Termination of Higher-Order Rewrite Systems

by

Munehiro Iwami

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Yoshihito Toyama

School of Information Science
Japan Advanced Institute of Science and Technology

March 1999

Abstract

Higher-order rewriting is a natural extension of first-order term rewriting systems to reason with higher-order equations. Higher-order rewrite systems have been widely used as a model of higher-order functional and logic programming languages and as a basis of higher-order theorem provers.

The termination property is one of the fundamental notions of rewrite systems as computational models. In general, it is undecidable whether a given term rewriting system is terminating or not. A rewrite system is said to be terminating, if there is no infinite rewrite sequence. In terminating term rewriting systems, any strategy can compute one of answers for a given term.

In term rewriting systems, several sufficient conditions for proving the termination property have been successfully developed in particular cases. These techniques can be classified into two approaches: semantic methods and syntactic methods. Simplification orderings are representatives of syntactic methods. Many simplification orderings, for instance, the recursive path ordering, the improved recursive decomposition ordering and so on, have been proposed for term rewriting systems. The improved recursive decomposition ordering is one of the most powerful simplification orderings.

In higher-order rewrite systems, Jouannaud and Rubio gave a definition of recursive path ordering for higher-order rewrite systems by introducing an ordering on type structure recently.

In this thesis, we study the termination of higher-order rewrite systems by syntactic approaches.

First, we extend the improved recursive decomposition ordering to higher-order rewrite systems for proving termination of these systems. Our extension is inspired from Jouannaud and Rubio's idea. This ordering is called the *higher-order improved recursive decomposition ordering*. Further, we show that this ordering is more powerful than Jouannaud and Rubio's ordering.

Next, we introduce the notion of *simplification orderings* for higher-order rewrite systems. More precisely, we define the simplification ordering on algebraic terms where an algebraic term is η-long β-normal form without λ-abstractions. By this definition, we can analyze the termination of higher-order rewrite systems in abstract level. Further, we define a new recursive path ordering for higher-order rewrite systems, called the *higher-order recursive path ordering*. Our ordering extends Jouannaud and Rubio's ordering, which does not allow comparing two type incompatible terms. We show through several examples that our ordering can be applied to prove termination of higher-order rewrite systems to which Jouannaud and Rubio's ordering cannot be applied.

Finally, we extend the persistent property of termination to order sorted term rewriting systems. Zantema showed that termination is persistent for term rewriting systems without collapsing or duplicating rules. We show that Zantema's result can be extended to order sorted term rewriting systems, i.e, termination is persistent for order sorted term rewriting systems without collapsing, decreasing or duplicating rules.

Acknowledgments

I would like to thank my principal advisor Prof. Yoshihito Toyama and Prof. Masahiko Sakai for their helpful discussions, encouragements and suggestions. I would like to thank Prof. Tetsuo Ida, Prof. Hajime Ishihara and Prof. Hiroakira Ono for their useful comments and suggestions.

I also wish to thank Dr. Takahito Aoto, Mr. Keiichiro Kusakari, Mr. Takashi Nagaya, Dr. Hitoshi Ohsaki, Dr. Taro Suzuki and the members of Toyama-laboratory.

Finally, I wish to express my thanks to my parents and the late Dr. Yuko Yashima for their supports.

Contents

List of Figures

Chapter 1

Introduction

1.1 Term Rewriting Systems

Term rewriting systems can offer both flexible computing and effective reasoning with equations and have been widely used as a model of functional and logic programming languages and as a basis of theorem provers, symbolic computation, algebraic specification and verification.

The notion of rewriting is naturally introduced from equational reasoning. Now, we consider the following example. The equations define addition of natural numbers represented by the constant 0 and the successor function s.

$$\begin{cases} x + 0 = x \\ x + s(y) = s(x + y) \end{cases}$$

By applying these equations, we can reason $s(0) + s(s(0)) = s(s(s(0)))$ as follows.

$$\begin{aligned} s(0) + s(s(0)) &= s(s(0) + s(0)) \\ &= s(s(s(0) + 0)) \\ &= s(s(s(0))) \end{aligned}$$

In the above equational reasoning, the equation $s(0) + s(s(0)) = s(s(s(0)))$ can be interpreted as "$s(s(s(0)))$ is the result of computing $s(0) + s(s(0))$", but not via versa. This computational aspect for equational reasoning naturally leads to rewrite systems.

A rewrite system is a set of directed equations, called rewrite rules. A computation of rewrite system is performed by replacing some objects with other objects by using directed equations. Answers of computations are called *normal forms*.

If objects of rewriting are first-order terms, then the rewrite system is called (first-order) term rewriting system. The following rewrite system computes addition of natural numbers represented by the constant 0 and the successor function s.

$$\mathcal{R}_1 = \begin{cases} x + 0 \to x \\ x + s(y) \to s(x + y) \end{cases}$$

The following rewrite steps are computation of $s(0) + s(s(0))$ with respect to $\mathcal{R}_1$.

$$\begin{aligned} s(0) + s(s(0)) &\to_{\mathcal{R}_1} s(s(0) + s(0)) \\ &\to_{\mathcal{R}_1} s(s(s(0) + 0)) \\ &\to_{\mathcal{R}_1} s(s(s(0))) \end{aligned}$$

Term $s(s(s(0)))$ is a normal form of $s(0) + s(s(0))$ with respect to $\mathcal{R}_1$, i.e., it is the answer of this computation.

A textbook and surveys on term rewriting systems are written by Baader and Nipkow [4], Jouannaud and Dershowitz [11] and Klop [32].

1.2 Higher-Order Rewrite Systems

If objects of rewriting are higher-order terms, then the rewrite system is called higher-order rewrite system. Higher-order rewriting is a natural extension of first-order term rewriting systems to reason with higher-order equations [38, 56]. Higher-order rewrite systems have been widely used as a model of higher-order functional and logic programming languages like ML [41, 50], Haskell [17] and λ-Prolog [65], and as a basis of higher-order theorem provers like Isabelle [49, 51], TPS [1], Nuprl [7], and algebraic specification [6, 42]. First, Klop [32] has introduced the higher-order rewrite systems, called combinatory reduction systems, in 1980. Further, several higher-order rewrite systems have been suggested by Nipkow [43], van Oostrom [48], van de Pol [55], van Raamsdonk [58] and Wolfram [65]. Nipkow defined the higher-order rewrite systems in order to investigate the meta-theory of systems like Isabelle and λ-Prolog. The λ-calculus used in this system does not work as a computational mechanism but as a language for representing formulae. Recently the basic results for term rewriting systems have been lifted to higher-order rewrite systems by Nipkow and so forth. For instance, Nipkow showed critical pair lemma for higher-order rewrite systems [43] and confluence of orthogonal higher-order rewrite systems [44]. We take an η-long β-normal form as a higher-order term and follow the definition of higher-order rewrite systems introduced by Nipkow [38, 43] in this thesis.

Term rewriting systems cannot deal with the notion of higher-order directly. However, functional programming languages may contain higher-order functions. Also, the notion of bound variables is used in mathematics, logics and programming languages naturally. Since higher-order rewrite systems can deal with the notion of higher-order naturally and theoretically, they can denote higher-order functions and bound variables directly.

We consider the following higher-order rewrite system. The operation *map* applies the function $\lambda\, x.X(x)$ to each element in list.

$$\mathcal{R}_2 = \begin{cases} map(\lambda\, x.X(x), []) \to [] \\ map(\lambda\, x.X(x), N : L) \to X(N) : map(\lambda\, x.X(x), L) \\ x + 0 \to x \\ x + s(y) \to s(x + y) \end{cases}$$

Higher-order rewrite systems have the more expressive power than term rewriting systems as follows.

- Higher-order rewrite systems treat higher-order functions directly.

  We consider the rewrite rule $map(\lambda\, x.X(x), N : L) \to X(N) : map(\lambda\, x.X(x), L)$ in $\mathcal{R}_2$. Since function $\lambda\, x.X(x)$ is not first-order, the frame of term rewriting systems cannot express directly.

- Higher-order rewrite systems treat bound variables directly.

  The rewrite system $\mathcal{R}_2$ has the bound variable x. However bound variables cannot be expressed directly in the frame of term rewriting systems.

The following rewrite steps are computation of $map(\lambda\, x.s(0) + x, 0 : s(0) : [])$ with respect to $\mathcal{R}_2$. Term $map(\lambda\, x.s(0) + x, 0 : s(0) : [])$ denotes the operation that applies the function $\lambda\, x.s(0) + x$ to each element in list $[0, 1]$ (encoded as $0 : s(0) : []$).

$$\begin{aligned} map(\lambda\, x.s(0) + x, 0 : s(0) : []) &\to_{\mathcal{R}_2} s(0) + 0 : map(\lambda\, x.s(0) + x, s(0) : []) \\ &\to_{\mathcal{R}_2} s(0) + 0 : s(0) + s(0) : map(\lambda\, x.s(0) + x, []) \\ &\to_{\mathcal{R}_2} s(0) + 0 : s(0) + s(0) : [] \\ &\to_{\mathcal{R}_2} s(0) : s(0) + s(0) : [] \\ &\to_{\mathcal{R}_2} s(0) : s(s(0) + 0) : [] \\ &\to_{\mathcal{R}_2} s(0) : s(s(0)) : [] \end{aligned}$$

Term $s(0) : s(s(0)) : []$ is a normal form of $map(\lambda\, x.s(0) + x, 0 : s(0) : [])$ with respect to $\mathcal{R}_2$. Term $s(0) : s(s(0)) : []$ denotes the list $[1, 2]$.

1.3 Termination

A rewrite system is called *terminating* if there is no infinite rewrite sequence. The notion of termination for rewrite systems corresponds to the existence of answers of computations. So termination is the fundamental notion of term rewriting systems as computation models [12]. Huet and Lankford [18] showed that termination is undecidable for term rewriting systems in general. However, several sufficient conditions for proving this property have been successfully developed in particular cases. These techniques can be classified into two approaches: semantic methods and syntactic methods.

Simplification orderings are representatives of syntactic methods [9, 12, 60, 63]. The notion of simplification orderings was introduced by Dershowitz [9]. Many simplification

orderings have been defined on term rewriting systems. For instance, Plaisted [52, 53] defined the *path of subterm ordering*. Dershowitz [9] defined the *recursive path ordering*. Kamin and lévy [30] introduced the *lexicographic path ordering*. Jouannaud, Lescanne and Reinig [26] introduced the *recursive decomposition ordering*. Kapur, Narendran and Sivakumar [31] defined the *path ordering of Kapur, Narendran and Sivakumar*. Rusinowitch [59] defined the *improved recursive decomposition ordering*. Steinbach [61, 62] revisited the above orderings. The improved recursive decomposition ordering is one of the most powerful simplification orderings [61, 62]. An overview and comparison of simplification orderings have been given by Steinbach [60, 63].

In a semantical methods terms are interpreted in some well-founded ordered set in such a way that each rewrite sequence maps to a descending chain, and hence term rewriting system is terminating. The methods were studied by Lankford [35], Ben-Cherifa and Lescanne [5], Zantema [66] and so forth.

Termination is one of the most important properties of higher-order rewriting, like first-order rewriting. Termination criteria for higher-order rewrite systems have been studied since 1992. Loría-Sáenz and Steinbach [36] first extended recursive path ordering for proving termination of higher-order rewrite systems. Their extension method is based on an interpretation that maps higher-order terms to first-order terms. Lysne and Piris [37] also extended recursive path ordering for proving termination of higher-order rewrite systems by introducing the notions of termination functions, critical positions and dominations. We gave an extension of improved recursive decomposition ordering to higher-order rewrite systems in [19, 20, 21] according to Lysne and Piris's method. Jouannaud and Rubio [27] gave a simple definition of recursive path ordering for higher-order rewrite systems by introducing an ordering on type structure. Further, Walukiewicz [64] extended the Jouannaud and Rubio's ordering to AC-reduction ordering for proving termination of higher-order rewrite systems modulo AC-theories where A stands for associativity and C for commutativity. van de Pol [54, 55] and Kahrs [29] extended semantic methods in term rewriting systems to higher-order rewrite systems.

In proving termination of higher-order rewrite systems by syntactic approaches, main difficulty is to show the stability under substitutions of ordering. The partial ordering $>$ is called stable under substitutions if $s > t$ implies $s\theta\downarrow_\beta > t\theta\downarrow_\beta$ for any substitution θ where $s\theta\downarrow_\beta$ and $t\theta\downarrow_\beta$ are β-normal forms. By the subterm property, $X(a) > a$ holds where X is the higher-order variable. However, higher-order variables in a term destroy the stability under substitutions. For instance, we consider the substitution $\theta = \{X \leftarrow \lambda\, x.b\}$ where b is the new function symbol such that $a > b$. The ordering $>$ is not stable under substitutions since $X(a)\theta\downarrow_\beta = b < a = a\theta\downarrow_\beta$. Hence, the methods for guaranteeing the stability under substitutions in term rewriting systems cannot use in higher-order rewrite systems directly.

1.4 Overview and Main Results of the Thesis

We discuss the termination of higher-order rewrite systems by syntactic approaches in this thesis.

In chapter 2, we introduce definitions and notations of abstract reduction systems, sorted term rewriting systems and higher-order rewrite systems [3, 27, 28, 38, 56, 64].

In chapter 3, we introduce the notion of simplification orderings for term rewriting systems [9, 12, 39]. Also we give the recursive path ordering and improved recursive de-

composition ordering for term rewriting systems as an example of simplification orderings [12, 60, 63]. The improved recursive decomposition ordering is one of the most powerful simplification orderings for term rewriting systems.

In chapter 4, we extend the improved recursive decomposition ordering to higher-order rewrite systems for proving termination [21, 23]. Our extension method is inspired from Jouannaud and Rubio's one [27, 28] and the particular properties of improved recursive decomposition ordering. This ordering is called the *higher-order improved recursive decomposition ordering*. Further, we show that this ordering is more powerful than the Jouannaud and Rubio's ordering.

In chapter 5, we introduce the notion of *simplification orderings* for higher-order rewrite systems [24, 25]. More precisely, we define simplification orderings on algebraic terms where an algebraic term is in η-long β-normal form without λ-abstractions. By this definition, we can analyze the termination of higher-order rewrite systems in abstract level. Further, we define a new recursive path ordering for higher-order rewrite systems, called the *higher-order recursive path ordering*. We show that this ordering can be used for proving termination of higher-order rewrite systems. Our ordering extends Jouannaud and Rubio's ordering [27, 28], which does not allow comparing two type incompatible terms. We show through several examples that our ordering can be applied to prove termination of higher-order rewrite systems to which Jouannaud and Rubio's ordering cannot be applied.

In chapter 6, we extend the persistence of termination to order sorted term rewriting systems. First-order term rewriting systems are special cases of higher-order rewrite systems. So we analyze the termination of first-order term rewriting systems using the notion of persistence. A property P of term rewriting systems is *persistent* if for any many-sorted term rewriting system $\mathcal{R}$, $\mathcal{R}$ has the property P if and only if its underlying term rewriting system $\Theta(\mathcal{R})$, which results from $\mathcal{R}$ by omitting its sort information, has the property P. Usual many-sorted term rewriting system was extended with ordered sorts by Aoto and Toyama [3]. And it was shown that the persistency of confluence [2] is preserved for this extension in [3]. Zantema [66] showed that termination is persistent for term rewriting systems without collapsing or duplicating rules. We show that the above Zantema's result is preserved for Aoto and Toyama's extension in the subclass of order sorted term rewriting systems, i.e., termination is persistent for order sorted term rewriting systems without collapsing, decreasing or duplicating rules.

Finally, we summarize the main results in this thesis:

- We extend the improved recursive decomposition ordering to higher-order rewrite systems for proving termination.
- We introduce a framework of simplification ordering and the new recursive path ordering in higher-order rewrite systems for proving termination.
- We show that termination is persistent for order sorted term rewriting systems without collapsing, decreasing or duplicating rules.

Chapter 2

Preliminaries

In this chapter, we introduce the notions of abstract reduction systems, sorted term rewriting systems and higher-order rewrite systems. We mainly follow the basic notations in the literatures [3, 27, 28, 38, 56, 55, 64].

2.1 Abstract Reduction Systems

We introduce an abstract notion of rewriting and partial orderings, before define term rewriting systems and higher-order rewrite systems. Term rewriting systems can be considered as abstract reduction systems such that objects of rewriting are terms. Higher-order rewrite systems can be considered as abstract reduction systems such that objects of rewriting are higher-order terms. Many properties of rewrite systems can be stated in abstract reduction systems.

Definition 2.1.1 An *abstract reduction system* (ARS for short) is a pair $\mathcal{A} = \langle A, \rightarrow \rangle$ consisting of a set A and a binary relation $\rightarrow \subseteq A \times A$. Instead of $(a, b) \in \rightarrow$ we write $a \rightarrow b$.

Definition 2.1.2 Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS.

- Identity of elements in A is denoted by $=$.
- The relation $\rightarrow^+$ is the transitive closure of $\rightarrow$.
- The relation $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$.
- The relation $\leftrightarrow^*$ is the reflexive, symmetric and transitive closure of $\rightarrow$.

Definition 2.1.3 Let $\mathcal{A} = \langle A, \rightarrow \rangle$ be an ARS. Let $a \in A$ be given.

- If there is no element $b \in A$ such that $a \rightarrow b$, then we say that $a \in A$ is a *normal form* (with respect to $\rightarrow$).

- If $b \in A$ is a normal form such that $a \rightarrow^* b$ then we say that b *is a normal form of* a.
- A *rewrite sequence* (or *reduction sequence*) from a is a finite or infinite sequence $a = a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots$.
- a is *terminating* if there is no infinite rewrite sequence $a = a_0 \rightarrow a_1 \rightarrow a_2 \rightarrow \ldots$.
- a is *confluent* if for any b, $c \in A$ such that $a \rightarrow^* b$ and $a \rightarrow^* c$, there exists $d \in A$ such that $b \rightarrow^* d$ and $c \rightarrow^* d$.
- An ARS $\mathcal{A} = \langle A, \rightarrow \rangle$ is *terminating* (*confluent*) if for any $a \in A$, a is terminating (confluent).

A binary relation on A is called a *(strict) partial ordering* on A if it is irreflexive and transitive on A. The partial ordering is usually denoted by $>$. The converse of $>$ is denoted by $<$. The reflexive closure of $>$ is denoted by $\geq$.

A partial ordering $>$ on A is *total* if for any a, $b \in A$ we have either $a > b$ or $a = b$ or $a < b$.

Definition 2.1.4 A partial ordering $>$ on A is *well-founded* if $>$ has no infinite descending sequences, i.e., there is no sequence of the form $a_0 > a_1 > a_2 > \ldots$ of elements in A.

Definition 2.1.5 A partial ordering $>$ on A is a *well-partial ordering* if for every infinite sequence $a_0, a_1, a_2, \ldots$ of elements in A there are indexes k, l $(k < l)$ such that $a_k \leq a_l$. Given a well-partial ordering $>$ on A, $\langle A, > \rangle$ is called a *well-partially ordered set*.

Note that $\langle A_1, >_1 \rangle$, $\langle A_2, >_2 \rangle$ are well-partially ordered sets and $A_1 \cap A_2 = \emptyset$ then $\langle A_1 \cup A_2, >_1 \cup >_2 \rangle$ is also a well-partially ordered set.

A binary relation on A is called a *quasi-ordering* on A if it is reflexive and transitive on A. The quasi-ordering is usually denoted by $\gtrsim$. The converse of $\gtrsim$ is denoted by $\lesssim$. The strict part of a quasi-ordering $\gtrsim$ is a partial ordering $>$ defined as $\gtrsim \setminus \lesssim$. Every quasi-ordering $\gtrsim$ induces an equivalence relation $\sim$ defined as $\gtrsim \cap \lesssim$. It is easy to see that $> = \gtrsim \setminus \sim$.

Definition 2.1.6 A quasi-ordering $\gtrsim$ on A is *well-founded* if its strict part is well-founded.

The multiset extension and the lexicographic extension are essential methods in constructing more complex order structures from simple ones.

Definition 2.1.7 A *multiset* over A is an unordered collection of elements of A in which elements may have multiple occurrences. Given a binary relation $>$ on A, the *multiset extension* $>^{mult}$ or $\gg$ is defined as the transitive closure of the following relation $\Rightarrow$ on the set of multisets of elements in A. $M \cup \{a\} \Rightarrow M \cup \{b_1,\ldots,b_n\}$ where $n \geq 0$ and $a > b_i$ for any $i \in \{1,\ldots,n\}$.

