cup_{\tau \in ST} V^\tau$. We assume that V^τ is countably infinite for any $\tau \in ST$. The set $T_{ST}(C,V)^\tau$ of *simply typed terms* of type τ over C, V is defined as follows: (1) $C^\tau \cup V^\tau \subseteq T_{ST}(C,V)^\tau$ and (2) if $s \in T(C,V)_{ST}^{\tau_1 \times \cdots \tau_n \to \tau}$ and $t_i \in T(C,V)_{ST}^{\tau_i}$ for $i = 1, \ldots, n$ then $(s\ t_1 \ldots t_n) \in T_{ST}(C,V)^\tau$. A simply typed term s has type τ (denoted by s^τ) when $s \in T_{ST}(C,V)^\tau$. The set of $T_{ST}(C,V)$ of all simply typed terms is $\cup_{\tau \in ST} T(C,V)^\tau$; $T_{ST}(C,V)$ is abbreviated as T_{ST}. Let t be a term in T_{ST}. The head symbol $head(t)$ of t is defined as follows: (1) $head(t) = t$ if $t \in V \cup C$ and (2) $head(t) = head(s)$ if $t = (s\ t_1 \ldots t_n)$. The set of variables occurring in t is denoted by $Var(t)$.

A *position* is a sequence of natural numbers. The empty set ϵ is called the *root position*. The set of positions in a term t is denoted by $Pos(t)$. The subterm $t\ |_p$ of t at position p is defined as follows: (1)$t\ |_p = t$ if $p = \epsilon$ and (2) $t\ |_p = t_i\ |_q$ if $t = (t_0\ t_1 \ldots t_n)$ and $p = iq$. The term obtained from a term s by replacing its subterm at position p with a term t is denoted by $s[t]_p$. The set $Pos(t)$ is divided into three parts. We say $p \in Pos(t)$ is at a *variable position* of if $t\ |_p \in V$, at a constant position if $t\ |_p \in C$, otherwise p is at an application position.

We denote the set of variable positions, constant positions and application positions in a term t by $Pos_v(t)$, $Pos_c(t)$, $Pos_a(t)$, respectively.

A pair $\langle l, r\rangle$ of simply typed terms is a *simply typed rewrite rule* when (1) l and r have the same type, (2) $head(l) \in C$ and (3) $Var(r) \subseteq Var(l)$. A simply typed rewrite rule $\langle l, r\rangle$ is written as $l \to r$. A $\langle B, C, R\rangle$ consisting of a set B of basic types, a set C of constants and a set R of simply typed rewrite rules is called a *simply typed term rewriting systems* (STTRSs, for short).

A *substitution* σ is a function from V to $T_{ST}(C, V)$ such that its *domain*, defined as the set $\{x \in V \mid \sigma(x) \neq x\}$, is finite and $\sigma(x) \in T_{ST}(C, V)^\tau$ whenever $x \in V^\tau$. The homomorphic extension of σ to $T_{ST}(C, V)$ is also denoted by σ. $\sigma(t)$ is denoted by $t\sigma$

The rewrite relation $\to_{\mathcal{R}}$ induced by a STTRS $\mathcal{R}$ is the smallest relation over $T_{ST}(C, V)$ satisfying the following conditions: (1) $l\sigma \to r\sigma$ for all $l \to r \in R$ and for all substitutions σ and (2) if $s \to_{\mathcal{R}} t$ the $(s_0 \dots s \dots s_n) \to_{\mathcal{R}} (s_0 \dots t \dots s_n)$ for all $s_0, \dots, s_n$. A STTRS $\mathcal{R}$ is *confluent* if $\leftarrow^*_{\mathcal{R}} \cdot \to^*_{\mathcal{R}} \subseteq \to^*_{\mathcal{R}} \cdot \leftarrow^*_{\mathcal{R}}$. A STTRS $\mathcal{R}$ is *terminating* if there is no infinite rewrite sequence $t_0 \to_{\mathcal{R}} t_1 \to_{\mathcal{R}} \dots$.