The important property of the multiset extension is that $>$ is a well-founded ordering on A if and only if $>^{mult}$ is a well-founded ordering on multisets of elements of A [10].

Definition 2.1.8 Given a binary relation $>$ on A, the *lexicographic extension* $>^{lex}$ on A^n for some fixed n is defined as follows: $\langle\ a_1,\ldots,a_k\ \rangle >^{lex} \langle\ b_1,\ldots,b_m\ \rangle$ if and only if $m < k$ and $\forall j\ (1 \leq j \leq m)$, $a_j = b_j$, or $\exists j\ (1 \leq j \leq min\{m,k\})$ such that $a_j > b_j$ and $\forall i\ (1 \leq i < j)$, $a_i = b_i$.

It is well-known that $>$ is a well-founded ordering on A if and only if the lexicographic extension $>^{lex}$ is a well-founded ordering on A^n.

2.2 Sorted Term Rewriting Systems

In this section, we introduce the basic notions of sorted term rewriting systems. Sorted term rewriting systems are considered as higher-order rewrite systems without higher-order functions or bound variables. Usual term rewriting systems [4] are considered as special cases of sorted term rewriting systems.

Let $\mathcal{S}$ be a set of *sorts* or *basic types* and $\mathcal{V}$ be a set of countably infinite *sorted variables*. We assume that $\mathcal{S}$ is equipped with a well-founded partial ordering $\succ$. We write $b \succeq b'$ if and only if $b \succ b'$ or $b = b'$.

We assume there is a set $\mathcal{V}^b$ of countably infinite variables of sort b for each sort $b \in \mathcal{S}$. Let $\mathcal{F}$ be a set of sorted function symbols. We assume that each sorted function symbol $f \in \mathcal{F}$ is given with the sorts of its arguments and the sort of its value, all of which are included in $\mathcal{S}$. We write $f{:}b_1 \times \ldots \times b_n \rightarrow b'$ if and only if f takes n arguments of sorts $b_1,\ldots,b_n$ respectively to a value of sort b'. Function symbols of 0 argument are *constants*.

Definition 2.2.1 The set $\mathcal{T}(\mathcal{F},\mathcal{V}) = \cup_{b\in\mathcal{S}}\ \mathcal{T}(\mathcal{F},\mathcal{V})^b$ of all *sorted terms* built from $\mathcal{F}$ and $\mathcal{V}$ is defined as follows:

(1) $\mathcal{V}^b \subseteq \mathcal{T}(\mathcal{F},\mathcal{V})^b$,

(2) $f{:}b'_1 \times \ldots \times b'_n \rightarrow b'$, $t_i \in \mathcal{T}(\mathcal{F},\mathcal{V})^{b_i}$ and $b_i \preceq b'_i$ $(i = 1,\ldots,n)$ then $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})^{b'}$. Here $\mathcal{T}(\mathcal{F},\mathcal{V})^b$ denotes the set of all terms of sort b.

We define the set of all *strict sorted terms* if (2) is replaced by (2′) if $f{:}b'_1 \times \ldots \times b'_n \rightarrow b'$, $t_i \in \mathcal{T}(\mathcal{F},\mathcal{V})^{b_i}$, $b_i \preceq b'_i$ $(i = 1,\ldots,n)$ and $b_j = b'_j$ whenever $t_j \in \mathcal{V}$ then $f(t_1,\ldots,t_n) \in \mathcal{T}(\mathcal{F},\mathcal{V})^{b'}$. We write $t : b$ if t is of sort b. $\mathcal{V}(t)$ denotes the set of all variables that appear in t. $\mathcal{T}(\mathcal{F},\mathcal{V})^b$ and $\mathcal{T}(\mathcal{F},\mathcal{V})$ are abbreviated as $\mathcal{T}^b$ and $\mathcal{T}$, respectively.

Let $\Box^b$ be a special constant (*hole*) of sort b. Elements of $\mathcal{T}(\mathcal{F} \cup \{\Box^b \mid b \in \mathcal{S}\},\mathcal{V})$ are called *contexts* over $\mathcal{T}(\mathcal{F},\mathcal{V})$. We write $C{:}b_1 \times \ldots \times b_n \to b'$ if and only if the sort of context C is b' and it has n holes $\Box^{b_1},\ldots,\Box^{b_n}$. If $C{:}b'_1 \times \ldots \times b'_n \to b'$ and $t_1{:}b_1, \ldots, t_n{:}b_n$ with $b_i \preceq b'_i$ $(i = 1,\ldots,n)$ then $C[t_1,\ldots,t_n]$ denotes the term obtained from C by replacing holes with $t_1,\ldots,t_n$ from left to right. A context that contains one hole is denoted by $C[\ \ ]$. A term t is said to be a *subterm* of s if and only if $s = C[t]$ for some context C.

A *position* or *occurrence* in a term can be viewed as a finite sequence of natural numbers. $O(t)$ denotes the set of all positions of a term t. $Ot(t)$ denotes the set of all *terminal positions* (positions of all leaves) of t. The letter ϵ denotes the *root position*. The letters p, q, w, z stand for positions. We write $w \preceq z$ if w is a prefix of z.

The *subterm of* t *at position* p is denoted by $t|_p$ and we write $t \trianglerighteq t|_p$. If t and $t \neq t|_p$ then $t|_p$ is called a *proper subterm* of t, denoted by $t \triangleright t|_p$.

A *substitution* θ is a mapping from $\mathcal{V}$ to $\mathcal{T}$ such that $x \in \mathcal{V}^b$ implies $\theta(x) \in \mathcal{T}^b$. Substitutions are extended to homomorphisms from $\mathcal{T}$ to $\mathcal{T}$. $\theta(t)$ is usually written as $t\theta$.

A *sorted rewrite rule* on $\mathcal{T}$ is a pair $l \to r$ such that $l \notin \mathcal{V}$, $\mathcal{V}(r) \subseteq \mathcal{V}(l)$, l and r are strict and if $l : b$ and $r : b'$ then $b \succeq b'$.

Definition 2.2.2 A *sorted term rewriting system* (STRS) is a pair $(\mathcal{F},\mathcal{R})$ where $\mathcal{F}$ is a set of sorted function symbols and $\mathcal{R}$ is a set of sorted rewrite rules on $\mathcal{T}(\mathcal{F},\mathcal{R})$. $(\mathcal{F},\mathcal{R})$ is often abbreviated as $\mathcal{R}$ and in that case $\mathcal{F}$ is defined to be the set of function symbols that appear in $\mathcal{R}$.

Given a STRS $\mathcal{R}$, we write $s \to_{\mathcal{R}} t$ if and only if $s = C[l\theta]$ and $t = C[r\theta]$ for some rewrite rule $l \to r \in \mathcal{R}$, context C and substitution θ. We call $s \to_{\mathcal{R}} t$ a *rewrite step* or *reduction* from s to t of $\mathcal{R}$. $l\theta$ is called *redex* of this rewrite step.

Usual *many-sorted TRSs* are special cases of STRSs, i.e, $\mathcal{S}$ is equipped with a empty relation $\emptyset$. Also, usual TRSs are special cases of STRSs, i.e., $\mathcal{S}$ is a singlton set with a empty relation $\emptyset$. We write just TRS instead of usual TRS.

If $r \in \mathcal{V}$ then the rewrite rule $l \to r$ is said to be *collapsing*. If some variable has more occurrences in r than it has l then the rewrite rule $l \to r$ is said to be *duplicating*. If $l : b$, $r : b'$ and $b \succ b'$ then the rewrite rule $l \to r$ is said to be *decreasing*.

A binary relation $>$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$ is said to be *stable under contexts* if $s > t$ then $C[s] > C[t]$ for any context $C[\ \ ]$. A binary relation $>$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$ is said to be *stable under substitutions* if $s > t$ then $s\theta > t\theta$ for any substitution θ. A *rewrite relation* is a bianry relation on terms that is stabel under contexts and substitutions.

2.3 Higher-Order Terms

In this section, we introduce the notion of higher-order terms and the simply typed λ-calculus [16]. More precisely, a higher-order term is an η-long β-normal simply typed λ-terms in this thesis. Higher-order rewrite systems defined by Nipkow [38, 43] have simply typed λ-calculus as a meta language.

Let $\mathcal{S}$ be a set of basic types or sorts, b, b', b'',.... The set $\mathcal{T}_\mathcal{S}$ of *types* is generated from the set of basic types by the *constructor* $\rightarrow$ as follows: $\mathcal{T}_\mathcal{S} := \mathcal{S} \mid \mathcal{T}_\mathcal{S} \rightarrow \mathcal{T}_\mathcal{S}$. We use σ, τ and ρ to denote types. We use the abbreviation $\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \tau$ for $\sigma_1 \rightarrow (\ldots \rightarrow (\sigma_n \rightarrow \tau) \ldots)$. If b is a basic type then σ_i $(i = 1,\ldots,n)$ is called *input type* and b is called *output type* of $\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow b$. We denote by $O(\sigma)$ the output type of σ.

We assume a set of *variables* $\mathcal{V}_\tau$ and a set of *constants* $\mathcal{C}_\tau$ for each type $\tau \in \mathcal{T}_\mathcal{S}$, where $\mathcal{V}_\tau \cap \mathcal{V}_{\tau'} = \mathcal{C}_\tau \cap \mathcal{C}_{\tau'} = \emptyset$ if $\tau \neq \tau'$. The set of all variables is $\mathcal{V} = \cup_{\tau \in \mathcal{T}_\mathcal{S}} \mathcal{V}_\tau$, which is disjoint from the set of all constants $\mathcal{C} = \cup_{\tau \in \mathcal{T}_\mathcal{S}} \mathcal{C}_\tau$. $\mathcal{C}$ is called *signature*. If $f{:}\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow b \in \mathcal{C}$ and b is a basic type then $arity(f) = n$. Arbitrary variables are denoted by x, y, z ..., free variables by upper case letters F, G, X, ... and constants by a, c, d, e, ... *Higher-order variable* is a variable having a non-basic type.

The set of *untyped λ-terms* is generated from $\mathcal{C}$ and $\mathcal{V}$ according to the grammar: $\mathcal{T} := \mathcal{V} \mid \mathcal{C} \mid (\lambda\ \mathcal{V}.\ \mathcal{T}) \mid (\mathcal{T}\mathcal{T})$. Terms are denoted by l, r, s, t, ... The application of s to t is denoted by (st). We write $s(t_1,\ldots,t_n)$ for $(\ldots (st_1) \ldots t_n)$. We use $FV(t)$ for the set of *free variables* and $BV(t)$ for the set of *bound variables* of t. We may assume that bound variables are different from free ones. Further, we assume that for any variable $X{:}\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow b \in \mathcal{V}$, b is a basic type.

Definition 2.3.1 A *type judgment* stating that t is of type τ is written as $t : \tau$. The following rules inductively define the set of *simply typed λ-terms* $\mathcal{T}(\mathcal{C}, \mathcal{V})$.

- $x \in \mathcal{V}_\tau$ implies $x : \tau$.
- $c \in \mathcal{C}_\tau$ implies $c : \tau$.
- $s : \sigma \rightarrow \tau$ and $t : \sigma$ imply $(st) : \tau$.
- $x : \sigma$ and $s : \tau$ imply $(\lambda\, x.s) : \sigma \rightarrow \tau$.

In the rest of this paper, simply typed λ-terms are written as *λ-terms*. A λ-term is *ground* if it contains no free variables. $\mathcal{T}(\mathcal{C})$ denotes the set of ground λ-terms.

Definition 2.3.2 The *order of a type* $\varphi = \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow b$, where $b \in \mathcal{S}$, is defined as follows:

$$Ord(\varphi) = \begin{cases} 1 & \text{if } n = 0, \text{ i.e, } \varphi = b \in \mathcal{S} \\ 1 + k & \text{otherwise, where } k = max(Ord(b_1), \ldots, Ord(b_n)) \end{cases}$$

We say a symbol is of order n if it has a type of order n. A *term of order n* is restricted to constants of order $\leq n+1$ and variables of order $\leq n$.

A term of order 1 is called a *first-order term*. A term of order n is called a *higher-order term* if $n > 1$.

For instance, if a term $X(s_1, \ldots, s_n)$ is second order, then all subterms s_i must be first-order terms.

Substitutions are written as in $\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$ where λ-term t_i is assumed different from variable x_i and x_i and t_i have the same type ($i = 1, \ldots, n$). We use the letter θ for substitutions. Substitutions behave as endomorphisms defined on free variables. Letting $\theta = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$, $dom(\theta)$ denotes the set $\{x_1, \ldots, x_n\}$ and $range(\theta)$ denotes the set $\{t_1, \ldots, t_n\}$. A substitution θ is *ground* if $range(\theta) \subseteq \mathcal{T}(\mathcal{C})$.

We assume that the usual definition of *α-conversion* between λ-terms [16]. We write $s =_\alpha t$ if s and t are equivalent modulo α-conversion. In the following we consider that α-equivalent terms are identified. We write just $s = t$ when $s =_\alpha t$.

Definition 2.3.3 The *β-reduction* and *η-reduction* in λ-calculus are defined as follows:
$(\lambda\, x.s)(t) \rightarrow_\beta s\ \{x \leftarrow t\}$,
$\lambda\, x.s(x) \rightarrow_\eta s$ if $x \notin FV(s)$.

Since the simply typed λ-calculus is confluent and terminating with respect to β-reduction (η-reduction), every λ-term s has a β-normal form (η-normal form) which is denoted $s\downarrow_\beta$ ($s\downarrow_\eta$). Let s be in β-normal form. Then, s is of the form $\lambda\, x_1 \ldots x_n.\alpha(u_1, \ldots, u_m)$. The *$\eta$-expanded form* of s is defined by
$s\uparrow^\eta = \lambda\, x_1 \ldots x_{n+k}.\alpha(u_1\uparrow^\eta, \ldots, u_m\uparrow^\eta, \ldots, x_{n+k}\uparrow^\eta)$
where $s{:}\sigma_1 \rightarrow \ldots \rightarrow \sigma_{n+k} \rightarrow b$, b is basic type and $x_{n+1}, \ldots, x_{n+k} \notin FV(u_i)$ $(1 \leq i \leq m)$.

Given a λ-term s, we denote by $s\downarrow$ its unique *η-long β-normal form* (*η-long β-normal term*), defined as the β-normal form of its η-expanded form. We say shortly that s is *normalized* when s is in η-long β-normal form. $\mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$ denotes the set of normalized terms. $\mathcal{T}(\mathcal{C})\downarrow$ denotes the set of ground normalized terms.

Definition 2.3.4 A substitution $\theta{:}\mathcal{V} \rightarrow \mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$ is *normalized* if $range(\theta) \subseteq \mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$.

We suppose that for every basic type b there is a constant of type b not occurring in $\mathcal{C}$, denoted by $\Box$ (called *hole*). A *context* is a normalized term with occurrences of $\Box$. A context with only one occurrence of $\Box$ is denoted by $C[\ ]$.

Lemma 2.3.5 Normalized terms have one of the following two forms [27] : $(\lambda\ x.s)$ for some normalized term s, or $\alpha(s_1, \ldots, s_n)$ for some $\alpha \in \mathcal{C} \cup \mathcal{V}$ and normalized terms s_1, $\ldots$, s_n.

A *position* or *occurrence* in a normalized term can be viewed as a finite sequence of natural numbers [4]. $O(t)$ denotes the set of all positions of a normalized term t. $Ot(t)$ denotes the set of all *terminal positions* (positions of all leaves) of a normalized term t. The letter ϵ denotes the *root position*. The letters p, q, w, z stand for positions. We write $w \preceq z$ if w is a prefix of z.

The *subterm of* t *at position* p is denoted by $t|_p$ and we write $t \trianglerighteq t|_p$. If $t \neq t|_p$ then $t|_p$ is called the *proper subterm* of t, denoted by $t \triangleright t|_p$.

Lemma 2.3.6 Let $C[s]$ and t be normalized terms such that s and t have the same basic type. Then, $C[t]$ is normalized.

Two terms s and t are called *unifiable* if and only if there exists a substitution θ such that $s\theta = t\theta$. The term s *matches* the term t if and only if there exists a substitution θ such that $s\theta = t$. The problem to decide if a term s matches a term t and to compute the substitution θ is called *matching problem*.

2.4 Higher-Order Rewrite Systems

In this section, we introduce the notion of higher-order rewrite systems. We follow the definition of higher-order rewrite systems in Nipkow [38, 43]. Higher-order rewrite systems are rewrite systems on η-long β-normal forms in this thesis. Higher-order rewrite systems are generalizations of (first-order) term rewriting systems to terms with higher-order functions and bound variables. Since the unifiability of λ-terms is undecidable in general [15], we restrict to a certain subclass of λ-terms which behave like first-order terms with respect to unification.

Definition 2.4.1 A normalized term t is called a *pattern* if every free occurrence of a variable X is in a subterm $X(u_1,\ldots,u_n)$ of t, such that $u_1,\ldots,u_n$ are η-equivalent to a list of distinct bound variables.

We give the examples of pattern in the following.

Example 2.4.2 Examples of patterns are $\lambda\, x.c(x)$, F, $\lambda\, x.F(\lambda\, z.x(z))$ and $\lambda\, xy.F(x,y)$. Examples of non-patterns are $F(c)$, $\lambda\, x.F(x,x)$ and $\lambda\, x.G(H(x))$.

The following theorem about unification of patterns is showed by Miller [40].

Theorem 2.4.3 It is decidable whether two patterns are unifiable. If they are unifiable, the most general unifier can be computed.

A *rewrite rule* is a pair $l \to r$ such that l and r are normalized terms with the same basic type, l is not $\beta\eta$-equivalent to free variable, l is a pattern and $FV(l) \supseteq FV(r)$. A *higher-order rewrite system* (HRS) is a set of rewrite rules. The letter $\mathcal{R}$ denotes a higher-order rewrite system. Then, the restriction $FV(l) \supseteq FV(r)$ is preserved under substitutions, i.e., for any substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$, $FV(l) \supseteq FV(r)$ implies $FV(l\theta\downarrow) \supseteq FV(r\theta\downarrow)$ holds [38].

Given a HRS $\mathcal{R}$, a normalized term s is rewritten to a term t with respect to $\mathcal{R}$, written $s \to_{\mathcal{R}} t$, if $s = C[l\theta\downarrow]$ and $t = C[r\theta\downarrow]$ for some rewrite rule $l \to r \in \mathcal{R}$, context $C[\ \ ]$ and substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$. Note that t is normalized since s is so.

The relation $\to_{\mathcal{R}}$ is decidable if the matching problem is decidable for the left-hand sides of the rewrite rule in $\mathcal{R}$. Since theorem 2.4.3 and the matching problem is special case of the unification problem, the relation $\to_{\mathcal{R}}$ is decidable for any HRS $\mathcal{R}$.

Chapter 3

Orderings for Term Rewriting Systems

Huet and Lankford [18] showed that termination is undecidable for TRSs in general. Dauchet [8] showed that termination is undecidable even for one rule TRSs. However, there are several methods for proving termination of TRSs that are successful for particular cases. A well-known method for proving termination of TRSs is the notion of *simplification orderings* introduced by Dershowitz [9]. In this chapter, we introduce several orderings and explain how to use these orderings to prove termination of TRSs [9, 12, 13, 39, 60, 63].

3.1 Simplification Ordering

In this section, we introduce simplification orderings, homeomorphic embedding relations and Kruskal's tree theorem for proving termination of TRSs. Homeomorphic embedding relations and Kruskal's tree theorem guarantee the well-foundedness of simplification orderings. We consider some well-founded ordering $>$ on terms. If $s > t$ for any reduction $s \rightarrow_{\mathcal{R}} t$ then TRS $\mathcal{R}$ is terminating. Intuitively, the notion of simplification ordering guarantees the well-foundedness of partial ordering on terms.

The *root symbol* of a term is defined by $root(\alpha(s_1,\ldots,s_m)) = \alpha$ $(m \geq 0)$ if $\alpha \in \mathcal{F} \cup \mathcal{V}$. A term s_i is called an *immediate subterm* of term $s = \alpha(s_1, \ldots, s_i, \ldots, s_n)$. Then, $st(s) = \langle s_1,\ldots,s_i,\ldots,s_n\rangle$ and $st^m(s) = \{s_1,\ldots,s_i,\ldots,s_n\}$ denote the sequence and the multiset of immediate subterms of s, respectively.

The *status* is a function $status{:}\mathcal{F} \rightarrow \{mult, left, right\}$. Thus every function symbol has one of the following statuses: $mult$ (the arguments will be compared as multiset), $left$ (lexicographical comparison from left to right), $right$ (lexicographical comparison from right to left).

Definition 3.1.1 A partial ordering $>$ is a *simplification order* on $\mathcal{T}(\mathcal{F})$ if it possesses the following properties:

(1) $s > t$ implies $f(u_1,\ldots,s,\ldots,u_n) > f(u_1,\ldots,t,\ldots,u_n)$ for $f \in \mathcal{F}$. (the *replacement property*),

(2) $s > s_i$ for any $s_i \in st^m(s)$ (the *subterm property*).

If signature $\mathcal{F}$ is finite, then any simplification order on $\mathcal{T}(\mathcal{F})$ are well-founded, by Kruskal's tree theorem presented later. If signature $\mathcal{F}$ is infinite, the subterm property is not enough to guarantee well-foundedness.

Example 3.1.2 We consider the signature $\mathcal{F} = \{a_i \mid i \geq 0\}$, where each symbol a_i is a constant. Then the ordering $>$ defined by $a_i > a_j$, for all $i < j$. This ordering is a simplification order and yet is not well-founded.

Note that in the case that the signature $\mathcal{F}$ is finite, definition 3.1.1 and the following definition coincide, since any partial ordering on $\mathcal{F}$ is a well-partial ordering.

Definition 3.1.3 Let $\succ$ be a well-partial ordering on $\mathcal{F}$. A partial ordering $>$ is a *simplification ordering* on $\mathcal{T}(\mathcal{F})$ if it possesses the following three properties:

(1) $s > t$ implies $f(u_1,\ldots,s,\ldots,u_n) > f(u_1,\ldots,t,\ldots,u_n)$ for $f \in \mathcal{F}$. (the *replacement property*),

(2) $s > s_i$ for any $s_i \in st^m(s)$ (the *subterm property*),

(3) $f(u_1,\ldots,u_n) > g(u_{i1},\ldots,u_{im})$ if $f, g \in \mathcal{F}$, $f \succ g$, $1 \leq i1 < \ldots < im \leq n$, $arity(f) = n$ and $arity(g) = m$.

Definition 3.1.4 Let $\succ$ be a partial ordering on $\mathcal{F}$. The *homeomorphic embedding relation* $\rhd_{emb}$ on $\mathcal{T}(\mathcal{F})$ is defined inductively as follows:
$s = f(s_1,\ldots,s_n) \rhd_{emb} g(t_1,\ldots,t_m) = t$ ($arity(f) = n$ and $arity(g) = m$)
if and only if either one of the following conditions holds:

(1) $f \succ g$ and there exist indexes $j1,\ldots,jm$ such that $1 \leq j1 < j2 < \ldots < jm \leq n$ and $s_{ji} \trianglerighteq_{emb} t_i$ $(i = 1,\ldots,m)$.

(2) $s_j \trianglerighteq_{emb} t$ for some j.

The *embedding relation* $>_{emb}$ on $\mathcal{T}(\mathcal{F})$ is a homeomorphic embedding relation without condition (1).

Lemma 3.1.5 Let $\succ$ be a well-partial ordering on $\mathcal{F}$ and $>$ be a simplification ordering $>$ on $\mathcal{T}(\mathcal{F})$. Then, $\rhd_{emb} \subseteq >$ holds.

The following Kruskal's tree theorem clarifies the relation between termination, homeomorphic embedding relation and simplification ordering.

Theorem 3.1.6 (Kruskal's Tree Theorem, [34]) If $\succ$ is a well-partial ordering on $\mathcal{F}$ then $\rhd_{emb}$ is a well-partial ordering on $\mathcal{T}(\mathcal{F})$.

For a proof of this result we refer to [39]. A proof of the version for well-quasi-ordering can be found in [14]. The following theorem is obtained by lemma 3.1.5 and theorem 3.1.6.

Theorem 3.1.7 Let $\succ$ be a well-partial ordering on $\mathcal{F}$ and $>$ be a simplification ordering on $\mathcal{T}(\mathcal{F})$. Then, $\langle \mathcal{T}(\mathcal{F}), > \rangle$ is a well-partially ordered set.

Corollary 3.1.8 Let $\succ$ be a well-partial ordering on $\mathcal{F}$ and $>$ be a simplification ordering on $\mathcal{T}(\mathcal{F})$. Then, $>$ is well-founded.

Note that we assume $\mathcal{T}(\mathcal{F})$ is a nonempty set in the rest of this chapter. The following theorem gurantees the termination property for TRSs.

Theorem 3.1.9 Let $\mathcal{R}$ be a TRS on $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Let $\succ$ be a well-founded ordering on $\mathcal{F}$ and $>$ a simplification ordering on $\mathcal{T}(\mathcal{F})$ such that $l\theta > r\theta$ for any ground substitution $\theta{:}\mathcal{V} \rightarrow \mathcal{T}(\mathcal{F})$ and any rewrite rule $l \rightarrow r$ in $\mathcal{R}$. Then, $\mathcal{R}$ is terminating on $\mathcal{T}(\mathcal{F}, \mathcal{V})$.

3.2 Recursive Path Orderig and Decomposition Ordering

In this section, we introduce the recursive path ordering (RPO) and the improved recursive decomposition ordering (IRD). Dershowitz [9] defined the RPO and Rusinowitch [59] defined the IRD. Steinbach [60, 63] revisited these orderings. In particular, the IRD were given a simplified version. We mainly follow the notations of Steinbach. The RPO is known as the most useful simplification ordering. It is known that the IRD is one of the most powerful simplification orderings [59, 60, 63]. The main difference between RPO and IRD is method of comparing two terms. The comparison of RPO is top to bottom sequential method and depends on the root symbols in terms. The comparison of IRD is parallel method and depends on all subterms along the path from root to leaf in terms. First, we introduce the RPO as follows.

Definition 3.2.1 (RPO) Let $>_{\mathcal{F}}$ be a partial ordering on $\mathcal{F}$. Let s and t be terms. The *recursive path ordering* (RPO) $>_{RPO}$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is defined as follows [9, 12, 63]:

$s >_{RPO} t$ if and only if

(1) $s_i \geq_{RPO} t$ for some $s_i \in st^m(s)$, or

(2) $root(s) >_{\mathcal{F}} root(t)$ and $s >_{RPO} t_i$ for all $t_i \in st^m(t)$, or

(3) $root(s) = root(t)$, $status(root(s)) = mult$ and $st^m(s) >_{RPO}^{mult} st^m(t)$, or

(4) $root(s) = root(t)$, $status(root(s)) \neq mult$, $st(s) >_{RPO}^{status(root(s))} st(t)$ and $s >_{RPO} t_i$ for all $t_i \in st^m(t)$.

where $>_{RPO}^{mult}$ and $>_{RPO}^{status(root(s))}$ are extensions of $>_{RPO}$ associated with the statuses $mult$ and $status(root(s))$, respectively.

We show that RPO can be used for proving termination of TRS in the following.

Theorem 3.2.2 If $>_{\mathcal{F}}$ is a well-partial ordering on $\mathcal{F}$ then the RPO is a simplification ordering on $\mathcal{T}(\mathcal{F})$.

Lemma 3.2.3 The RPO is stable under ground substitutions.

The following theorem show that termination of TRS can be proved using the RPO. It is obtained by the above two results.

Theorem 3.2.4 Let $\mathcal{R}$ be a TRS on terms. Let $>_{\mathcal{F}}$ be a well-founded ordering on $\mathcal{F}$. If for any rewrite rule $l \to r$ in $\mathcal{R}$ we have $l >_{RPO} r$, then $\mathcal{R}$ is terminating.

We give the termination proof of TRS using RPO.

Example 3.2.5 We consider the following TRS $\mathcal{R}$.

$$\mathcal{R} = \{\ x * (y + z) \to (x * y) + (x * z)$$

Let $l = x * (y + z)$, $r = (x * y) + (x * z)$ and $* >_{\mathcal{F}} +$.
We show $l >_{RPO} r$ as follows.

- In order to show $l >_{RPO} r$, we have to show that $l >_{RPO} x * y$ and $l >_{RPO} x * z$ since $* >_{\mathcal{F}} +$ and definition of RPO.
- $l >_{RPO} x*y$ and $l >_{RPO} x*z$ hold since $\{x, y+z\} >_{RPO}^{mult} \{x, y\}$ and $\{x, y+z\} >_{RPO}^{mult} \{x, z\}$.

Hence, $l >_{RPO} r$ holds. Therefore, $\mathcal{R}$ is terminating by theorem 3.2.4.

Next, we introduce the IRD as follows. IRD was defined by Rusinowitch [59]. Further Steinbach [60, 61, 62, 63] gave a simple definition of IRD. We follow the Steinbach's simplified version of IRD in this thesis.

Definition 3.2.6 Path-decomposition and decomposition [61, 62] are defined as follows:

- For $u \in O(t)$, the *path-decomposition* $dec_u(t)$ is defined as follows.

$$\begin{cases} dec_\epsilon(t) = \{t\} \\ dec_{i.v}(\alpha(t_1, \cdots, t_n)) = \{\alpha(t_1, \cdots, t_n)\} \cup dec_v(t_i) \end{cases}$$

  Note that $i.v \in Ot(\alpha(t_1,\cdots,t_n))$ implies $v \in Ot(t_i)$.

- We also define the *decomposition* $dec\ (\{t_1,\cdots,t_n\}) = \{dec_u(t_i) \mid i \in \{1,\cdots,n\},\ u \in Ot(t_i)\}$.

- For the path-decomposition $dec_u(t)$, $sub(dec_u(t),s) = \{s' \in dec_u(t) \mid s \rhd s'\}$.

- The notation

$$\begin{array}{l} -s >_1 t \\ -s' >_2 t' \end{array}$$

  means $s >_1 t$ or $(s = t$ and $s' >_2 t')$.

We give the example of path-decomposition and decomposition for some term.

Example 3.2.7 Let $t = (x * y) + (x * z)$. Then, we have the path-decomposition and decomposition for term t as follows.

- $dec_{21}(t) = \{t\} \cup dec_1(x * z) = \{t, x * z\} \cup dec_\epsilon(x) = \{t, x * z, x\}$.
- $dec(\{t\}) = \{dec_{11}(t), dec_{12}(t), dec_{21}(t), dec_{22}(t)\} = \{\{t, x * y, x\}, \{t, x * y, y\}, \{t, x * z, x\}, \{t, x * z, z\}\}$.
- $sub(dec_{21}(t), x) = \emptyset$.
- $sub(dec_{21}(t), t) = \{x * z, x\}$.

Definition 3.2.8 (IRD) Let $>_\mathcal{F}$ be a partial ordering on $\mathcal{F}$. Let s and t be terms. The *improved recursive decomposition ordering* (IRD) on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is defined as follows [59, 60, 61, 62, 63]:

$s >_{IRD} t$ if and only if $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$ where $\gg\gg_{EL}$ is the multiset extension of $\gg_{EL}$.

$dec_p(u) \ni u' >_{EL} v' \in dec_q(v)$ is defined by the following (1), (2) and (3).

(1) $root(u') >_\mathcal{F} root(v')$, or

(2) $root(u') = root(v')$, $status(root(u')) = mult$, and

- $sub(dec_p(u), u') \gg_{EL} sub(dec_q(v), v')$
- $dec(st^m(u')) \gg\gg_{EL} dec(st^m(v'))$, or

(3) $root(u') = root(v')$, $status(root(u')) \neq mult$, $st(u') >_{IRD}^{status(root(u'))} st(v')$ and $\{u'\} \gg_{IRD} st^m(v')$.

where $>_{IRD}^{status(root(u'))}$ is the extension of $>_{IRD}$ associated with the status $status(root(u'))$.

We show that IRD can be used for proving termination of TRS in the following.

Theorem 3.2.9 If $>_{\mathcal{F}}$ is a well-partial ordering on $\mathcal{F}$ then the IRD is a simplification ordering on $\mathcal{T}(\mathcal{F})$.

Lemma 3.2.10 The IRD is stable under ground substitutions.

The following theorem can be obtained directly from the above two results.

Theorem 3.2.11 Let $\mathcal{R}$ be a TRS on terms. Let $>_{\mathcal{F}}$ be a well-founded ordering on $\mathcal{F}$. If for any rewrite rule $l \to r$ in $\mathcal{R}$ we have $l >_{IRD} r$, then $\mathcal{R}$ is terminating.

We give the termination proof of TRS using IRD.

Example 3.2.12 We consider the following TRS $\mathcal{R}$.

$$\mathcal{R} = \{\ x * (y + z) \to (x * y) + (x * z)$$

Let $l = x * (y + z)$, $r = (x * y) + (x * z)$ and $* >_{\mathcal{F}} +$. Then, we show $l >_{IRD} r$ as follows.

- $dec(\{l\}) = \{dec_1(l), dec_{21}(l), dec_{22}(l)\} = \{\{l, x\}, \{l, y + z, y\}, \{l, y + z, z\}\}$.
- $dec(\{r\}) = \{dec_{11}(r), dec_{12}(r), dec_{21}(r), dec_{22}(r)\} = \{\{r, x * y, x\}, \{r, x * y, y\}, \{r, x * z, x\}, \{r, x * z, z\}\}$.

(1) $dec_1(l) = \{l, x\} \gg_{EL} \{r, x * y, x\} = dec_{11}(r)$.
Because $sub(dec_1(l), l) = sub(dec_{11}(r), x*y)$ and $dec(args(l)) = \{\{x\}, \{y+z, y\}, \{y+z, z\}\} \gg\gg_{EL} \{\{x\}, \{y\}\} = dec(args(x * y))$ hold.

(2) By the same argument of (1), $dec_1(l) = \{l, x\} \gg_{EL} \{r, x * z, x\} = dec_{21}(r)$ holds.

(3) By $* >_{\mathcal{F}} +$, $dec_{21}(l) = \{l, y + z, y\} \gg_{EL} \{r, x * y, y\} = dec_{12}(r)$ holds.

(4) By $* >_{\mathcal{F}} +$, $dec_{22}(l) = \{l, y + z, z\} \gg_{EL} \{r, x * z, z\} = dec_{22}(r)$ holds.

Since above cases (1), (2), (3) and (4), $dec(\{l\}) \gg\gg_{EL} dec(\{r\})$ holds. Hence, $l >_{IRD} r$ holds. Therefore, $\mathcal{R}$ is terminating by theorem 3.2.11.

The next theorem shows that the IRD is more powerful than RPO in (first-order) TRS [59, 63].

Theorem 3.2.13 ([59, 63]) Let s and t be terms. Then $s >_{RPO} t$ implies $s >_{IRD} t$.

Chapter 4

Improved Recursive Decomposition Ordering for Higher-Order Rewrite Systems

4.1 Introduction

Simplification orderings are representatives of syntactic methods for proving termination of TRSs [9, 12, 60, 63]. Many simplification orderings, for instance, the recursive path ordering (RPO) [9, 12, 63], the improved recursive decomposition ordering (IRD) [59, 60, 61, 62, 63] and so on, have been defined on TRSs. The IRD is among the most powerful simplification orderings [59, 63]. We give the properties of RPO and IRD as follows. The method of comparing two terms with respect to the RPO depends on the root symbols. The relation between these symbols with respect to the precedence is responsible for decreasing one or both of the terms in the recursive definition of the RPO. If one of the terms gets empty then the other one is greater. The IRD does not use a strict structure of terms as the RPO. More precisely, this property of the IRD can lead to a term to be represented by the multiset of all of its subterms and the comparison of two terms is done by comparing the corresponding multisets.

In case of HRSs, the termination property is also important. Loría-Sáenz and Steinbach [36] first extended RPO for proving termination of HRSs. Their extension method is based on an interpretation that maps normalized terms to first-order terms. Lysne and Piris [37] also extended RPO for proving termination of HRSs by introducing the notions of termination functions, critical positions and dominations. We gave an extension of IRD to HRSs in [19, 20, 21] according to Lysne and Piris's method. Jouannaud and Rubio [27] gave a simple definition of RPO for HRSs by introducing an ordering on type structure. van de Pol [55] and Kahrs [29] extended semantic method to HRSs.

In this chapter we propose IRD for HRSs, called the *higher-order improved recursive decomposition ordering* (HIRD). Our method is inspired by Jouannaud and Rubio's idea for RPO [27] and particular properties of IRD. Further we show that our ordering is more powerful than their ordering.

In section 4.2 we define the *typed improved recursive decomposition ordering* (TIRD) by introducing a new concept of *pseudo-terminal positions*. Section 4.3 presents the definition of the *higher-order improved recursive decomposition ordering* (HIRD) and shows that HIRD is a powerful tool for proving termination of HRSs. Finally, we compare HIRD

with Jouannad and Rubio's ordering in section 4.4.

4.2 Typed Improved Recursive Decomposition Ordering

The higher-order improved recursive decomposition ordering on $\mathcal{T}(\mathcal{C}, \mathcal{V})\downarrow$ is defined by two comparison stages. In the first stage, higher-order terms s and t are interpreted as algebraic terms s' and t' having the extended signature $\lambda\mathcal{C}$. In the second stage, s' and t' are compared by a typed improved recursive decomposition ordering on algebraic terms.

If we simply extended the improved recursive decomposition ordering (IRD) [59, 60, 61, 62, 63] to handle higher-order by using Jouannaud and Rubio's idea of type structure [27] for the second stage, the resulting ordering would not be stable under ground normalized substitutions. For instance, let $a{:}\sigma$, $b{:}\sigma \in \mathcal{C}$, $Y{:}\sigma \to \sigma \in \mathcal{V}$ and $a >_{\mathcal{C}} b$. Let $s = Y(a)$, $t = a$. Since the improved recursive decomposition ordering $>_{IRD}$ is a simplification ordering [60, 61], it has a subterm property. Hence $s >_{IRD} t$ holds. However, the substitution $\theta = \{Y \leftarrow \lambda\ x.b\}$ does not preserve this ordering as $s\ \theta\downarrow = b <_{IRD} a = t\ \theta\downarrow$. In order to avoid this problem, we introduce the notion of *pseudo-terminal positions*.

First, we define the typed improved recursive decomposition ordering on algebraic terms. It requires a partial ordering on $\mathcal{T}_{\mathcal{S}}$ in addition to a partial ordering on $\mathcal{C}$.

Note that we assume there are only finitely many symbols of a given output type in the signature $\mathcal{C}$ in the rest of this chapter.

Definition 4.2.1 *Algebraic terms* are normalized terms with no λ-abstractions. An algebraic term is *ground* if it contains no variables. The type of algebraic term t is denoted by $type(t)$.

Definition 4.2.2 The *root symbol* of an algebraic term is defined by $root(\alpha(s_1,\ldots,s_m)) = \alpha$ if $\alpha \in \mathcal{C} \cup \mathcal{V}$. The size $\mid t \mid$ of algebraic term t is defined as the number of symbols occurring in t. An algebraic term s_i is called an *immediate subterm* of algebraic term $s = \alpha(s_1, \ldots, s_i, \ldots, s_n)$. Then, $st(s) = \langle\ s_1,\ldots,s_i,\ldots,s_n\ \rangle$ and $st^m(s) = \{s_1,\ldots,s_i,\ldots,s_n\}$ denote the sequence and the multiset of immediate subterms of s, respectively.

In the following, we use a partial ordering $>_{\mathcal{T}_{\mathcal{S}}}$ on $\mathcal{T}_{\mathcal{S}}$ instead of a quasi-ordering $\gtrsim_{\mathcal{T}_{\mathcal{S}}}$ in [27, 28]. The following definitions are obtained by definitions in [28] replacing a quasi-ordering $\gtrsim_{\mathcal{T}_{\mathcal{S}}}$ with a partial ordering $>_{\mathcal{T}_{\mathcal{S}}}$.

Definition 4.2.3 Let $>_{\mathcal{T}_{\mathcal{S}}}$ be a partial ordering on $\mathcal{T}_{\mathcal{S}}$. An algebraic term $s : \sigma$ is *type compatible* if for any proper subterm $t : \tau$ of s then $\sigma \geq_{\mathcal{T}_{\mathcal{S}}} \tau$ and $t : \tau$ is type compatible. A normalized term $s : \sigma$ is *type compatible* if for any proper subterm $t : \tau$ of s then $\sigma \geq_{\mathcal{T}_{\mathcal{S}}} \tau$ and $t : \tau$ is type compatible.