Terms in which no variable occurs twice or more are called *linear*. A rewrite rule $l \to r$ is *left-linear* if l is a linear term. A STTRS is overlapping if there exists rewrite rules $l \to r$ and $l' \to r'$ without common variables (after renaming) and a non-variable position $p \in Pos_c(l) \cup Pos_a(l)$ such that $l \mid_p$ and l' are unifiable, and if $p = \epsilon$ then $l \to r$ is not obtained from $l' \to r'$ by renaming variable symbols. Let σ be a most general unifier of $l \mid_p$ and l'. We call $\langle l[r'\sigma]_p, r\sigma\rangle$ a *critical pair* of $\mathcal{R}$. A STTRS $\mathcal{R}$ is *orthogonal* if the rewrite rules in $\mathcal{R}$ are left-linear and no overlapping.

Let $\mathcal{R}$ be a STTRS. The *parallel reduction relation* induced by $\mathcal{R}$ is the smallest relation $\Vdash_{\mathcal{R}}$ such that (1) if $t \in V \cup C$ then $t \Vdash_{\mathcal{R}} t$, (2)if $l\sigma \to r\sigma$ then $l\sigma \Vdash_{\mathcal{R}} r\sigma$ for all $l \to r \in R$ and for all substitution σ, (3)if $n \geq 1$ and $s_i \Vdash_{\mathcal{R}} t_i$ for $i = 0, \dots, n$ then $(s_0\ s_1 \dots s_n) \Vdash_{\mathcal{R}} (t_0\ t_1 \dots t_n)$.

Example 2.1 *([1]) Let $\mathcal{R} = \langle B, C, R\rangle$ be a STTRS where $B = \{Nat, List\}$, $C = \{0^{Nat}, s^{Nat\to Nat}, []^{List}, :^{Nat\times List\to List}, map^{(Nat\to Nat)\times List\to List}, \circ^{(Nat\to Nat)\times(Nat\to Nat)\to(Nat\to Nat)}, twice^{(Nat\to Nat)\to(Nat\to Nat)}$ and*

$$R = \left\{ \begin{array}{l} (map\ F\ [])\to[] \\ (map\ F\ (:\ x\ xs))\to(:\ (F\ x)\ (map\ F\ xs)) \\ ((\circ\ F\ G)\ x)\to(F\ (G\ x)) \\ (twice\ F)\to(\circ\ F\ F) \end{array} \right.$$

3 Confluence of Simply Typed Term Rewriting Systems Revisited

In this section, we consider confluence of simply typed term rewriting systems.

Definition 3.1 *([1]) The set of arities is the smallest set $Ar(B)$ such that (1) $B \subseteq Ar(B)$ and (2) if $n \geq 1$ and $a_0, a_1, \dots, a_n \in Ar(B)$ then $\langle a_1 \dots a_n a_0\rangle \in Ar(B)$. The mapping $ar : ST(B) \to Ar(B)$ is defined as follows:*

$$ar(\tau) = \left\{ \begin{array}{l} \tau \quad if \quad \tau \in B, \\ \langle ar(\tau_1) \dots ar(\tau_n) ar(\tau_0)\rangle \\ if \quad \tau = \tau_1 \times \dots \times \tau_n \to \tau_0. \end{array} \right.$$

Arities can denote the simple types concisely. For example, $ar((Nat\to Nat) \times List\to List) = \langle\langle NatNat\rangle ListList\rangle$. We also employ the following conventions for simplicity:

1. $Nat, List$, etc. are abbreviated as N, L etc. when no confusion arises.
2. $\langle a_1 \dots a_n\rangle$ is as a_1^n when $a_1 = \dots = a_n$. (Note that this implies that $(N^2)^2$ differs from N^4; the former denotes $\langle\langle NN\rangle\langle NN\rangle\rangle$ while the latter is $\langle NNNN\rangle$.)
3. Outer most brackets of arities are omitted. (For example, $\langle\langle NL\rangle LL\rangle$ is written as $\langle NL\rangle LL$).