Note that we assume algebraic terms and normalized terms are always type compatible in the rest of this chapter.

Definition 4.2.4 A type σ is *compatible with* $>_{\mathcal{T}_\mathcal{S}}$ *on* $\mathcal{T}_\mathcal{S}$ if it is a basic type or it is of the form $\sigma_1 \to \ldots \to \sigma_n \to \tau$, where τ is a basic type, and $\tau \geq_{\mathcal{T}_\mathcal{S}} \sigma_i$ and σ_i is compatible with $>_{\mathcal{T}_\mathcal{S}}$ for all $i \in \{1, \ldots, n\}$.

Definition 4.2.5 A partial ordering $>_{\mathcal{T}_\mathcal{S}}$ on $\mathcal{T}_\mathcal{S}$ is *consistent with the type structure* if the following conditions hold:

(1) $\forall f : \sigma \in \mathcal{C}$, type σ is compatible with $>_{\mathcal{T}_\mathcal{S}}$.

(2) $\sigma \to \tau \geq_{\mathcal{T}_\mathcal{S}} \tau$.

Lemma 4.2.6 ([28]) Let $>_{\mathcal{T}_\mathcal{S}}$ be a partial ordering on $\mathcal{T}_\mathcal{S}$ consistent with the type structure and let $s : \sigma$ be a normalized term. Then, $s : \sigma$ is a type compatible term if the following conditions hold.

(1) For every $x : \tau \in FV(s)$ we have that τ is a compatible type.

(2) If $s : \sigma$ is of the form $\lambda\, x_1 \ldots x_n.u{:}\sigma_1 \to \ldots \to \sigma_n \to b$ for some basic type b, then σ_i is a compatible type for all $x_i \in FV(u)$.

From here on we assume $>_{\mathcal{T}_\mathcal{S}}$ is well-founded on $\mathcal{T}_\mathcal{S}$ in this chapter.

Let $>_\mathcal{S}$ be a partial ordering on $\mathcal{S}$ and the constructor $\to$. The RPO based on a precedence $>_\mathcal{S}$ on $\mathcal{S} \cup \{\to\}$ is consistent with the type structure [28]. Hence we can take the RPO as a partial ordering $>_{\mathcal{T}_\mathcal{S}}$ on $\mathcal{T}_\mathcal{S}$ in the rest of this chapter.

Example 4.2.7 Let $List$ and Nat be basic types and let $List >_\mathcal{S} Nat$ and $List >_\mathcal{S} \to$. Then $List >_{\mathcal{T}_\mathcal{S}} Nat \to Nat$ holds, since the ordering $>_{\mathcal{T}_\mathcal{S}}$ is the RPO on $\mathcal{T}_\mathcal{S}$ and we consider the type $Nat \to Nat$ as the form $\to(Nat, Nat)$.

A partial ordering $>_\mathcal{C}$ on $\mathcal{C}$ compares symbols of the same output type only and must be well-founded. A variable $x{:}\sigma$ is considered as a constant symbol comparable only itself.

Definition 4.2.8 The *extended signature* $\lambda\mathcal{C}$ is defined by the following:

- $\widetilde{\mathcal{V}} = \{X_{O(\sigma)} \mid X : \sigma \in \mathcal{V}\}$.
- $\widetilde{\mathcal{C}} = \{\alpha_{O(\sigma)} \mid \alpha : \sigma \in \mathcal{C}\}$.

- $\lambda\mathcal{C} = \widetilde{\mathcal{C}} \cup \bigcup_{\sigma \in \mathcal{T}_S} \{c_{O(\sigma)}\} \cup \bigcup_{\sigma,\tau \in \mathcal{T}_S} \{\lambda_{\sigma \to \tau}\}$.

Note that we have a constant symbol $c_{O(\sigma)}$ for each type σ and a unary symbol $\lambda_{\sigma \to \tau}$ for each type $\sigma \to \tau$. Hence if $\mathcal{C}$ has a finite set of constant symbols for each type, then $\lambda\mathcal{C}$ has a finite set of constant symbols for each type. The precedence $>_{\mathcal{C}}$ on $\mathcal{C}$ will be extended to the precedence $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$. The symbol $\lambda_{\sigma \to \tau}$ may have one of any status. The compatible property for the $\lambda_{\sigma \to \tau}$ symbol requires the $\sigma \to \tau \geq_{\mathcal{T}_S} \tau$.

In the following, we consider algebraic terms on $\lambda\mathcal{C}$.

Definition 4.2.9 The set of algebraic terms on $\lambda\mathcal{C}$ is denoted by $\mathcal{T}(\lambda\mathcal{C}, \widetilde{\mathcal{V}})$ and the set of ground algebraic terms on $\lambda\mathcal{C}$ is denoted by $\mathcal{T}(\lambda\mathcal{C})$.

We generalize the definition of a set of terminal positions. We define the notion of pseudo-terminal positions as follows.

Definition 4.2.10 Let t be an algebraic term. A position $q \in O(t)$ is a *pseudo-terminal position* if it satisfies both the following conditions.

(1) $q \in Ot(t) \vee root(t\mid_q) \in \widetilde{\mathcal{V}}$.

(2) $\forall q' \prec q, [root(t\mid_{q'}) \notin \widetilde{\mathcal{V}}]$.

We write $q \in Op(t)$ if q is a pseudo-terminal position of t.

Definition 4.2.11 Path-decompositions and decompositions [61, 62] can be naturally extended to higher-order terms by using pseudo-terminal positions.

- For $u \in Op(t)$, the *path-decomposition* $dec_u(t)$ is defined as follows:

$$\begin{cases} dec_\epsilon(t) = \{t\} \\ dec_{i.v}(\alpha(t_1, \cdots, t_n)) = \{\alpha(t_1, \cdots, t_n)\} \cup dec_v(t_i) \end{cases}$$

  Note that $i.v \in Op(\alpha(t_1,\cdots,t_n))$ implies $v \in Op(t_i)$.

- The *decomposition* is defined as follows:
  $dec(\{t_1,\cdots,t_n\}) = \{dec_u(t_i) |\ i{\in}\{1,\cdots,n\}, u{\in}Op(t_i)\}$.

- For the path-decomposition $dec_u(t)$, $sub(dec_u(t),s)$ is defined as follows:
  $sub(dec_u(t),s) = \{s' {\in} dec_u(t) | s \rhd s'\}$.

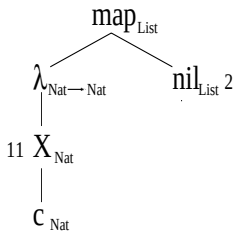

Figure 4.1: $map_{List}(\lambda_{Nat \to Nat}(X_{Nat}(c_{Nat})), nil_{List})$.

Example 4.2.12 Let $List$ and Nat be basic types. We consider the term $s = map_{List}$ $(\lambda_{Nat \to Nat}\ (X_{Nat}\ (c_{Nat}))$, $nil_{List})$ where $X_{Nat} \in \widetilde{\mathcal{V}}$. Figure 4.1 denotes the tree structure of s. We have the set of pseudo-terminal positions $Op(s) = \{11,\ 2\}$. Then $dec(\{s\}) = \{dec_{11}(s), dec_2(s)\}$ where $dec_{11}(s) = \{s,\ \lambda_{Nat \to Nat}(X_{Nat}(c_{Nat})),\ X_{Nat}(c_{Nat})\}$ and $dec_2(s) = \{s,\ nil_{List}\}$. Further $sub(dec_{11}(s),\ \lambda_{Nat \to Nat}(X_{Nat}(c_{Nat}))) = \{X_{Nat}(c_{Nat})\}$ and $sub(dec_2(s),\ s) = \{nil_{List}\}$.

Next, we define the typed improved recursive decomposition ordering (TIRD). This ordering is based on the improved recursive decomposition ordering (IRD) and extended by introducing the notion of pseudo-terminal positions and ordering on types.

Definition 4.2.13 (TIRD) Let $>_{\lambda\mathcal{C}}$ be a partial ordering on $\lambda\mathcal{C}$. Let s and t be type compatible algebraic terms. The *typed improved recursive decomposition ordering* (TIRD) on type compatible algebraic terms is defined as follows:

$s >_{TIRD} t$ if and only if $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$ where $\gg\gg_{EL}$ is the multiset extension of $\gg_{EL}$.

$dec_p(u) \ni u' >_{EL} v' \in dec_q(v)$ is defined by the following (1) and (2).

(1) $type(u') >_{\mathcal{T}_S} type(v')$, or

(2) $type(u') = type(v')$, and

  a) $root(u') >_{\lambda\mathcal{C}} root(v')$, or

  b) $root(u') = root(v')$, $status(root(u')) = mult$, and

    - $sub(dec_p(u), u') \gg_{EL} sub(dec_q(v), v')$
    - $dec(st^m(u')) \gg\gg_{EL} dec(st^m(v'))$, or

  c) $root(u') = root(v')$, $status(root(u')) \neq mult$, $st(u') >_{TIRD}^{status(root(u'))} st(v')$ and $\{u'\} \gg_{TIRD} st^m(v')$.

where $>_{TIRD}^{status(root(u'))}$ is the extension of $>_{TIRD}$ associated with the status $status$ $(root\ (u'))$.

Example 4.2.14 We compare the following algebraic terms s and t with respect to TIRD. Let $List$ and Nat be basic types.

- $s = map_{List}(\lambda_{Nat \to Nat}(X_{Nat}(c_{Nat})),cons_{List}(N_{Nat},L_{Nat}))$.
- $t = cons_{List}(X_{Nat}(N_{Nat}),map_{List}(\lambda_{Nat \to Nat}(X_{Nat}(c_{Nat})),L_{List}))$.

Then we have $dec(\{s\})$ and $dec(\{t\})$ as follows:

- $dec(\{s\}) = \{dec_{11}(s),\ dec_{21}(s),\ dec_{22}(s)\}$.
- $dec(\{t\}) = \{dec_{1}(t),\ dec_{211}(t),\ dec_{22}(t)\}$.

Given the following precedences and status: $map_{List} >_{\lambda\mathcal{C}} cons_{List}$, $List >_{\mathcal{S}} Nat$, $List >_{\mathcal{S}} \to$ and $status(map_{List}) = mult$.

(1) We consider $dec_{21}(s)$ and $dec_{1}(t)$ as follows:

- $dec_{21}(s) = \{s,cons_{List}(N_{Nat},L_{List}),N_{Nat}\}$.
- $dec_{1}(t) = \{t,X_{Nat}(N_{Nat})\}$.

$s >_{EL} t$ by $root(s) >_{\lambda\mathcal{C}} root(t)$ and $cons_{List}(N_{Nat},L_{List}) >_{EL} X_{Nat}(N_{Nat})$ by $List >_{\mathcal{T}_{\mathcal{S}}} Nat$. Hence, $dec_{21}(s) \gg_{EL} dec_{1}(t)$ holds.

(2) Next, we compare $dec_{21}(s)$ and $dec_{211}(t)$.

- $dec_{21}(s) = \{s,cons_{List}(N_{Nat},L_{List}),N_{Nat}\}$.
- $dec_{211}(t) = \{t,\ map_{List}\ (\lambda_{Nat \to Nat}(X_{Nat}\ (c_{Nat})),L_{List}),\ \lambda_{Nat \to Nat}\ (X_{Nat}\ (c_{Nat})),\ X_{Nat}\ (c_{Nat})\}$.

We have $s >_{EL} \lambda_{Nat \to Nat}(X_{Nat}(c_{Nat}))$ by $List >_{\mathcal{T}_{\mathcal{S}}} Nat \to Nat$. Let $t' = map_{List}\ (\lambda_{Nat \to Nat}(X_{Nat}\ (c_{Nat})),L_{List})$. We also have $s >_{EL} t'$ since $root(s) = root(t')$, $status(root(s)) = mult$ and $sub\ (dec_{21}(s),\ s) = \{cons_{List}(N_{Nat},L_{List}),\ N_{Nat}\} \gg_{EL} \{\lambda_{Nat \to Nat}(X_{Nat}\ (c_{Nat})),\ X_{Nat}\ (c_{Nat})\} = sub\ (dec_{211}(t),\ t')$. Hence, $dec_{21}(s) \gg_{EL} dec_{211}(t)$ holds.

(3) We have $dec_{22}(s)$ and $dec_{22}(t)$ as follows:

- $dec_{22}\ (s) = \{s,\ cons_{List}(N_{Nat},L_{List}),\ L_{List}\}$
- $dec_{22}(t) = \{t,\ map_{List}(\lambda_{Nat \to Nat}(X_{Nat}(c_{Nat})),L_{List}),\ L_{List}\}$.

Then $dec22(s) \gg_{EL} dec_{22}(t)$ holds.

Therefore, we have $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$, i.e., $s >_{TIRD} t$.

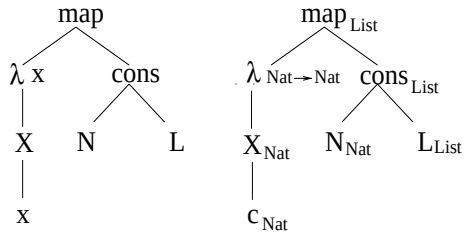

Figure 4.2: $map(\lambda x.X(x),cons(N,L))$ and $\|map(\lambda x.X(x),cons(N,L))\|_{\{x\}}$.

4.3 Higher-Order Improved Recursive Decomposition Ordering

The higher-order improved recursive decomposition ordering on normalized terms is defined as TIRD comparing algebraic terms interpreted from given normalized terms. Thus, we define an interpretation function that maps normalized terms to algebraic terms as follows.

Definition 4.3.1 Let V be a set of variables. The *interpretation function* $\|\ \ \|_V$ from normalized terms on $\mathcal{C} \cup \bigcup_{\sigma \in \mathcal{T}_\mathcal{S}} \{c_{O(\sigma)}\}$ to algebraic terms on $\lambda\mathcal{C}$ is defined by:

- $\|(\lambda x.s){:}\sigma{\rightarrow}\tau\|_V = \lambda_{\sigma\rightarrow\tau}(\|s\|_V)$.
- $\|\alpha(s_1,\cdots,s_n){:}\sigma\|_V = \alpha_\sigma(\|s_1\|_V,\cdots,\|s_n\|_V)$ if $\alpha \in \mathcal{C}$.
- $\|X(s_1,\cdots,s_n){:}\sigma\|_V = X_\sigma(\|s_1\|_V,\cdots,\|s_n\|_V)$ if $X{\in}\mathcal{V}\backslash V$.
- $\|x(s_1,\cdots,s_n){:}\sigma\|_V = c_\sigma(\|s_1\|_V,\cdots,\|s_n\|_V)$ if $x \in V \cup \bigcup_{\sigma\in\mathcal{T}_\mathcal{S}}\{c_{O(\sigma)}\}$.

Let s be a normalized term and $V \supseteq BV(s)$. We have $\|s\|_V|_p = \|s|_p\|_V$ for any $p \in O(s)$. Whenever we write $\|s\|_V$ for normalized terms s, we assume that $V \supseteq BV(s)$.

Example 4.3.2 Let $\mathcal{S} = \{Nat,\ List\}$, $\mathcal{C} = \{nil{:}List,\ cons{:}Nat \rightarrow List \rightarrow List,$ $map{:}(Nat \rightarrow Nat) \rightarrow List \rightarrow List\}$ and $\{X{:}Nat \rightarrow Nat,\ N{:}Nat,\ L{:}List\} \subseteq \mathcal{V}$.

Let $V = \{x\}$. Let $s = map(\lambda x.X(x),cons(N,L))$. The following examples explain the interpretation function.

- $\|\ s\ \|_V = map_{List}(\lambda_{Nat\rightarrow Nat}(X_{Nat}(c_{Nat})),cons_{List}(N_{Nat},L_{List}))$. (See figure 4.2).
- $\|s|_{11}\|_V = \|X(x)\|_V = X_{Nat}(c_{Nat})$.
- $\|s\|_V|_{11} = map_{List}(\lambda_{Nat\rightarrow Nat}(X_{Nat}(c_{Nat})),cons_{List}(N_{Nat},L_{List}))\ |_{11} = X_{Nat}(c_{Nat})$.

Next, we define the higher-order improved recursive decomposition ordering by using definitions 4.2.13 and 4.3.1.

Definition 4.3.3 (HIRD) Let s and t be type compatible normalized terms. Then the *higher-order improved recursive decomposition ordering* (HIRD) on type compatible normalized terms is defined as follows: $s >_{HIRD} t$ if and only if $\|s\|_V >_{TIRD} \|t\|_V$ where $V = BV(s) \cup BV(t)$. In the following, $\|s\|_V$ and $\|t\|_V$ are abbreviated just as $\|s\|$ and $\|t\|$, respectively.

We must show that HIRD is a partial ordering that is stable under ground normalized substitutions and ground contexts and it is well-founded on ground normalized terms. These properties are essential for applying HIRD to termination proof of HRS. In this chapter $\mathcal{T}(\lambda\mathcal{C})$ denotes the set of type compatible ground algebraic terms and $\mathcal{T}(\mathcal{C})\downarrow$ denotes the set of type compatible ground normalized terms. First, we show that the HIRD is a partial ordering on $\mathcal{T}(\mathcal{C})\downarrow$.

Lemma 4.3.4 The TIRD is a partial ordering on $\mathcal{T}(\lambda\mathcal{C})$.

Proof. To prove the transitivity of $\gg\gg_{EL}$, it is enough to show that $>_{EL}$ is transitive. Let $s' \in dec_p(s)$, $t' \in dec_q(t)$ and $u' \in dec_r(u)$. If $type(s') >_{\mathcal{T}_S} type(t')$ or $type(t') >_{\mathcal{T}_S} type(u')$, the claim is trivial. If $type(s') = type(t')$ and $type(t') = type(u')$, we can show that $dec_p(s) \ni s' >_{EL} u' \in dec_q(t)$ by induction on $|s'| + |t'| + |u'|$. We next show that $s \not>_{TIRD} s$ by proving the irreflexivity of $>_{EL}$. For any term s and s' in $dec_p(s)$, we can easily prove that $dec_p(s) \ni s' \not>_{EL} s' \in dec_p(s)$ by induction on $|s|$. □

Lemma 4.3.5 The HIRD is a partial ordering on $\mathcal{T}(\mathcal{C})\downarrow$.

Proof. It is straightforward by lemma 4.3.4. □

Next, we show that the HIRD is well-founded on $\mathcal{T}(\mathcal{C})\downarrow$.

Lemma 4.3.6 The TIRD has the subterm property for $\mathcal{T}(\lambda\mathcal{C})$.

Proof. Let s and t be algebraic terms such that $s \rhd t$. We can show $s >_{TIRD} t$ by induction on $|s|$. □

Lemma 4.3.7 The TIRD is stable under ground contexts.

Proof. Let s and t be algebraic terms with the same basic type. We have to show that $s >_{TIRD} t$ implies $C[s] >_{TIRD} C[t]$ for any ground context $C[\ ]$. It can be proved by induction on $|C[\ ]|$ by the similar way to IRD. □

Lemma 4.3.8 The TIRD is a simplification order on $\mathcal{T}(\lambda\mathcal{C})$.

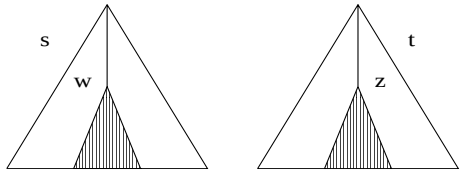

Figure 4.3: Case (1) in lemma 4.3.15

Proof. By lemmas 4.3.4, 4.3.6 and 4.3.7, the TIRD is a partial ordering on $\mathcal{T}(\lambda\mathcal{C})$ that is stable under ground contexts and has the subterm property. □

In the following lemma, we use the hypothesis that there are finitely many symbols for a given output type σ. This is the modification of Kruskal's tree theorem for finite signature.

Lemma 4.3.9 ([28]) Let $t_0, t_1, t_2, \ldots$ be an infinite sequence with type compatible ground algebraic terms of the same type σ. Then, there exist indexes i, j $(i < j)$ such that $t_i \leq_{emb} t_j$.

Lemma 4.3.10 The TIRD is well-founded on $\mathcal{T}(\lambda\mathcal{C})$.