Aoto and Yamada gave the translation from STTRSs to many sorted TRSs [1]. A many sorted TRS is specified by a triple $\langle S, F, R\rangle$ where S is a set of sorts, F a set of S-sorted function symbols, and R a set of rewrite rules.

Definition 3.2 *([1]) A translation Θ from simply typed signature (terms, STTRSs) to many sorted signature (terms, many sorted TRSs, respectively).*

1. $\Theta(C) = \{@_a \mid a \in Ar(B) \setminus B\}$

   where each function symbol in $\Theta(C)$ is associated with its sort as follows:

   $$sort(f) = \left\{ \begin{array}{l} ar(\tau) \quad if \quad f \in C^\tau \\ \langle ar(\tau_1) \dots ar(\tau_n) ar(\tau_0)\rangle \\ \times a_1 \times \dots \times a_n \to a_0 \\ if \quad f = @_{\langle a_1 \dots a_n a_0\rangle} \end{array} \right.$$

2. $$\Theta(t) = \left\{ \begin{array}{l} t \quad if \quad t \in C \cup V, \\ @_{ar(\tau)(\Theta(s), \Theta(t_1), \dots, \Theta(t_n))} \\ if \quad t = (s^\tau\, t_1 \dots t_n) \end{array} \right.$$

3. $\Theta(R) = \{\Theta(l) \to \Theta(r) \mid l \to r \in R\}$
4. $\Theta(\langle B, C, R\rangle) = \langle Ar(B), \Theta(C), \Theta(R)\rangle$

Theorem 3.3 *([1]) Let $\mathcal{R}$ be a STTRS. Then, $s \to_{\mathcal{R}} t$ if and only if $\Theta(s) \to_{\Theta(\mathcal{R})} \Theta(t)$.*

Corollary 3.4 *Let $\mathcal{R}$ be a STTRS. $\mathcal{R}$ is confluent (terminating) if and only if $\Theta(\mathcal{R})$ is confluent (terminating, respectively).*

Thus, the confluence problem of STTRS is reduced to the confluence problem of many sorted TRSs. Clearly, many sorted TRS is confluence when its underlying unsorted TRS obtained by eliminating its sort information is confluence.

Lemma 3.5 *Let $\mathcal{R}$ be a STTRS. Then, $\mathcal{R}$ is orthogonal if and only if $\Theta(\mathcal{R})$ is orthogonal.*

Rosen showed that any orthogonal TRS is confluent [8].

Theorem 3.6 *([8]) Let $\mathcal{R}$ be a orthogonal TRS. Then, $\mathcal{R}$ is confluent.*

Yamada showed that any orthogonal STTRS is confluent by proving parallel reduction of STTRS has the diamond property [13]. We give the simple proof using the translation Θ.

Theorem 3.7 *([13]) Let $\mathcal{R}$ be a orthogonal STTRS. Then $\mathcal{R}$ is confluent.*

Proof. By lemma 3.5, $\Theta(\mathcal{R})$ is orthogonal. From theorem 3.6, $\Theta(\mathcal{R})$ is confluent. By corollary 3.4, $\mathcal{R}$ is confluent. □

Definition 3.8 *A STTRS $\mathcal{R}$ is* parallel closed *if and only if for every critical pair $\langle P, Q\rangle$ of $\mathcal{R}$ we have $P \Downarrow Q$.*

Lemma 3.9 *Let $\mathcal{R}$ be a STTRS. Then, $\mathcal{R}$ is left-linear parallel closed if and only if $\Theta(\mathcal{R})$ is left-linear parallel closed.*

Huet showed that any left-linear parallel closed TRS is confluent [3].