Proof. By lemmas 4.3.8 and 4.3.9, the TIRD is well-founded on $\mathcal{T}(\lambda\mathcal{C})$. □

Lemma 4.3.11 The HIRD is well-founded on $\mathcal{T}(\mathcal{C})\downarrow$.

Proof. It is straightforward by lemma 4.3.10. □

We write a normalized substitution just as a substitution in the rest of this chapter.

We prove that the stability under ground substitution θ where $dom(\theta)$ is a singleton set. Since any ground substitution θ can be denoted by composition of ground substitutions θ_i where $dom(\theta_i)$ is a singleton set, the stability under ground substitutions is easily reduced from the above statement.

The figures 4.3, 4.4 and 4.5 exhibit the cases (1), (2) and (3) in the lemma 4.3.15, respectively.

Example 4.3.12 We consider the following normalized terms.

$s = X(\lambda xy.x)$ and $t = X(\lambda xy.y)$ where $X{:}(Nat \to Nat \to Nat) \to Nat$, $x{:}Nat$ and $y{:}Nat$. Then, $\| s \| = X_{Nat}(\lambda_{Nat \to Nat \to Nat}(\lambda_{Nat \to Nat}(c_{Nat}))) = \| t \|$ holds.

Let $\theta = \{X \leftarrow \lambda z.z(0,1)\}$ where $z{:}Nat \to Nat \to Nat$, $0 : Nat$ and $1 : Nat$. Since $s\theta\downarrow = 0_{Nat}$ and $t\theta\downarrow = 1_{Nat}$, $\| s\theta\downarrow \| \neq \| t\theta\downarrow \|$. Therefore, it does not hold that $\| s \| = \| t \|$ implies $\| s\theta\downarrow \| = \| t\theta\downarrow \|$ for any ground substitution θ in general.

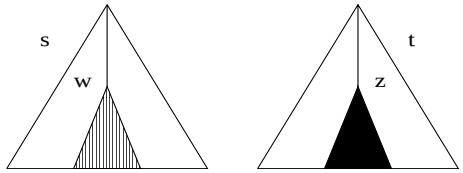

Figure 4.4: Case (2) in lemma 4.3.15

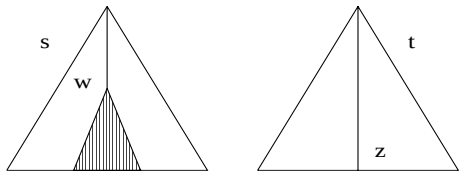

Figure 4.5: Case (3) in lemma 4.3.15

The above example explains why each bound variable is restricted into a basic type to guarantee the stability of ground substitutions.

We assume that each bound variable in normalized terms has a basic type in the rest of this chapter.

Lemma 4.3.13 Let s and t be normalized terms. Then, $\| s \|_V = \| t \|_V$ implies $\| s\theta\downarrow \|_V = \| t\theta\downarrow \|_V$ for any ground substitution θ.

Proof. We define a restriction of substitution θ with respect to set V.

$$\theta/_V(x) = \begin{cases} \theta(x) & \text{if } x \notin V \\ x & \text{if } x \in V. \end{cases}$$

We show $\| s \|_V = \| t \|_V$ implies $\| s(\theta/_V)\downarrow \|_V = \| t(\theta/_V)\downarrow \|_V$ by induction on structure of s. We abbreviate an index V.

In the case that $s = \alpha(s_1,\cdots,s_m)$ for $\alpha \in \mathcal{C}$ or $s = \lambda x.s_1$ for $x \in V$, the proof is straightforward. We consider the case that $s = \alpha(s_1,\cdots,s_m)$ if $\alpha \in \mathcal{V} \setminus V$.

Since $s = \alpha(s_1,\cdots,s_m)$ and $\| s \| = \| t \|$, we have $t = \alpha(t_1,\cdots,t_m)$ such that $\| s_i \| = \| t_i \|$ $(1 \leq i \leq m)$.

Then $s\theta = (\alpha\theta)(s_1\theta,\cdots,s_m\theta)$ and $t\theta = (\alpha\theta)(t_1\theta,\cdots,t_m\theta)$. Letting $s' = \alpha\theta$, we show $\|s'(s_1\theta,\cdots,s_m\theta)\downarrow\| = \|s'(t_1\theta,\cdots,t_m\theta)\downarrow\|$ by induction on size of s'.

If $s' = \lambda x_1\cdots x_m.x_i$ then $\| s_i\theta\downarrow \| = \| t_i\theta\downarrow \|$ by induction hypothesis. Let $s' = \lambda\ x_1 \cdots\ x_m.f(u_1,\cdots,u_n)$ $(n \geq 0$ and $f \in \mathcal{C})$. Let $\gamma = \{x_1 \leftarrow s_1\theta,\cdots,x_m \leftarrow s_m\theta\}$.

$\|s'(s_1\theta,\cdots,s_m\theta)\downarrow\|$

$= f(\|u_1\gamma\downarrow\|, \cdots, \|u_n\gamma\downarrow\|)$

$= f(\|(\lambda\ x_1 \cdots\ x_m.u_1)\ (s_1\theta, \cdots, s_m\theta)\downarrow\|, \cdots, \|(\lambda\ x_1 \cdots\ x_m.u_n)\ (s_1\theta, \cdots, s_m\theta)\downarrow\|)$

$= f(\|(\lambda\ x_1 \cdots\ x_m.u_1)\ (t_1\theta, \cdots, t_m\theta)\downarrow\|, \cdots, \|(\lambda\ x_1 \cdots\ x_m.u_n)\ (t_1\theta, \cdots, t_m\theta)\downarrow\|)$ by induction hypothesis. □

Definition 4.3.14 Let s be a algebraic term and θ a ground substitution. $s\theta\downarrow$ denotes $\| t\theta\downarrow \|$ for some normalized term t such that $\| t \| = s$. Let $\{s_1,\cdots,s_n\}$ be a (multi)set of algebraic terms. $\{s_1,\cdots,s_n\}\theta\downarrow$ denotes $\{s_1\theta\downarrow,\cdots,s_n\theta\downarrow\}$.

Lemma 4.3.15 Let $dec_w(\|s\|) \gg_{EL} dec_z(\|t\|)$ where s and t be normalized terms, $w \in Op(s)$ and $z \in Op(t)$. Then for any ground substitution θ, the following three claims hold.

(1) If $\|s\|_w = \|t\|_z$ and $root(\|t\|_z) \in \widetilde{\mathcal{V}}$ then $dec_{w.j}(\|s\theta\downarrow\|) \gg_{EL} dec_{z.j}(\|t\theta\downarrow\|)$ for any $j \in N^*$ such that $z.j \in Op(\|t\theta\downarrow\|)$.

(2) If $\|s\|_w \neq \|t\|_z$ and $root(\|t\|_z) \in \widetilde{\mathcal{V}}$ then $dec_{w.i}(\|s\theta\downarrow\|) \gg_{EL} dec_{z.j}(\|t\theta\downarrow\|)$ for any j, $i \in N^*$ such that $z.j \in Op(\|t\theta\downarrow\|))$ and $w.i \in Op(\|s\theta\downarrow\|)$.

(3) If $root(\|t\|_z) \notin \widetilde{\mathcal{V}}$ then $dec_{w.i}(\|s\theta\downarrow\|) \gg_{EL} dec_z(\|t\theta\downarrow\|)$ for any $i \in N^*$ such that $w.i \in Op(\|s\theta\downarrow\|)$.

Proof. Let $s' = \| s \|$ and $t' = \| t \|$. We show that the claim (1) $\wedge$ (2) $\wedge$ (3) by induction on $|s'| + |t'|$. Assume that $dec_w(s') \gg_{EL} dec_z(t')$.

(1) Consider the case $s'|_w = t'|_z$ and $root(t'|_z) \in \widetilde{\mathcal{V}}$.

Since the assumption $dec_w(s') \gg_{EL} dec_z(t')$ and the definition of multiset extension, we consider the case that $dec_w(s') = M \cup \{s_1,\cdots,s_m\}$, $dec_z(t') = M \cup \{t_1,\cdots,t_n\}$ and for any $k \in \{1,\cdots,n\}$, there exists $l \in \{1,\cdots,m\}$ such that $dec_w(s') \ni s_l >_{EL} t_k \in dec_z(t')$.

For any $j \in N^*$ ($z.j \in Op(t'\theta\downarrow)$), we can denote that $dec_{w.j}(s'\theta\downarrow) = M\theta\downarrow \cup \{s_1\theta\downarrow, \cdots, s_m\theta\downarrow\} \cup L$ and $dec_{z.j}(t'\theta\downarrow) = M\theta\downarrow \cup \{t_1\theta\downarrow ,\cdots, t_n\theta\downarrow\} \cup L$ where $L = \{v \mid v \in sub(dec_j(s'|_w\theta\downarrow),s'|_w\theta\downarrow)\}$ by lemma 4.3.13. Hence we have to show that $dec_w(s') \ni s_l >_{EL} t_k \in dec_z(t')$ implies $dec_{w.j}(s'\theta\downarrow) \ni s_l\theta\downarrow >_{EL} t_k\theta\downarrow \in dec_{z.j}(t'\theta\downarrow)$. We distinguish the cases with respect to the definition of $>_{EL}$.

$(1-1)$ $type(s_l) >_{\mathcal{T}_S} type(t_k)$. Then, $type(s_l\theta\downarrow) >_{\mathcal{T}_S} type(t_k\theta\downarrow)$ holds.

$(1-2)$ $type(s_l) = type(t_k)$.

(a) If $root(s_l) >_{\lambda C} root(t_k)$ then $root(s_l\theta\downarrow) >_{\lambda C} root(t_k\theta\downarrow)$ holds.

(b) If $root(s_l) = root(t_k)$, $status(root(s_l)) = mult$ and $sub\ (dec_w(s'),s_l) \gg_{EL} sub\ (dec_z(t'),t_k)$ then we can show $sub\ (dec_{w.j}(s'\theta\downarrow),\ s_l\theta\downarrow) \gg_{EL} sub\ (dec_{z.j}(t'\theta\downarrow),\ t_k\theta\downarrow)$ by induction hypothesis.

(c) We consider the case that $root(s_l) = root(t_k)$, $status(root(s_l)) = mult$, $sub\ (dec_w(s'),s_l) = sub\ (dec_z(t'),t_k)$ and $dec(st^m(s_l)) \gg\gg_{EL} dec(st^m(t_k))$. Since types and induction hypothesis, it follows that $dec\ (st^m(s_l\theta\downarrow)) \gg\gg_{EL} dec\ (st^m(t_k\theta\downarrow))$.

(d) Consider the case that $root\ (s_l) = root\ (t_k)$, $status\ (root(s_l)) \neq mult$, $st\ (s_l) >_{TIRD}^{status(root(s_l))} st\ (t_k)$ and $\{s_l\} \gg_{TIRD} st^m\ (t_k)$. We can show $st\ (s_l\theta\downarrow) >_{TIRD}^{status(root(s_l\theta\downarrow))} st\ (t_k\theta\downarrow)$ and $\{s_l\theta\downarrow\} \gg_{TIRD} st^m\ (t_k\theta\downarrow)$ by types, in case of $type(s_l) >_{\mathcal{T}_S} type(t_k)$ and by induction hypothesis, in case of $type(s_l) = type(t_k)$.

(2) In case of $s'|_w \neq t'|_z$ and $root\ (t'|_z) \in \widetilde{\mathcal{V}}$, we have to consider only the case that $dec_w(s') = M \cup \{s_1,\cdots,s_m\}$ and $dec_z(t') = M \cup \{t_1,\cdots,t_n\}$. Since the assumption $dec_w(s') \gg_{EL} dec_z(t')$ and the definition of multiset extension, for any $k \in \{1,\cdots,n\}$ there exists $l \in \{1,\cdots,m\}$ such that $dec_w(s') \ni s_l >_{EL} t_k \in dec_z(t')$. For any $j \in N^*$ $(z.j \in Op(t'\theta\downarrow))$ and any $i \in N^*$ $(w.i \in Op(s'\theta\downarrow))$, we can show that $dec_{w.i}(s'\theta\downarrow) = M\theta\downarrow \cup \{s_1\theta\downarrow,\cdots, s_m\theta\downarrow\} \cup L$ where $L = \{v \mid v \in sub(dec_i(s'|_w\theta\downarrow), s'|_w\theta\downarrow)\}$ and $dec_{z.j}(t'\theta\downarrow) = M\theta\downarrow \cup \{t_1\theta\downarrow,\cdots,t_n\theta\downarrow\} \cup L'$ where $L' = \{v' \mid v' \in sub(dec_j(t'|_z\theta\downarrow), t'|_z\theta\downarrow)\}$ by lemma 4.3.13. Since $t'|_z \notin dec_w(s')$, $t'|_z \in \{t_1, \cdots, t_n\}$ holds. That is, $dec_w(s') \ni s_l >_{EL} t'|_z \in dec_z(t')$. $type(s_l) >_{\mathcal{T}_S} type(t'|_z)$ holds, by $root(t'|_z) \in \widetilde{\mathcal{V}}$. Since $t'|_z\theta\downarrow \rhd y'$ for any $y' \in L'$ and the type compatibility, $type(t'|_z\theta\downarrow) \geq_{\mathcal{T}_S} type(y')$. Hence $type(s_l\theta\downarrow) >_{\mathcal{T}_S} type(y')$, i.e., $dec_{w.i}(s'\theta\downarrow) \ni s_l\theta\downarrow >_{EL} y' \in dec_{z.j}(t'\theta\downarrow)$ for any $y' \in sub(dec_j(t'|_z\theta\downarrow), t'|_z\theta\downarrow)$. Further we can show that $dec_w(s') \ni s_l >_{EL} t_k \in dec_z(t')$ implies $dec_{w.i}(s'\theta\downarrow) \ni s_l\theta\downarrow >_{EL} t_k\theta\downarrow \in dec_{z.j}(t'\theta\downarrow)$, in similar to the proof of (1).

(3) In case of $root(t'|_z) \notin \widetilde{\mathcal{V}}$, for any $i \in N^*$ $(w.i \in Op(t'\theta\downarrow))$, we can denote that $dec_{w.i}(s'\theta\downarrow) = M\theta\downarrow \cup \{s_1\theta\downarrow,\cdots, s_m\theta\downarrow\} \cup L$ where $L = \{v \mid v \in sub(dec_i(s'|_w\theta\downarrow), s'|_w\theta\downarrow)\}$ and $dec_z(t'\theta\downarrow) = M\theta\downarrow \cup \{t_1\theta\downarrow,\cdots, t_n\theta\downarrow\}$ by lemma 4.3.13. Hence we can show that $dec_w(s') \ni s_l >_{EL} t_k \in dec_z(t')$ implies $dec_{w.i}(s'\theta\downarrow) \ni s_l\theta\downarrow >_{EL} t_k\theta\downarrow \in dec_z(t'\theta\downarrow)$, in similar to the proof of (1). □

Lemma 4.3.16 Let s and t be algebraic terms. Then $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$ implies $dec(\{s\}) \cap dec(\{t\}) = \emptyset$.

Proof. Let $M \in dec(\{s\}) \cap dec(\{t\})$. Then $M = dec_w(s) = dec_z(t)$ for some $w \in Op(s)$ and $z \in Op(t)$. Then $s = t$ must hold from the definition of the path-decomposition. This contradicts the assumption from the irreflexivity of $\gg_{EL}$. Hence $dec(\{s\}) \cap dec(\{t\}) = \emptyset$. □

Lemma 4.3.17 The HIRD is stable under ground substitutions, i.e., $s >_{HIRD} t$ implies $s\theta\downarrow >_{HIRD} t\theta\downarrow$.

Proof. Assume that $s >_{HIRD} t$, i.e., $dec(\{\|\, s \,\|\}) \gg\gg_{EL} dec(\{\|\, t \,\|\})$ where s and t are normalized terms. We show that $dec(\{\|\, s\theta\downarrow \,\|\}) \gg\gg_{EL} dec(\{\|\, t\theta\downarrow \,\|\})$ holds for any ground substitution θ. By lemma 4.3.16, we can assume that for any $z \in Op(\|t\|)$, there exists $w \in Op(\|s\|)$ such that $dec_w(\|s\|) \gg_{EL} dec_z(\|t\|)$. Then we can show the following claim since the definition of the multiset extension and lemma 4.3.15: for any z' $\in Op(\|t\theta\downarrow\|)$, there exists $w' \in Op(\|s\theta\downarrow\|)$ such that $dec_{w'}(\|s\theta\downarrow\|) \gg_{EL} dec_{z'}(\|t\theta\downarrow\|)$. □

In order to show that the HIRD is stable under ground contexts in lemma 4.3.19, we need the following lemma.

Lemma 4.3.18 Let s and t be normalized terms. Let $V = BV(s) \cup BV(t)$. For any set of variables V' such that $V' \supseteq V$, $\|s\|_V >_{TIRD} \|t\|_V$ implies $\|s\|_{V'} >_{TIRD} \|t\|_{V'}$.

Proof. Let $V' \setminus V = \{x_1,\cdots,x_m\}$ $(m \geq 0)$. Let $\theta = \{x_1 \leftarrow c,\cdots,x_m \leftarrow c\}$. By definition 4.3.1, we can consider that $\|s\|_{V'}$ and $\|t\|_{V'}$ equal to $\|s\theta\downarrow\|_V$ and $\|t\theta\downarrow\|_V$ for $\theta = \{x_1 \leftarrow c,\cdots,x_m \leftarrow c\}$, respectively. $\|s\theta\downarrow\|_V >_{TIRD} \|t\theta\downarrow\|_V$ holds by lemma 4.3.17. Hence $\|s\|_{V'} >_{TIRD} \|t\|_{V'}$ holds. □

Lemma 4.3.19 The HIRD is stable under ground contexts, i.e., $s >_{HIRD} t$ implies $C[s] >_{HIRD} C[t]$ for any ground context C and normalized terms s and t with the same basic type such that $C[s]$ and $C[t]$ are normalized terms.

Proof. By definition 4.3.3, we prove that $\|s\|_V >_{TIRD} \|t\|_V$ implies $\|C[s]\|_{V'} >_{TIRD} \|C[t]\|_{V'}$, where $V = BV(s) \cup BV(t)$ and $V' = BV(C[s]) \cup BV(C[t])$ by induction on $|C[\ ]|$. Then it is obvious that $V' \supseteq V$ holds. It is enough to consider the only case that $C[\ ] = \alpha(u_1,\cdots,\Box,\cdots,u_n)$. Since lemma 4.3.18, $\|s\|_{V'} >_{TIRD} \|t\|_{V'}$. We consider the following cases.

1. If $\alpha : \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \sigma \in \mathcal{C}$ then $\|\alpha\ (u_1, \cdots, s, \cdots, u_n)\|_{V'} = \alpha_\sigma\ (\|u_1\|_{V'}, \cdots, \|s\|_{V'}, \cdots, \|u_n\|_{V'})$ and $\|\alpha\ (u_1, \cdots, t, \cdots, u_n)\|_{V'} = \alpha_\sigma\ (\|u_1\|_{V'}, \cdots, \|t\|_{V'}, \cdots, \|u_n\|_{V'})$. Hence $\|\alpha\ (u_1, \cdots, s, \cdots, u_n)\|_{V'} >_{TIRD} \|\alpha\ (u_1, \cdots, t, \cdots, u_n)\|_{V'}$ by lemma 4.3.7.

2. $\alpha = \lambda\, x$ for some variable x of type σ. $\|\lambda\, x.s\|_{V'} = \lambda(\|s\|_{V'})$ and $\|\lambda\, x.t\|_{V'} = \lambda(\|t\|_{V'})$. Hence $\|\lambda\, x.s\|_{V'} >_{TIRD} \|\lambda\, x.t\|_{V'}$ by lemma 4.3.7. □

Note that we assume that there exists at least one constant in $\mathcal{C}$ for each type in the following. Next theorem guarantees that we can use the HIRD to prove the termination of HRSs on type compatible normalized terms.

Theorem 4.3.20 Let $\mathcal{R}$ be a HRS on type compatible normalized terms. Let $>_{\lambda\mathcal{C}}$ be a partial ordering on $\lambda\mathcal{C}$. If $l >_{HIRD} r$ for any rewrite rule $l \rightarrow r$ in $\mathcal{R}$ then $\mathcal{R}$ is terminating on type compatible normalized terms.

Proof. Since terms in a rewrite sequence can always be considered as ground, we can use the previous properties. Assume that $s \rightarrow_{\mathcal{R}} t$, where s and t are ground normalized terms. Then $s = C[l\theta\downarrow]$ and $t = C[r\theta\downarrow]$ for a rewrite rule $l \rightarrow r \in \mathcal{R}$, a substitution $\theta{:}\mathcal{V} \rightarrow \mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$ and a ground context C. Let $FV(l\theta\downarrow) = \{x_1, \ldots, x_n\}$ where $x_i{:}\sigma_i$ $(1 \leq i \leq n)$. Let $\theta' = \{x_1 \leftarrow c_{\sigma_1}, \ldots, x_n \leftarrow c_{\sigma_n}\}$. Note that $l\theta\theta'\downarrow$ is a ground term and substitution $\theta\theta'$ is ground. Since the assumption $l >_{HIRD} r$ and lemma 4.3.17, $\|\, l\theta\theta'\downarrow\,\| >_{TIRD} \|\, r\theta\theta'\downarrow\,\|$ holds. Thus, it follows that $\|\, s\,\| = \|\, C[l\theta\theta'\downarrow]\,\| >_{TIRD} \|\, C[r\theta\theta'\downarrow]\,\| = \|\, t\,\|$ by lemma 4.3.19. Hence, $s >_{HIRD} t$ holds. Since the HIRD is well-founded on $\mathcal{T}(\mathcal{C})\downarrow$ by lemma 4.3.11, $\mathcal{R}$ is terminating on type compatible normalized terms. □

Example 4.3.21 Let $\mathcal{S} = \{Nat, List\}$, $\mathcal{C} = \{nil{:}List,\ cons{:}Nat \rightarrow List \rightarrow List,$ $map{:}(Nat \rightarrow Nat) \rightarrow List \rightarrow List\}$, $\{X{:}Nat \rightarrow Nat,\ N{:}Nat,\ L{:}List\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \left\{ \begin{array}{l} map(\lambda\, x.X(x), nil) \rightarrow nil \\ map(\lambda\, x.X(x), cons(N, L)) \rightarrow cons(X(N), map(\lambda\, x.X(x), L)). \end{array} \right.$$

The termination proof uses precedences and status: $List >_{\mathcal{S}} Nat$, $List >_{\mathcal{S}} \rightarrow$, $map_{List} >_{\lambda\mathcal{C}} cons_{List}$ and $status(map_{List}) = mult$. Since it is obvious that $map(\lambda x.X(x),nil) >_{HIRD} nil$, we consider the second rule. Let $s = map(\lambda x.X(x),cons(N,L))$ and $t = cons(X(N),map\ (\lambda x.X(x),L))$. Then we have $dec(\{\!\|\ s\ \|\!\})$ and $dec(\{\!\|\ t\ \|\!\})$.

- $dec(\{\!\|s\|\!\}) = \{dec_{11}(\|s\|),\ dec_{21}(\|s\|),\ dec_{22}(\|s\|)\}$.
- $dec(\{\!\|t\|\!\}) = \{dec_{1}(\|t\|),\ dec_{211}(\|t\|),\ dec_{22}(\|t\|)\}$.

By example 4.2.14, we have $dec(\{\!\|s\|\!\}) \gg\gg_{EL} dec(\{\!\|t\|\!\})$, i.e., $\|s\| >_{TIRD} \|t\|$. Hence $s >_{HIRD} t$ holds. Therefore, $\mathcal{R}$ is terminating on type compatible normalized terms by theorem 4.3.20.

4.4 Related Works

In this section, we compare our higher-order improved recursive decomposition ordering (HIRD) with the Jouannaud and Rubio's ordering [27, 28]. We denote the Jouannaud and Rubio's ordering as *higher-order recursive path ordering* (HRPO) in the rest of this chapter. The IRD is more powerful than RPO in the frame of TRSs [59, 63]. The similar result holds in the frame of HRSs. The next theorem shows that HIRD is more powerful than HRPO on type compatible normalized terms. We use our interpretation function $\|\quad\|$ in definition 4.3.1 for HRPO instead of it defined in [27, 28].

Theorem 4.4.1 Let s and t be type compatible normalized terms. Then $s >_{HRPO} t$ implies $s >_{HIRD} t$.

Proof. This proof is obtained by a modification of the proof of the following claim. $u >_{RPO} v$ implies $u >_{IRD} v$ for any term u and v [60, 61]. It holds that $s' >_{HRPO} t'$ if and only if $\|s'\| >_{TRPO} \|t'\|$ by definition of the HRPO [27, 28] and $s' >_{HIRD} t'$ if and only if $\|s'\| >_{TIRD} \|t'\|$ by the definition 4.3.3. We have to show that $s >_{TRPO} t$ implies $s >_{TIRD} t$ for any algebraic terms s and t. The proof is performed by using induction on $|s| + |t|$. Let $s = \alpha_\sigma(t_1,\cdots,t_m)$ and $t = \beta_\tau(s_1,\cdots,s_n)$. We distinguish the cases with respect to the definition of TRPO, *typed recursive path ordering* [27, 28].

(1) $\sigma >_{\mathcal{T}_\mathcal{S}} \tau$. For any $w \in Op(s)$ and any $z \in Op(t)$, $dec_w(s) \ni s >_{EL} t \in dec_z(t)$. For any $t_j \in dec_z(t)$, $type(t) \geq_{\mathcal{T}_\mathcal{S}} type(t_j)$ from type compatibility. Thus, $dec_w(s) \ni s >_{EL} t_j \in dec_z(t)$ holds since $type(s) >_{\mathcal{T}_\mathcal{S}} type(t_j)$. Since for any $z \in Op(t)$, there exists $w \in Op(s)$ such that $dec_w(s) \gg_{EL} dec_z(t)$, we have $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$.

(2) $\sigma = \tau$.

(a) $\alpha_\sigma \notin \widetilde{\mathcal{V}}$ and $s_i \geq_{TRPO} t$ for some i.
By induction hypothesis, $s_i \geq_{TIRD} t$. Hence $dec(\{s_i\}) \underline{\gg\gg}_{EL} dec(\{t\})$ holds. Then $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$ because for any $dec_w(s_i) \in dec(\{s_i\})$, there exists $dec_{i.w}(s) \in dec(\{s\})$ such that $dec_{i.w}(s) \gg_{EL} dec_w(s_i)$.

(b) $\alpha_\sigma >_{\lambda\mathcal{C}} \beta_\tau$ and $s >_{TRPO} t_i$ for all i.
For all $i \in \{1,\cdots,n\}$, $s >_{TIRD} t_i$ holds by induction hypothesis. Thus, $dec(\{s\}) \gg\gg_{EL} dec(\{t_i\})$ $(i = 1,\cdots,n)$ holds by definition 4.2.13. Note that $dec(\{t\}) = \{d \cup \{t\} \mid d \in dec(\{t_i\}),\ i \in \{1,\cdots,n\}\}$. For any $z \in Op(t_i)$, there exists $w \in Op(s)$ such that $dec_w(s) \gg_{EL} dec_z(t_i)$ $(i = 1,\cdots,n)$. Since $root(s) = \alpha_\sigma >_{\lambda\mathcal{C}} \beta_\tau = root(t)$, $dec_w(s) \ni s >_{EL} t \in dec_{i.z}(t)$ holds. Since there exists no subterm of t that is equal to s (otherwise s would not be greater than all t_i's), for any $z' \in Op(t)$, there exists $w' \in Op(s)$ such that $dec_{w'}(s) \gg_{EL} dec_{z'}(t)$. Hence $dec(\{s\}) \gg\gg_{EL} dec(\{t\})$ holds.

(c) $\alpha_\sigma = \beta_\tau$, $status(\alpha_\sigma) = mult$ and $st^m(s) \gg_{TRPO} st^m(t)$.
It holds that $st^m(s) \gg_{TIRD} st^m(t)$ by induction hypothesis. For any $t_j \in st^m(t)$, there exists $s_i \in st^m(s)$ such that $s_i \geq_{TIRD} t_j$ and $st^m(s) \neq st^m(t)$ by the definition of the multiset extension. Then for any $t_j \in st^m(t)$, there exists $s_i \in st^m(s)$ such that $dec(\{s_i\}) \underline{\gg\gg}_{EL} dec(\{t_j\})$ $\cdots$ $(*)$. Note that $dec(\{s\}) = \{d \cup \{s\} \mid d \in dec(\{s_i\}),\ i \in \{1,\cdots,m\}\}$ and $dec(\{t\}) = \{d \cup \{t\} \mid d \in dec(\{t_j\}),\ j \in \{1,\cdots,n\}\}$. We have to show that for any $z \in Op(t)$, there exists $w \in Op(s)$ such that $dec_w(s) \ni s >_{EL} t \in dec_z(t)$. Then we have to prove that either $sub(dec_w(s),s) \gg_{EL} sub(dec_z(t),t)$ or $sub(dec_w(s),s) = sub(dec_z(t),t)$ and $dec(st^m(s)) \gg\gg_{EL} dec(st^m(t))$. This can be easily shown with $(*)$.

(d) $\alpha_\sigma = \beta_\tau$, $status(\alpha_\sigma) \neq mult$, $st(s) >_{TRPO}^{status(\alpha_\sigma)} st(t)$ and $\{s\} \gg_{TRPO} st^m(t)$.
Then $st(s) >_{TIRD}^{status(\alpha_\sigma)} st(t)$ and $\{s\} \gg_{TIRD} st^m(t)$ by induction hypothesis. □

The following example shows that Jouannaud and Rubio's ordering is properly included in our HIRD.

Example 4.4.2 ([60]) Let $\mathcal{S} = \{form\}$, $\mathcal{C} = \{\neg{:}form \to form,\ \supset{:}form \to form \to form,\ \vee{:}form \to form \to form\}$, $\{X{:}form,\ Y{:}form,\ Z{:}form\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \{\ \neg X \supset (Y \supset Z) \longrightarrow Y \supset (X \vee Z).$$

Give the precedence: $\neg_{form} >_{\lambda\mathcal{C}} \supset_{form} >_{\lambda\mathcal{C}} \vee_{form}$. Then $\neg\ X \supset (Y \supset Z) >_{HIRD} Y \supset (X \vee Z)$ but $\neg\ X \supset (Y \supset Z) \not>_{HRPO} Y \supset (X \vee Z)$.

Hence, we can show the termination of $\mathcal{R}$ on type compatible normalized terms using our HIRD, but Jouannaud and Rubio's ordering cannot show it. Therefore, HIRD is more powerful than the ordering defined by Jouannaud and Rubio since theorem 4.4.1 and example 4.4.2.

Jouannaud and Rubio [28] showed that for some non-type compatible terms, termination is also assured by the termination of HRS on type compatible terms. However, the restrictions for type structure is essential in their approach.

4.5 Conclusion

In this chapter, we have extended the IRD to HRSs for proving termination of HRSs. Our extension method is inspired by Jouannaud and Rubio's idea [27, 28] and the particular properties of IRD. We have shown that our ordering is more powerful than the ordering defined by Jouannaud and Rubio. Hence our ordering is powerful tool for proving termination of HRS on type compatible terms. Jouannaud and Rubio [28] showed that for some non-type compatible terms, termination is also assured by the termination of HRS on type compatible terms. However the restriction of type structure is essential in this chapter and Jouannaud and Rubio's works [27, 28]. In order to remove this restriction, we consider the RPO for HRSs that is independent of an ordering on type structure in the next chapter.

Chapter 5

Simplification Ordering for Higher-Order Rewrite Systems

5.1 Introduction

Recently Jouannaud and Rubio [27, 28] extended the recursive path ordering (RPO) on first-order terms, called algebraic terms, to higher-order terms by using a first-order interpretation on λ-terms. They showed that this ordering can prove termination of several interesting examples. However, in this ordering, two higher-order terms have to be compared by type first, then by root function symbol, before the comparison can proceed recursively on the arguments. This unnatural priority between type and function symbol restricts their ordering to only on type compatible terms.

In this chapter we introduce the notion of *simplification orderings* on algebraic terms and propose a new powerful RPO on higher-order terms, called the *higher-order recursive path ordering* (HRPO). Though our approach was inspired by the first-order interpretation method described in Jouannaud and Rubio [27, 28], we develop the higher-order recursive path ordering within a more natural framework of a simplification ordering. Our key idea in higher-order recursive path ordering is the concept of *envelopes* for typed terms that allows us to treat higher-order variables as function symbols. We clarify that the priority between type and function symbol introduced in Jouannaud and Rubio is not essential and any partial ordering on function symbols can be used freely to define the higher-order recursive path ordering. Thus we can remove the type compatible term limitation in Jouannaud and Rubio's ordering [27, 28].

Section 5.2 presents the definition of simplification orderings on algebraic terms. We define the higher-order recursive path ordering on normalized terms in section 5.3. Section 5.4 develops the technique of envelopes, and section 5.5 compares our approach to that of Jouannaud and Rubio.

5.2 Simplification Ordering for Higher-Order Rewrite Systems

In this section, we introduce the notion of simplification orderings for HRSs. More precisely, we define the simplification orderings on algebraic terms. Further we show the termination criteria for HRSs using a framework of simplification orderings.

Definition 5.2.1 Let $\mathcal{L} = \{\lambda_{\tau\rightarrow(\sigma\rightarrow\tau)} \mid \sigma,\ \tau \in \mathcal{T}_S\}$ and $\mathcal{B} = \{c_\sigma \mid \sigma \in \mathcal{T}_S\}$ where $\lambda_{\tau\rightarrow(\sigma\rightarrow\tau)}{:}\tau \rightarrow (\sigma \rightarrow \tau)$ and $c_\sigma{:}\sigma$ are fresh constants. We define the new signature $\lambda\mathcal{C} = \mathcal{C} \cup \mathcal{L} \cup \mathcal{B}$.

Algebraic terms are typed terms over $\lambda\mathcal{C}$ obtained from normalized terms over $\mathcal{C}$ through the following interpretation function.

Definition 5.2.2 The *interpretation function* $\|\ \ \|$ from normalized terms over the signature $\mathcal{C} \cup \mathcal{B}$ to algebraic typed terms over the signature $\lambda\mathcal{C}$ is defined by:

- $\|(\lambda x.s){:}\sigma{\rightarrow}\tau\| = \lambda_{\tau\rightarrow(\sigma\rightarrow\tau)}(\|s\{x \leftarrow c_\sigma\}\|)$.
- $\|\alpha(s_1,\cdots,s_n)\| = \alpha(\|s_1\|,\cdots,\|s_n\|)$ if $\alpha \in \mathcal{C} \cup \mathcal{B} \cup \mathcal{V}$.

Then, the set of algebraic terms is $\mathcal{T}(\lambda\mathcal{C},\mathcal{V}) = \{\| t \| \mid t \in \mathcal{T}(\mathcal{C},\mathcal{V})\downarrow\}$ and the set of ground algebraic terms is $\mathcal{T}(\lambda\mathcal{C}) = \{\| t \| \mid t \in \mathcal{T}(\mathcal{C})\downarrow\}$.

Lemma 5.2.3 Let $C[s] \in \mathcal{T}(\mathcal{C})\downarrow$ and $s \in \mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$. Let $\theta = \{x_1 \leftarrow c_{\sigma_1},\ldots,x_n \leftarrow c_{\sigma_n}\}$ where $x_i{:}c_{\sigma_i}$ $(1 \leq i \leq n)$ and $FV(s) = \{x_1,\ldots,x_n\}$. Then, $\| C[s] \| = \| C \|[\| s\theta \|]$.

Proof. It is straightforward by the definition of the interpretation function. □

$Fun(t)$ denotes the set of constants in an algebraic term t. The size $| t |$ is defined as the number of symbols occurring in t.

Definition 5.2.4 A *root symbol* of an algebraic term is defined by $root(\alpha(s_1,\ldots,s_m)) = \alpha$ if $\alpha \in \lambda\mathcal{C} \cup \mathcal{V}$. Note that an algebraic term $\alpha(s_1,\ldots,s_n)$ has a basic type if and only if $\alpha \notin \mathcal{L}$.

An algebraic term s_i is called an *immediate subterm* of an algebraic term $s = \alpha(s_1, \ldots, s_i, \ldots, s_n)$. Then, $st(s) = \langle\ s_1,\ldots,s_i,\ldots,s_n\ \rangle$ and $st^m(s) = \{s_1,\ldots,s_i,\ldots,s_n\}$ denote the sequence and the multiset of immediate subterms of s, respectively.

Definition 5.2.5 Let $\succ$ be a partial ordering on $\lambda\mathcal{C}$. The *homeomorphic embedding relation* $\rhd_{emb}$ on $\mathcal{T}(\lambda\mathcal{C})$ is defined inductively as follows:
$s = \alpha(s_1,\ldots,s_n) \rhd_{emb} \beta(t_1,\ldots,t_m) = t$ $(arity(\alpha) = n$ and $arity(\beta) = m)$
if and only if

(1) $\alpha \succ \beta$ and there exist indexes $j1,\ldots,jm$ such that $1 \leq j1 < j2 < \ldots < jm \leq n$ and $s_{ji} \trianglerighteq_{emb} t_i$ $(i = 1,\ldots,m)$, or

(2) $s_j \trianglerighteq_{emb} t$ for some j.

Definition 5.2.6 Let $\succ$ be a partial ordering on $\lambda\mathcal{C}$. A partial ordering $>$ is a *simplification ordering* on $\mathcal{T}(\lambda\mathcal{C})$ if it possesses the following three properties:

(1) $s > t$ implies $\alpha(u_1,\ldots,s,\ldots,u_n) > \alpha(u_1,\ldots,t,\ldots,u_n)$ for $\alpha \in \lambda\mathcal{C}$. (the *replacement property*),

(2) $s > s_i$ for any $s_i \in st^m(s)$ (the *subterm property*),

(3) $\alpha(u_1,\ldots,u_n) > \beta(u_{i1},\ldots,u_{im})$ if $\alpha, \beta \in \lambda\mathcal{C}$, $\alpha \succ \beta$, $1 \leq i1 < \ldots < im \leq n$, $arity(\alpha) = n$ and $arity(\beta) = m$.

Lemma 5.2.7 Let $\succ$ be a partial ordering on $\lambda\mathcal{C}$ and $>$ be a simplification ordering on $\mathcal{T}(\lambda\mathcal{C})$. Then, $\rhd_{emb} \subseteq >$ holds.

Proof. We show that $s \rhd_{emb} t$ implies $s > t$ by induction on $\mid s \mid$.

- Basic step: $\mid s \mid = 1$. Since $\alpha, \beta \in \lambda\mathcal{C}$ and $s = \alpha \rhd_{emb} \beta = t$, $\alpha \succ \beta$. Hence, $s = \alpha > \beta = t$ holds.
- Induction step: We consider that $s = \alpha(s_1, \ldots, s_n) \rhd_{emb} \beta(t_1, \ldots, t_m) = t$.
  (1) By induction hypothesis, $s_{ji} \geq t_i$ holds $(i = 1, \ldots, m)$. By replacement property, $\beta(s_{j1}, \ldots, s_{jm}) \geq \beta(t_1, \ldots, t_m)$ holds. Since $\alpha \succ \beta$ and $1 \leq j1 < j2 < \ldots < jm \leq n$, $\alpha(s_1, \ldots, s_m) > \beta(s_{j1}, \ldots, s_{jm})$ holds. Hence, $s = \alpha(s_1, \ldots, s_n) > \beta(t_1, \ldots, t_m) = t$ holds.
  (2) By induction hypothesis, $s_j \geq t$ for some j. By subterm property, $s > s_j$ holds. Hence, $s > t$ holds. □

Remark 5.2.8 If we assume that $\mathcal{S}$ and $\mathcal{C}$ are finite then we can give a well-partial ordering $>$ on $\lambda\mathcal{C}$, i.e., $(\lambda\mathcal{C},>)$ is a well-partially ordered set: See lemma A.1.1 in appendix.

Theorem 5.2.9 Let $\succ$ be a well-partial ordering on $\lambda\mathcal{C}$. Then, a homeomorphic embedding relation $\rhd_{emb}$ is a well-partial ordering on $\mathcal{T}(\lambda\mathcal{C})$.

Proof. It is straightforward by Kruskal's tree theorem [12, 39]. □

Theorem 5.2.10 Let $\succ$ be a well-partial ordering on $\lambda\mathcal{C}$. Then, simplification orderings on $\mathcal{T}(\lambda\mathcal{C})$ are well-founded.