Theorem 3.10 *([3]) Let $\mathcal{R}$ be a left-linear parallel closed TRS. Then, $\mathcal{R}$ is confluent.*

Theorem 3.11 *Let $\mathcal{R}$ be a left-linear parallel closed STTRS. Then, $\mathcal{R}$ is confluent.*

Proof. By lemma 3.9, $\Theta(\mathcal{R})$ is left-linear parallel closed. From theorem 3.10, $\Theta(\mathcal{R})$ is confluent. By corollary 3.4, $\mathcal{R}$ is confluent. □

4 Modularity of Simply Typed Term Rewriting Systems

Modularity of first-order TRSs has been studied until now. Many results of modularity of TRSs have been obtained [2, 7, 9]. We consider modularity of STTRS in this section.

Definition 4.1 *Let $\langle B_1, C_1, R_1\rangle$ and $\langle B_2, C_2, R_2\rangle$ be STTRSs. Then, $\langle B_1, C_1, R_1\rangle$ and $\langle B_2, C_2, R_2\rangle$ are* disjoint *if they do not share constant symbols, that is, $C_1 \cap C_2 = \emptyset$.*

Definition 4.2 *A property P is* modular *for disjoint STTRSs if, for any disjoint STTRSs $\langle B_1, C_1, R_1\rangle$ and $\langle B_2, C_2, R_2\rangle$ that have the property P, their union $\langle B_1 \cup B_2, C_1 \cup C_2, R_1 \cup R_2\rangle$ also has the property P.*

Example 4.3 *Termination is* not *modular for disjoint STTRSs by following counterexample [10].*

Let $\mathcal{R}_1 = \langle B_1, C_1, R_1\rangle$ be a STTRS where $B_1 = \{Nat\}$, $C_1 = \{f^{Nat\times Nat\times Nat\to Nat}, 0^{Nat}, 1^{Nat}\}$ and $\mathcal{R}_2 = \langle B_2, C_2, R_2\rangle$ be a STTRS where $B_2 = \{Nat\}$, $C_2 = \{g^{Nat\times Nat\to Nat}\}$.

$$R_1 = \{\ (f\ 0\ 1\ x)\to(f\ x\ x\ x)$$

$$R_2 = \begin{cases} (g\ x\ y)\to x \\ (g\ x\ y)\to y \end{cases}$$

$\mathcal{R}_1$ and $\mathcal{R}_2$ are terminating, however $\mathcal{R}_1 \cup \mathcal{R}_2$ is not *terminating.*

$$\begin{aligned} &(f\ \underline{(g\ 0\ 1)}\ (g\ 0\ 1)\ (g\ 0\ 1)) \\ \to_{\mathcal{R}_1\cup\mathcal{R}_2} &(f\ 0\ \underline{(g\ 0\ 1)}\ (g\ 0\ 1)) \\ \to_{\mathcal{R}_1\cup\mathcal{R}_2} &\underline{(f\ 0\ 1\ (g\ 0\ 1))} \\ \to_{\mathcal{R}_1\cup\mathcal{R}_2} &\cdots \end{aligned}$$

Example 4.4 *Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be STTRSs. Let Θ be a translation in definition 3.2. Then, $\Theta(\mathcal{R}_1) \cup \Theta(\mathcal{R}_2)$ is* not *disjoint many sorted TRS in general. That is, translation Θ is not preserved disjointness.*

Let $\mathcal{R}_1 = \langle B_1, C_1, R_1\rangle$ be a STTRS where $B_1 = \{Nat\}$, $C_1 = \{f^{Nat\to Nat}\}$ and $\mathcal{R}_2 = \langle B_2, C_2, R_2\rangle$ be a STTRS where $B_2 = \{Nat\}$, $C_2 = \{g^{Nat\to Nat}\}$.

$$R_1 = \{\ (f\ x)\to x$$

$$R_2 = \{\ (g\ x)\to x$$

Then, $\Theta(\mathcal{R}_1) = \{@_{N^2}(f, x)\to x$ and $\Theta(\mathcal{R}_2) = \{@_{N^2}(g, x)\to x$.

Since $\Theta(C_1) \cap \Theta(C_2) = \{@_{N^2}\}$, $\Theta(\mathcal{R}_1)$ and $\Theta(\mathcal{R}_2)$ are not disjoint many sorted TRSs.