Proof. It is straightforward since lemma 5.2.7 and theorem 5.2.9. □

Note that we assume that there exists at least one constant in $\mathcal{C}$ for each type in this chapter. The following theorem guarantees the termination property for HRSs.

Theorem 5.2.11 Let $\mathcal{R}$ be a HRS on $\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$. Let $\succ$ be a well-partial ordering on $\lambda\mathcal{C}$ and $>$ a simplification ordering on $\mathcal{T}(\lambda\mathcal{C})$ such that $\| l\theta\downarrow \| > \| r\theta\downarrow \|$ for any ground substitution $\theta{:}\mathcal{V}\rightarrow\mathcal{T}(\mathcal{C}\cup\mathcal{B})\downarrow$ and any rewrite rule $l\rightarrow r$ in $\mathcal{R}$. Then $\mathcal{R}$ is terminating.

Proof. Assume that $\mathcal{R}$ is not terminating. There is an infinite rewrite sequence $t_0 \rightarrow_{\mathcal{R}} t_1 \rightarrow_{\mathcal{R}} t_2 \rightarrow_{\mathcal{R}} \ldots$. Without loss of generality, we may assume that terms in this infinite sequence are ground. By the definition of rewriting, $s\rightarrow_{\mathcal{R}} t$ if and only if there exist a rewrite rule $l\rightarrow r\in\mathcal{R}$, a substitution $\theta{:}\mathcal{V}\rightarrow\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$ and a ground context C such that $s=C[l\theta\downarrow]$ and $t=C[r\theta\downarrow]$. $FV(l\theta\downarrow)=\{x_1,\ldots,x_n\}$ where $x_i{:}\sigma_i$ $(1\leq i\leq n)$. Let $\theta'=\{x_1\leftarrow c_{\sigma_1},\ldots,x_n\leftarrow c_{\sigma_n}\}$. Note that $l\theta\theta'\downarrow = l\theta\downarrow\theta'$, $r\theta\theta'\downarrow = r\theta\downarrow\theta'$ and substitution $\theta\theta'{:}\mathcal{V}\rightarrow\mathcal{T}(\mathcal{C}\cup\mathcal{B})\downarrow$ is ground. Since $l\theta\downarrow\theta'$ and $r\theta\downarrow\theta'$ are ground and the assumption, $\| l\theta\downarrow\theta'\| > \| r\theta\downarrow\theta'\|$ holds. By replacement property, $\| C\|[\| l\theta\downarrow\theta'\|] > \| C\|[\| r\theta\downarrow\theta'\|]$. By lemma 5.2.3, it follows that $\| C[l\theta\downarrow]\| = \| C\|[\| l\theta\downarrow\theta'\|]$ and $\| C[r\theta\downarrow]\| = \| C\|[\| r\theta\downarrow\theta'\|]$. Hence, $\| s\| > \| t\|$. Thus we have the infinite sequence $\| t_0\| > \| t_1\| > \| t_2\| > \ldots$: contradiction to the well-foundedness of $>$ by theorem 5.2.10. Therefore, $\mathcal{R}$ is terminating on $\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$. □

5.3 Higher-Order Recursive Path Ordering

In this section, we introduce the RPO for HRSs. First we define the RPO on algebraic terms, called typed recursive path ordering. Next, we define the higher-order recursive path ordering on normalized terms using the interpretation function and typed recursive path ordering.

Definition 5.3.1 (TRPO) Let $>_{\lambda\mathcal{C}}$ be a partial ordering on $\lambda\mathcal{C}$. Let s and t be algebraic terms. The *typed recursive path ordering* (TRPO) on $\mathcal{T}(\lambda\mathcal{C},\mathcal{V})$ is defined as follows:

$s >_{TRPO} t$ if and only if $root(s)\notin\mathcal{V}$ and

(1) $s_i \geq_{TRPO} t$ for some $s_i\in st^m(s)$, or

(2) $root(s) >_{\lambda\mathcal{C}} root(t)$ and $s >_{TRPO} t_i$ for all $t_i\in st^m(t)$, or

(3) $root(s)=root(t)$, $status(root(s))=mult$ and $st^m(s) >_{TRPO}^{mult} st^m(t)$, or

(4) $root(s)=root(t)$, $status(root(s))\neq mult$, $st(s) >_{TRPO}^{status(root(s))} st(t)$ and $s >_{TRPO} t_i$ for all $t_i\in st^m(t)$.

where $>^{mult}$ and $>_{TRPO}^{status(root(s))}$ are the extension of $>_{TRPO}$ associated with the statuses $mult$ and $status(root(s))$, respectively.

We show that the TRPO is a simplification ordering on $\mathcal{T}(\lambda\mathcal{C})$. Note that a partial ordering $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ is given in the following three lemmas.

Lemma 5.3.2 The TRPO is a partial ordering on $\mathcal{T}(\lambda\mathcal{C})$.

Proof. We can show the transitivity and the irreflexivity of TRPO by using the same argument as that for the RPO on first-order terms [9]. □

Lemma 5.3.3 The TRPO has the subterm property on $\mathcal{T}(\lambda\mathcal{C})$, i.e., $s \in \mathcal{T}(\lambda\mathcal{C})$ and $s \rhd t$ imply $s >_{TRPO} t$.

Proof. Let s be a ground algebraic term and t be a proper subterm of s. We show that $s >_{TRPO} t$ by induction on $\mid s \mid$. Let $s = \alpha(s_1,\ldots,s_m)$ $(m \geq 1)$. Since t is a subterm of some s_i, $s_i \geq_{TRPO} t$ holds by induction hypothesis. Hence, $s >_{TRPO} t$ holds by case (1) of definition 5.3.1. □

Lemma 5.3.4 The TRPO has the replacement property on $\mathcal{T}(\lambda\mathcal{C})$, i.e., $s >_{TRPO} t$ implies $\alpha\,(u_1,\ldots,s,\ldots,u_n) >_{TRPO} \alpha\,(u_1,\ldots,t,\ldots,u_n)$ for $\alpha\,(u_1,\ldots,s,\ldots,u_n)$, $\alpha\,(u_1,\ldots,t,\ldots,u_n) \in \mathcal{T}(\lambda\mathcal{C})$.

Proof. If $\alpha : b_1 \to \ldots \to b_n \to b \in \lambda\mathcal{C}$ and $status\ (\alpha) = mult$ then $\alpha\ (u_1,\ldots,s,\ldots,u_n) >_{TRPO} \alpha\,(u_1,\ldots,t,\ldots,u_n)$ holds, since $\{u_1,\ldots,s,\ldots,u_n\} >_{TRPO}^{mult} \{u_1,\ldots,t,\ldots,u_n\}$. If $status\ (\alpha) \neq mult$ then $\alpha\ (u_1,\ \ldots,\ s,\ \ldots,\ u_n) >_{TRPO} \alpha\ (u_1,\ \ldots,\ t,\ \ldots,\ u_n)$ holds, since $\langle u_1,\ \ldots,\ s,\ \ldots,\ u_n\rangle >_{TRPO}^{status(\alpha)} \langle u_1,\ \ldots,\ t,\ \ldots,\ u_n\rangle$ and $\alpha\ (u_1,\ \ldots,\ s,\ \ldots,\ u_n) >_{TRPO} u$ for all $u \in \{u_1,\ \ldots,\ t,\ \ldots,\ u_n\}$. □

Lemma 5.3.5 Let $>_{\lambda\mathcal{C}}$ be a well-partial ordering on $\lambda\mathcal{C}$. If $\alpha >_{\lambda\mathcal{C}} \beta$, then $\alpha(u_1,\ \ldots\ ,u_n) >_{TRPO} \beta(u_{j1},\ \ldots\ ,u_{jm})$ where $\alpha,\ \beta \in \lambda\mathcal{C}$, $1 \leq j1 < \ldots < jm \leq n$, $arity(\alpha) = n$ and $arity(\beta) = m$.

Proof. It is straightforward by the definition of TRPO. □

Theorem 5.3.6 If $>_{\lambda\mathcal{C}}$ is a well-partial ordering on $\lambda\mathcal{C}$ then the TRPO is a simplification ordering on $\mathcal{T}(\lambda\mathcal{C})$.

Proof. By lemmas 5.3.2, 5.3.3, 5.3.4 and 5.3.5, the TRPO is a simplification ordering on $\mathcal{T}(\lambda\mathcal{C})$. □

We define the higher-order recursive path ordering on $\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$ using the interpretation function and the TRPO.

Definition 5.3.7 (HRPO) Let s and t be normalized terms. The *higher-order recursive path ordering* (HRPO) $s >_{HRPO} t$ is defined by $\parallel s \parallel >_{TRPO} \parallel t \parallel$.

Example 5.3.8 We consider the following normalized terms.

$s = X(\lambda xy.x)$ and $t = X(\lambda xy.y)$ where $X{:}(Nat \to Nat \to Nat) \to Nat$, $x{:}Nat$ and $y{:}Nat$. Then, $\parallel s \parallel = X(\lambda_{Nat\to Nat\to Nat}(\lambda_{Nat\to Nat}(c_{Nat}))) = \parallel t \parallel$ holds.

Let $\theta = \{X \leftarrow \lambda z.z(0,1)\}$ where $z{:}Nat \to Nat \to Nat$, $0{:}Nat$ and $1{:}Nat$. Since $s\theta\downarrow = 0$ and $t\theta\downarrow = 1$, $\parallel s\theta\downarrow \parallel \neq \parallel t\theta\downarrow \parallel$. Therefore, it does not hold that $\parallel s \parallel = \parallel t \parallel$ implies $\parallel s\theta\downarrow \parallel = \parallel t\theta\downarrow \parallel$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C}\cup\mathcal{B})\downarrow$ in general.

The above example explains why each bound variable is restricted into a basic type to guarantee the stability of ground substitutions.

Note that we assume that each bound variable in normalized terms has a basic type in the rest of this chapter.

Lemma 5.3.9 Let s and t be normalized terms. Then, $\| s \| = \| t \|$ implies $\| s\theta\downarrow \| = \| t\theta\downarrow \|$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C} \cup \mathcal{B})\downarrow$.

Proof. We show that $\| s \| = \| t \|$ implies $\| s\theta\downarrow \| = \| t\theta\downarrow \|$ by induction on the structure of s. In the case that $s = \alpha(s_1,\cdots,s_m)$ with $\alpha \in \mathcal{C}$ or $s = \lambda x.s_1$, the proof is straightforward. We consider the case that $s = \alpha(s_1,\cdots,s_m)$ with $\alpha \in \mathcal{V}$.

Since $s = \alpha(s_1,\cdots,s_m)$ and $\| s \| = \| t \|$, we have $t = \alpha(t_1,\cdots,t_m)$ such that $\| s_i \| = \| t_i \|$ $(1 \leq i \leq m)$.

Then $s\theta = (\alpha\theta)(s_1\theta,\cdots,s_m\theta)$ and $t\theta = (\alpha\theta)(t_1\theta,\cdots,t_m\theta)$. Letting $s' = \alpha\theta$, we show $\|s'(s_1\theta,\cdots,s_m\theta)\downarrow\| = \|s'(t_1\theta,\cdots,t_m\theta)\downarrow\|$ by induction on the size of s'.

If $s' = \lambda x_1 \cdots x_m.x_i$ then $\| s_i\theta\downarrow \| = \| t_i\theta\downarrow \|$ by induction hypothesis. Let $s' = \lambda x_1 \cdots x_m.f(u_1,\cdots,u_n)$ $(n \geq 0$ and $f \in \mathcal{C})$. Let $\gamma = \{x_1 \leftarrow s_1\theta,\cdots,x_m \leftarrow s_m\theta\}$.

$\|s'(s_1\theta,\cdots,s_m\theta)\downarrow\|$
$= f(\|u_1\gamma\downarrow\|, \cdots, \|u_n\gamma\downarrow\|)$
$= f(\ \|(\lambda\ x_1 \cdots\ x_m.u_1)\ (s_1\theta, \cdots,\ s_m\theta)\downarrow\|, \cdots,\ \|(\lambda\ x_1 \cdots\ x_m.u_n)\ (s_1\theta, \cdots,\ s_m\theta)\downarrow\|)$
$= f(\ \|(\lambda\ x_1 \cdots\ x_m.u_1)\ (t_1\theta, \cdots,\ t_m\theta)\downarrow\|, \cdots,\ \|(\lambda\ x_1 \cdots\ x_m.u_n)\ (t_1\theta, \cdots,\ t_m\theta)\downarrow\|)$
by induction hypothesis. □

Lemma 5.3.10 Let $root(\| s \|) \notin \mathcal{V}$ and $\| t \| \in st^m(\| s \|)$. Then, $\| t\theta\downarrow \| \in st^m(\| s\theta\downarrow \|)$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C} \cup \mathcal{B})\downarrow$.

Proof. We consider the following cases.

(1) $\|\alpha(t_1,\ldots,t_n)\| = \alpha(\|t_1\|,\ldots,\|t_n\|)$ and $\|\alpha(t_1,\ldots,t_n)\theta\downarrow\| = \alpha(\|t_1\theta\downarrow\|,\ldots,\|t_n\theta\downarrow\|)$ for $\alpha \in \mathcal{C} \cup \mathcal{B}$.

(2) $\|\lambda x.s : \tau \to \sigma\| = \lambda_{\sigma\to(\tau\to\sigma)}(\|s\{x \leftarrow c_\tau\}\|)$ and $\|(\lambda x.s)\theta\downarrow : \tau \to \sigma\| = \lambda_{\sigma\to(\tau\to\sigma)}(\|(s\theta\downarrow)\ \{x \leftarrow c_\tau\}\|)$. We may consider the case that $x \in FV(s)$ and $x \notin dom(\theta)$. Since $x \in FV(s)$ and $x \notin dom(\theta)$, $\|(s\theta\downarrow)\ \{x \leftarrow c_\tau\}\ \| = \|(s\ \{x \leftarrow c_\tau\})\theta\downarrow\|$. □

Lemma 5.3.11 The TRPO is stable under ground substitutions, i.e., $\| s \| >_{TRPO} \| t \|$ implies $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ on $\mathcal{T}(\lambda\mathcal{C})$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C} \cup \mathcal{B})\downarrow$.

Proof. Let s and t be normalized terms and θ be a ground substitution. We assume that $\| s \| >_{TRPO} \| t \|$ and show $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ on $\mathcal{T}(\lambda\mathcal{C})$ by induction on $| \| s \| | + | \| t \| |$.

- Basic step: $| \| s \| | + | \| t \| | = 2$.

  If $\| s \| = \alpha$ and $\| t \| = \beta$ $(\alpha, \beta \in \lambda\mathcal{C})$, then $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ holds since $\| s\theta\downarrow \| = \| s \|$ and $\| t\theta\downarrow \| = \| t \|$.

- Induction step:

  (1) $root(\| s \|) \notin \mathcal{V}$ and $\| s_i \| \geq_{TRPO} \| t \|$ for some $\| s_i \| \in st^m(\| s \|)$.
  From induction hypothesis and lemma 5.3.9, we have $\| s_i\theta\downarrow \| \geq_{TRPO} \| t\theta\downarrow \|$. By lemma 5.3.10, $\| s_i\theta\downarrow \| \in st^m(\| s\theta\downarrow \|)$ holds. Since $root(\| s \|) \notin \mathcal{V}$ and $root(\| s\theta\downarrow \|) = root(\| s \|)$, $root(\| s\theta\downarrow \|) \notin \mathcal{V}$ holds. We have $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ by definition of TRPO.

  (2) $root(\| s \|) \notin \mathcal{V}$, $root(\| s \|) >_{\lambda\mathcal{C}} root(\| t \|)$ and $\| s \| >_{TRPO} \| t_i \|$ for any $\| t_i \| \in st^m(\| t \|)$.
  By $root(\| t \|) \in \lambda\mathcal{C}$, $root(\| t\theta\downarrow \|) = root(\| t \|)$ holds. From lemma 5.3.10, we have $\| t_i\theta\downarrow \| \in st^m(\| t\theta\downarrow \|)$. Since $\| s \| >_{TRPO} \| t_i \|$ and induction hypothesis, $\| s\theta\downarrow \| >_{TRPO} \| t_i\theta\downarrow \|$ holds for any $\| t_i\theta\downarrow \| \in st^m(\| t\theta\downarrow \|)$.

  (3) $root(\| s \|) \notin \mathcal{V}$, $root(\| s \|) = root(\| t \|)$ and $st^m(\| s \|) >^{mult}_{TRPO} st^m(\| t \|)$.
  Since $root(\| s \|) = root(\| t \|)$ and $root(\| s \|) \notin \mathcal{V}$, $root(\| s\theta\downarrow \|) = root(\| s \|)$ and $root(\| t\theta\downarrow \|) = root(\| t \|)$ hold. Since induction hypothesis, lemma 5.3.9 and lemma 5.3.10, $st^m(\| s\theta\downarrow \|) >^{mult}_{TRPO} st^m(\| t\theta\downarrow \|)$ holds.

  (4) $root(\| s \|) \notin \mathcal{V}$, $root(\| s \|) = root(\| t \|)$, $st(\| s \|) >^{status(root(\| s \|))}_{TRPO} st(\| t \|)$ and $\| s \| >_{TRPO} \| t_i \|$ for any $\| t_i \| \in st^m(\| t \|)$.
  Since $root(\| s \|) = root(\| t \|)$ and $root(\| s \|) \notin \mathcal{V}$, $root(\| s\theta\downarrow \|) = root(\| s \|)$ and $root(\| t\theta\downarrow \|) = root(\| t \|)$ hold. Since induction hypothesis, lemma 5.3.9 and lemma 5.3.10, $st(\| s\theta\downarrow \|) >^{status(root(\| s \|))}_{TRPO} st(\| t\theta\downarrow \|)$ and $\| s\theta\downarrow \| >_{TRPO} \| t_i\theta\downarrow \|$ for any $\| t_i\theta\downarrow \| \in st^m(\| t\theta\downarrow \|)$ hold. □

Lemma 5.3.12 For any well-founded ordering $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ there exists a well-partial ordering $>^*_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ such that $>_{\lambda\mathcal{C}} \subseteq >^*_{\lambda\mathcal{C}}$ and $>_{TRPO} \subseteq >^*_{TRPO}$.

Proof. By structural induction we can show that if $>_{\lambda\mathcal{C}} \subseteq >^*_{\lambda\mathcal{C}}$ then $>_{TRPO} \subseteq >^*_{TRPO}$. Further we can show that every well-founded ordering is contained in a total well-founded ordering. Since every total well-founded ordering is a well-partial ordering, there exists a well-partial ordering $>^*_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$. □

Theorem 5.3.13 Let $\mathcal{R}$ be a HRS on $\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$. Let $>_{\lambda\mathcal{C}}$ be a well-founded ordering on $\lambda\mathcal{C}$. If for any rewrite rule $l \to r$ in $\mathcal{R}$ we have $l >_{HRPO} r$, then $\mathcal{R}$ is terminating.

Proof. Since lemma 5.3.12 and theorem 5.3.6, there exists a simplification ordering $>^*_{TRPO}$ on $\mathcal{T}(\lambda\mathcal{C})$ such that $>_{TRPO} \subseteq >^*_{TRPO}$. For any $l \to r \in \mathcal{R}$, $\| l\theta\downarrow \| >_{TRPO} \| r\theta\downarrow \|$ holds on $\mathcal{T}(\lambda\mathcal{C})$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C}\cup\mathcal{B})\downarrow$ because of $l >_{HRPO} r$ and lemma 5.3.11. Since $>_{TRPO} \subseteq >^*_{TRPO}$, $\| l\theta\downarrow \| >^*_{TRPO} \| r\theta\downarrow \|$ holds. From theorem 5.2.11, it follows that the HRS $\mathcal{R}$ is terminating. □

Example 5.3.14 We show the termination property of the following HRS $\mathcal{R}$ [38].

$\mathcal{S} = \{term,\ form\}$, $\mathcal{C} = \{\neg{:}form \to form,\ \wedge,\ \vee{:}form \to form \to form,\ \forall,\ \exists{:}(term \to form) \to form\}$, $\{P,\ Q{:}form,\ P'{:}term \to form,\ x{:}term\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \left\{ \begin{array}{l} \neg\neg P \to P \\ \neg(P \wedge Q) \to (\neg P) \vee (\neg Q) \\ \neg(P \vee Q) \to (\neg P) \wedge (\neg Q) \\ \neg\forall(\lambda\, x.P'(x)) \to \exists(\lambda\, x.\neg P'(x)) \\ \neg\exists(\lambda\, x.P'(x)) \to \forall(\lambda\, x.\neg P'(x)). \end{array} \right.$$

We give the precedence $\neg >_{\lambda\mathcal{C}} \vee$, $\neg >_{\lambda\mathcal{C}} \wedge$, $\neg >_{\lambda\mathcal{C}} \exists >_{\lambda\mathcal{C}} \lambda_{form \to (term \to form)}$, $\neg >_{\lambda\mathcal{C}} \forall >_{\lambda\mathcal{C}} \lambda_{form \to (term \to form)}$. Then, the following relations hold.