The set of function symbols occurring in many sorted TRS $\mathcal{R}$ is denoted by $Fun(\mathcal{R})$. We propose the improved version of translation Θ.

Definition 4.5 *We define a translation Γ from simply typed terms (STTRSs, signature) to many sorted terms (many sorted TRSs, signature, respectively).*

1. $\Gamma(t) = \begin{cases} t \quad if \quad t \in C \cup V, \\ @^t_{ar(\tau)(\Gamma(s),\Gamma(t_1),\ldots,\Gamma(t_n))} \\ if \quad t = (s^\tau t_1 \ldots t_n) \end{cases}$

2. $\Gamma(\mathcal{R}) = \{\Gamma(l) \to \Gamma(r) \mid l \to r \in \mathcal{R}\}$.

3. $\Gamma(C) = C \cup \{@_a^t \mid @_a^t \in Fun(\Gamma(\mathcal{R}))\}$ *where each function symbol in* $\Gamma(C)$ *is associated with its sort as follows:*

$$sort(f) = \begin{cases} ar(\tau) & if \quad f \in C^\tau, \\ \langle a_1 \ldots a_n a_0 \rangle \times a_1 \times \ldots \times a_n \to a_0 & \\ if \quad f = @^t_{\langle a_1 \ldots a_n a_0 \rangle} & \end{cases}$$

4. $\Gamma(\langle B, C, R \rangle) = \langle Ar(B), \Gamma(C), \Gamma(R) \rangle$.

Theorem 4.6 *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be disjoint STTRSs. Then,* $\Gamma(\mathcal{R}_1)$ *and* $\Gamma(\mathcal{R}_2)$ *are disjoint many sorted TRSs. That is, translation* Γ *is preserved disjointness.*

Example 4.7 *Let* $\mathcal{R}_1 = \langle B_1, C_1, R_1 \rangle$ *be a STTRS where* $B_1 = \{Nat\}$, $C_1 = \{f^{Nat \to Nat}\}$ *and* $\mathcal{R}_2 = \langle B_2, C_2, R_2 \rangle$ *be a STTRS where* $B_2 = \{Nat\}$, $C_2 = \{g^{Nat \to Nat}\}$, *i.e.* $C_1 \cap C_2 = \emptyset$.

$$\mathcal{R}_1 = \{ \ (f \ x) \to x$$

$$\mathcal{R}_2 = \{ \ (g \ x) \to x$$

Then, $\Gamma(\mathcal{R}_1) = \{@_{N^2}^{(f \ x)}(f, x) \to x$ *and* $\Gamma(\mathcal{R}_2) = \{@_{N^2}^{(g \ x)}(g, x) \to x$, *i.e.* $\Gamma(C_1) \cap \Gamma(C_1) = \emptyset$.

Since $\Gamma(C_1) \cap \Gamma(C_2) = \emptyset$, $\Gamma(\mathcal{R}_1)$ *and* $\Gamma(\mathcal{R}_2)$ *are disjoint many sorted TRSs.*

Theorem 4.8 *Let* $\mathcal{R}$ *be a STTRS. Then,* $s \to_{\mathcal{R}} t$ *if and only if* $\Gamma(s) \to_{\Gamma(\mathcal{R})} \Gamma(t)$.

Corollary 4.9 *Let* P *be a property of STTRS and* $\mathcal{R}$ *be a STTRS.* $\mathcal{R}$ *has a property* P *if and only if* $\Gamma(\mathcal{R})$ *has a property* P. *For example, property* P *is confluent (terminating and so on).*

Example 4.10 *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be disjoint confluent STTRSs.*

$\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are confluent*

$\iff$ $\Gamma(\mathcal{R}_1)$ *and* $\Gamma(\mathcal{R}_2)$ *are confluent*

$\iff$ $\Gamma(\mathcal{R}_1) \cup \Gamma(\mathcal{R}_2)$ *is confluent (by modularity of confluence for disjoint TRSs [11])*

$\iff$ $\Gamma(\mathcal{R}_1 \cup \mathcal{R}_2)$ *is confluent*

$\iff$ $\mathcal{R}_1 \cup \mathcal{R}_2$ *is confluent*

Hence, $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are confluent if and only if* $\mathcal{R}_1 \cup \mathcal{R}_2$ *is confluent. Therefore, confluence is modular for disjoint STTRSs.*

Corollary 4.11 *Let* P *be a property of TRS. If property* P *is modular for disjoint TRSs then property* P *is modular for disjoint STTRSs.*

5 Related Works of Translation

In this section, we mention the related works of translation.

Toyama proposed the *currying* as translation from typed S-expression rewriting system to curried typed S-expression rewriting system [12]. They showed the following lemma.

Lemma 5.1 *([12]) Let* Γ *be a currying and* $\mathcal{R}$ *a typed S-expression rewriting system. Then,* $s \to_{\mathcal{R}} t \Longrightarrow \Gamma(s) \to_{\Gamma(\mathcal{R})} \Gamma(t)$.

Kusakari and Chiba proposed the *currying* as translation from term rewriting system with higher-order variables to first-order term rewriting system [5]. They showed the following lemma.

Lemma 5.2 *([5]) Let* $@$ *be a currying and* $\mathcal{R}$ *a term rewriting system with higher-order variables. Then,* $s \to_{\mathcal{R}} t \iff s^{@} \to_{\mathcal{R}^{@}} t^{@}$.

6 Conclusion

In this paper, we have considered simply typed term rewriting systems. First, we have studied the confluence of simply typed term rewriting systems. Next, we have considered the modularity of simply typed term rewriting systems.

Acknowledgments

The author would like to thank Keiichiro Kusakari and Toshiyuki Yamada for their useful comments.

References

[1] T. Aoto and T. Yamada, "Termination of simply typed term rewriting by translation and labelling," Proc. 14th International Conf. on Rewriting Techniques and Applications, LNCS, 2706, pp.308–394, Springer-Verlag, 2004.

[2] F. Baader and T. Nipkow, "Term rewriting and all that," Cambridge University Press, 1998.

[3] G. Huet, "Confluence reductions: abstract properties and applications to term rewriting systems," J. ACM, 27, 4, pp.797–821, 1980.

[4] K. Kusakari, "On proving termination of term rewriting systems with higher-order variables," IPSJ Transactions on Programming, 42, SIG 7 (PRO 11), pp.35-45, 2001.

[5] K. Kusakari and Y. Chiba, "Higher-order Knuth-Bendix procedure and its applications," LA-Symposium (Summer), pp.9.1-9.12, 2004 (in Japanese).

[6] R. Mayr and T. Nipkow, "Higher-order rewrite systems and their confluence," Theoretical Computer Science, 192, pp.3–29, 1998.

[7] E. Ohlebusch, "Advanced topics in term rewriting," Springer-Verlag, 2002.

[8] B. K. Rosen, "Tree-manipulating systems and Church-Rosser theorems," J. ACM, 20, 1, pp.160–187, 1973.

[9] Terese, "Term rewriting systems," Cambridge University Press, 2003.

[10] Y. Toyama, "Counterexamples to termination for the direct sum of term rewriting systems," Information Processing Letters, 25, pp.141–143, 1987.

[11] Y. Toyama, "On the Church-Rosser property for the direct sum of term rewriting systems," J. ACM, 34, (1), pp.128–143, 1987.

[12] Y. Toyama, "Termination of S-expression rewriting systems: lexicographic path ordering for higher-order terms," Proc. 15th International Conf. on Rewriting Techniques and Applications, LNCS, 3091, Springer-Verlag, pp.40-54, 2004.

[13] T. Yamada, "Confluence and termination of simply typed term rewriting systems," Proc. 12th International Conf. on Rewriting Techniques and Applications, LNCS, 2051, pp.338–352, Springer-Verlag, 2001.