- $\neg\ \neg\ P >_{HRPO} P$.
- $\neg\ (P \wedge Q) >_{HRPO} (\neg\ P) \vee (\neg\ Q)$.
- $\neg\ (P \vee Q) >_{HRPO} (\neg\ P) \wedge (\neg\ Q)$.
- $\neg\ \forall\ (\lambda\, x.P'(x)) >_{HRPO} \exists\ (\lambda\, x.\neg\ P'(x))$.
- $\neg\ \exists\ (\lambda\, x.P'(x)) >_{HRPO} \forall\ (\lambda\, x.\neg\ P'(x))$.

Hence, $\mathcal{R}$ is terminating by theorem 5.3.13.

Example 5.3.15 We show the termination property of the following HRS $\mathcal{R}$.

$\mathcal{S} = \{Nat,\ List\}$, $\mathcal{C} = \{f{:}Nat \to Nat,\ g{:}(List \to List) \to List \to Nat,\ h{:}List \to Nat,\ \hat{f}{:}List \to List,\ \hat{g}{:}(Nat \to Nat) \to Nat \to List,\ \hat{h}{:}Nat \to List\}$, $\{X{:}List \to List,\ x{:}List,\ Z{:}List,\ Y{:}Nat \to Nat,\ y{:}Nat,\ W{:}Nat\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \left\{ \begin{array}{l} f(g(\lambda\, x.X(x), Z)) \to h(Z) \\ \hat{f}(\hat{g}(\lambda\, y.Y(y), W)) \to \hat{h}(W). \end{array} \right.$$

We give the precedence $f >_{\lambda\mathcal{C}} h$ and $\hat{f} >_{\lambda\mathcal{C}} \hat{h}$. Then, $f(g(\lambda\, x.X(x),Z)) >_{HRPO} h(Z)$ and $\hat{f}(\hat{g}(\lambda\, y.Y(y),W)) >_{HRPO} \hat{h}(W)$ hold. Hence, $\mathcal{R}$ is terminating by theorem 5.3.13.

On the other hand, Jouannaud and Rubio's ordering [27, 28] cannot deal with this example. Let $\gtrsim_{\mathcal{T}_\mathcal{S}}$ be a quasi-ordering on $\mathcal{T}_\mathcal{S}$. A normalized term $s : \sigma$ is *type compatible* if $t : \tau$ is type compatible and $\sigma \gtrsim_{\mathcal{T}_\mathcal{S}} \tau$ for any proper subterm $t : \tau$ of s. Since their ordering works on only type compatible terms and $f(g(\lambda\, x.X(x), Z)){:}Nat \rhd \lambda\, x.X(x){:}List \to List$, we have $Nat \gtrsim_{\mathcal{T}_\mathcal{S}} List \to List$ when the quasi-ordering $\gtrsim_{\mathcal{T}_\mathcal{S}}$ is a RPO on $\mathcal{T}_\mathcal{S}$. Hence, we have $Nat \gtrsim_\mathcal{S} List$ and $List \not\gtrsim_\mathcal{S} Nat$. Since $\hat{f}(\hat{g}(\lambda\, y.Y(y), W)){:}List \rhd \lambda\, y.Y(y){:}Nat \to Nat$ and the type compatibility, it is obtained that $List \gtrsim_{\mathcal{T}_\mathcal{S}} Nat \to Nat$. Hence, we have $List \gtrsim_\mathcal{S} Nat$ and $Nat \not\gtrsim_\mathcal{S} List$. This is contradiction. Thus termination of $\mathcal{R}$ on type compatible normalized terms cannot be shown by their ordering when $\gtrsim_{\mathcal{T}_\mathcal{S}}$ is a RPO on $\mathcal{T}_\mathcal{S}$.

5.4 Envelope for Typed Terms

The envelope G_τ is a subset of $\lambda\mathcal{C}$ such that every constant symbol occurring in ground algebraic terms with the type τ is contained in G_τ, i.e, $Fun(s) \subseteq G_\tau$ for any ground algebraic term $s{:}\tau$. Let $X{:}\sigma_1 \to \ldots \to \sigma_n \to b$ be a variable with an output type b and let θ a ground substitution such that $X\theta = \lambda x_1 \ldots x_n.t{:}\sigma_1 \to \ldots \to \sigma_n \to b \in \mathcal{T}(\mathcal{C})\downarrow$. Then, for any ground normalized terms $s_1{:}\sigma_1, \ldots, s_n{:}\sigma_n$, we have $Fun(\|X\theta(s_1,\ldots,s_n)\downarrow\|) \subseteq G_b$. This property allows us to treat the variable X as a constant symbol with the output type b in the TRPO under appropriate conditions about G_b. In fact, if we can give a precedence $>_{\lambda\mathcal{C}}$ satisfying $\alpha >_{\lambda\mathcal{C}} \beta$ for all $\beta \in G_b$, then $\alpha(t_1,\ldots,t_n) >_{TRPO} \|X\theta(s_1,\ldots,s_n)\downarrow\|$ follows from the definition of the TRPO. This means that X plays in the TRPO like a constant with $\alpha >_{\lambda\mathcal{C}} X$.

First, we define the envelope and the precedence between constants and free variables as follows:

Definition 5.4.1 We define the *envelope* G_τ ($\subseteq \lambda\mathcal{C}$) inductively as follows:

(1)
- $f{:}\tau_1 \to \ldots \to \tau_l \to \tau \in \mathcal{C}$ implies $f \in G_\tau$.
- $\lambda_{\sigma\to(\tau\to\sigma)}, c_\tau \in G_{\tau\to\sigma}$.

(2)
- $g{:}\sigma_1 \to \ldots \to \sigma_n \to \sigma \in G_\tau$ and $g \in \mathcal{C} \cup \mathcal{B}$ imply $G_{\sigma_1} \cup \ldots \cup G_{\sigma_n} \subseteq G_\tau$.
- $\lambda_{\sigma\to(\eta\to\sigma)} \in G_\tau$ implies $G_\sigma \subseteq G_\tau$.

Example 5.4.2 We consider the following signature: $\mathcal{C} = \{0{:}Nat,\ s{:}Nat \to Nat,\ +{:}Nat \to Nat \to Nat,\ nil{:}List,\ cons{:}Nat \to List \to List,\ map{:}(Nat \to Nat) \to List \to List\}$. Then, $G_{Nat} = \{0,\ s,\ +\}$ and $G_{List} = \{nil,\ cons,\ map,\ \lambda_{Nat\to(Nat\to Nat)},\ c_{Nat},\ 0,\ s,\ +\}$.

Lemma 5.4.3 Let s be a ground algebraic term with a type τ. Then, $Fun(s) \subseteq G_\tau$ holds.

Proof. It is straightforward by the definitions of the interpretation function and algebraic terms. □

By using envelopes we can extend the precedence $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ to that on $\lambda\mathcal{C} \cup \mathcal{V}$ as follows. Let $\alpha \in \lambda\mathcal{C}$ and let Y be a free variable with an output type b. Then we define $\alpha >_{\lambda\mathcal{C}} Y$ if $\alpha >_{\lambda\mathcal{C}} \beta$ for any $\beta \in G_b$. We say the *TRPO* (*HRPO*) *with envelopes* shortly when the TRPO is based on this extended precedence $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C} \cup \mathcal{V}$.

Lemma 5.4.4 Let $s = \alpha(s_1,\ldots,s_n)$ ($n \geq 0$) be an algebraic term and X a free variable with an output type b. If $\alpha >_{\lambda\mathcal{C}} X$ then $s >_{TRPO} t$ for any ground algebraic term $t{:}b$.

Proof. Trivial from the definitions of $\alpha >_{\lambda\mathcal{C}} X$ and of the TRPO. □

We next show that the TRPO with envelopes is stable under ground substitutions.

Lemma 5.4.5 The TRPO with envelopes is stable under ground substitutions, i.e., $\| s \| >_{TRPO} \| t \|$ with envelopes implies $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ on $\mathcal{T}(\lambda\mathcal{C})$ for any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C} \cup \mathcal{B})\downarrow$.

Proof. Let s and t be normalized terms and θ be a ground substitution. We assume that $\| s \| >_{TRPO} \| t \|$ with envelopes and show $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ on $\mathcal{T}(\lambda\mathcal{C})$ by induction on $| \| s \| | + | \| t \| |$.

- Basic step: $| \| s \| | + | \| t \| | = 2$.

  If $\| s \| = \alpha$ and $\| t \| = \beta$ ($\alpha, \beta \in \lambda\mathcal{C}$), then $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ holds since $\| s\theta\downarrow \| = \| s \|$ and $\| t\theta\downarrow \| = \| t \|$.

  If $\| s \| = \alpha$ and $\| t \| = X$ with an output type b ($\alpha \in \lambda\mathcal{C}$, $X \in \mathcal{V}$), then $\alpha >_{\lambda\mathcal{C}} X$. From $\| X\theta\downarrow \|{:}b$ and lemma 5.4.4 it follows that $\| s\theta\downarrow \| = \alpha >_{TRPO} \| X\theta\downarrow \| = \| t\theta\downarrow \|$.

- Induction step: The cases (1), (3) and (4) in the definition of TRPO are straightforward by the proof of lemma 5.3.11. We only show the case (2) in TRPO.

  Case (2) $root(\| s \|) \notin \mathcal{V}$, $root(\| s \|) >_{\lambda\mathcal{C}} root(\| t \|)$ and $\| s \| >_{TRPO} \| t_i \|$ for any $\| t_i \| \in st^m(\| t \|)$.

  If $root(\| t \|) \notin \mathcal{V}$, the claim follows by the same argument as that for the case (2) of lemma 5.3.11. Thus, we assume $root(\| t \|) \in \mathcal{V}$. Let $root(\| s \|) = \alpha$ and $t = X(t_1,\ldots,t_n)$ where $X{:}\sigma_1 \to \ldots \to \sigma_n \to b$. Then for any ground algebraic term $\| t\theta\downarrow \|{:}b$, we have $\| s\theta\downarrow \| >_{TRPO} \| t\theta\downarrow \|$ because of $root(\| s\theta\downarrow \|) = root(\| s \|) = \alpha$ and lemma 5.4.4. □

Lemma 5.4.6 For any well-founded ordering $>_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ there exists a well-partial ordering $>^*_{\lambda\mathcal{C}}$ on $\lambda\mathcal{C}$ such that $>_{\lambda\mathcal{C}} \subseteq >^*_{\lambda\mathcal{C}}$ and $>_{TRPO} \subseteq >^*_{TRPO}$ with envelopes.

Proof. It is straightforward by the same argument of lemma 5.3.12. □

From now on we restrict $l\theta\downarrow$ as a ground term in the reduction $C[l\theta\downarrow] \to_{\mathcal{R}} C[r\theta\downarrow]$ in the rest of this chapter.

Theorem 5.4.7 Let $\mathcal{R}$ be a HRS on $\mathcal{T}(\mathcal{C},\mathcal{V})\downarrow$. Let $>_{\lambda\mathcal{C}}$ be a well-founded ordering on $\lambda\mathcal{C}$. If for any rewrite rule $l \to r$ in $\mathcal{R}$ we have $l >_{HRPO} r$ with envelopes, then $\mathcal{R}$ is terminating.

Proof. Since theorem 5.3.6 and lemma 5.4.6, there exists a simplification ordering $>^*_{TRPO}$ on $\mathcal{T}(\lambda\mathcal{C})$. For any $l \to r \in \mathcal{R}$ and any ground substitution $\theta{:}\mathcal{V} \to \mathcal{T}(\mathcal{C} \cup \mathcal{B})\downarrow$, $\| l\theta\downarrow \| >_{TRPO} \| r\theta\downarrow \|$ holds on $\mathcal{T}(\lambda\mathcal{C})$ since $l >_{HRPO} r$ and lemma 5.4.5. Since $>_{TRPO} \subseteq >^*_{TRPO}$ with envelopes, $\| l\theta\downarrow \| >^*_{TRPO} \| r\theta\downarrow \|$ holds. From theorem 5.2.11, it follows that the HRS $\mathcal{R}$ is terminating. □

The following example explains how to apply the HRPO with envelopes to prove termination.

Example 5.4.8 Consider the following basic types, signature and HRS $\mathcal{R}$: $\mathcal{S} = \{Nat, List\}$, $\mathcal{C} = \{nil{:}List, cons{:}Nat \to List \to List, map{:}(Nat \to Nat) \to List \to List, 0{:}Nat, s{:}Nat \to Nat, +{:}Nat \to Nat \to Nat\}$, $\{X{:}Nat \to Nat, N{:}Nat, L{:}List, x{:}Nat\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \begin{cases} map(\lambda\, x.X(x), nil) \to nil \\ map(\lambda\, x.X(x), cons(N, L)) \to cons(X(N), map(\lambda\, x.X(x), L)). \end{cases}$$

To prove the termination property of $\mathcal{R}$ we use the precedence: $map >_{\lambda\mathcal{C}} cons$, 0, s, $+$ and $status(map) = mult$. Since it is obvious that $map(\lambda x.X(x), nil) >_{HRPO} nil$, we consider the second rule. From definition 5.4.1 we can obtain $G_{Nat} = \{0, s, +\}$. Since $map >_{\lambda\mathcal{C}} f$ for any $f \in G_{Nat}$, we have $map >_{\lambda\mathcal{C}} X$. Hence, $map\ (\lambda\, x.X(x), cons\ (N, L)) >_{HRPO} X(N)$ holds. Thus $map\ (\lambda\, x.X(x), cons\ (N, L)) >_{HRPO} cons\ (X(N), map\ (\lambda\, x.X(x), L))$ follows. Therefore, HRS $\mathcal{R}$ is terminating by theorem 5.4.7.

5.5 Related Works

In this section, we compare our HRPO with Jouannaud and Rubio's ordering [27, 28]. We first consider the following HRS $\mathcal{R}$.

$\mathcal{S} = \{Nat, List\}$, $\mathcal{C} = \{app{:}(Nat \to Nat) \to Nat \to Nat, list{:}Nat \to List, twice{:}Nat \to List, cons{:}Nat \to List \to List, nil{:}List, 0{:}Nat, s{:}Nat \to Nat, +{:}Nat \to Nat \to Nat\}$, $\{X{:}Nat \to Nat, Y{:}Nat, x{:}Nat\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \begin{cases} list(app(\lambda\, x.X(x), Y)) \to twice(X(Y)) \\ twice(Y) \to cons(Y, cons(Y, nil)) \\ list(Y) \to cons(Y, nil). \end{cases}$$

We first show termination of $\mathcal{R}$ by applying our ordering with envelopes. From the definition of envelopes we have $G_{Nat} = \{app, 0, s, +, \lambda_{Nat \to (Nat \to Nat)}, c_{Nat}\}$. Give the following precedence: $list >_{\lambda\mathcal{C}} twice >_{\lambda\mathcal{C}} cons$, nil and $list >_{\lambda\mathcal{C}} X$, i.e., $list >_{\lambda\mathcal{C}} \alpha$ for any $\alpha \in G_{Nat}$. Then, $list(app(\lambda\, x.X(x), Y)) >_{HRPO} twice(X(Y))$ with envelopes. From theorem 5.4.7 it follows that HRS $\mathcal{R}$ is terminating.

Jouannaud and Rubio's ordering [27, 28] cannot be applied to the above $\mathcal{R}$ because of the type compatible term limitation. Let $\gtrsim_{\mathcal{T}_\mathcal{S}}$ be a RPO on $\mathcal{T}_\mathcal{S}$. In this case, $list(app(\lambda\, x.X(x), Y))$ is not a type compatible normalized term. More precisely, app

$(\lambda\, x.X(x)$,$Y)$:$Nat \rhd \lambda\, x.X(x)$:$Nat \to Nat$ and $Nat \not\gtrsim_{\mathcal{T}_\mathcal{S}} Nat \to Nat$. Thus termination of $\mathcal{R}$ cannot be proven by their ordering.

Next, we consider the following HRS $\mathcal{R}$:

$\mathcal{S} = \{b, b'\}$. $\mathcal{C} = \{f{:}(b' \to b') \to b,\ g{:}b' \to b,\ d{:}b',\ e{:}b' \to b',\ h{:}(b \to b') \to b'\}$, $\{X{:}b' \to b',\ Y{:}b \to b',\ x{:}b',\ y{:}b\} \subseteq \mathcal{V}$ and

$$\mathcal{R} = \left\{ \begin{array}{l} f(\lambda\, x.X(x)) \to g(X(d)) \\ e(h(\lambda\, y.Y(y))) \to d. \end{array} \right.$$

Our ordering can prove termination of $\mathcal{R}$. We have $G_{b'} = \{d,e,h,\lambda_{b' \to (b \to b')},c_b\}$. Give the following precedence: $f >_{\lambda\mathcal{C}} g$, d and $e >_{\lambda\mathcal{C}} d$ and $f >_{\lambda\mathcal{C}} X$, i.e., $f >_{\lambda\mathcal{C}} \alpha$ for any $\alpha \in G_{b'}$. Then, $f(\lambda\, x.X(x)) >_{HRPO} g(X(d))$ holds. Hence, HRS $\mathcal{R}$ is terminating.

Jouannaud and Rubio's ordering [27, 28] again cannot prove termination of $\mathcal{R}$. Let $\gtrsim_{\mathcal{T}_\mathcal{S}}$ be a RPO on $\mathcal{T}_\mathcal{S}$. Since $e(h(\lambda\, y.Y(y))){:}b'$, $\lambda\, y.Y(y){:}b \to b'$ and $b' \not\gtrsim_{\mathcal{T}_\mathcal{S}} b \to b'$, $e(h(\lambda\, y.Y(y)))$ is not type compatible.

Jouannaud and Rubio [27, 28] proposed the *sort ordering* on $\mathcal{T}_\mathcal{S}$ as the other quasi-ordering $\gtrsim_{\mathcal{T}_\mathcal{S}}$. However, the sort ordering $\gtrsim_{\mathcal{T}_\mathcal{S}}$ does not work for this example. We have to show $f(\lambda\, x.X(x)){:}b > X(d){:}b'$ in the first rule of $\mathcal{R}$. Thus we have $b >_\mathcal{S} b'$. Then $e(h(\lambda\, y.Y(y)))$ is not type compatible for the sort ordering, because $e(h(\lambda\, y.Y(y))){:}b'$ and $\lambda\, y.Y(y){:}b \to b'$.

Jouannaud and Rubio [28] have showed that for some non-type compatible terms, termination is also assured by termination of HRS on type compatible terms. However, their results cannot be applied in above examples since quasi-orderings $\gtrsim_{\mathcal{T}_\mathcal{S}}$ are not consistent with type structure.

5.6 Conclusion

We have introduced a natural framework of a simplification ordering for analyzing termination of higher-order rewriting, and based on this framework we have proposed a powerful RPO on higher-order terms, called the higher-order recursive path ordering (HRPO). Our ordering has extended Jouannaud and Rubio's ordering, which does not allow comparing two type incompatible terms. We have shown through several examples that our ordering can be applied to prove termination of HRSs to which Jouannaud and Rubio's ordering cannot be applied. We believe that our ordering provides very useful means of proving termination which arises in various higher-order formal systems like higher-order functional and logic programming languages and higher-order theorem provers.

Chapter 6

Persistence of Termination for Term Rewriting Systems on Ordered Sorts

6.1 Introduction

Many-sorted TRSs are HRSs without higher-order functions or bound variables. Further usual (first-order) TRSs are special cases of many-sorted TRSs. In order to study the basic property of termination of HRSs, we try to analyze the termination of TRSs using the notion of persistence. Zantema [66] introduced the notion of *persistence* as follows. A property P of TRSs is *persistent* if for any many-sorted TRS $\mathcal{R}$, $\mathcal{R}$ has the property P if and only if its underlying TRS $\Theta(\mathcal{R})$, which results from $\mathcal{R}$ by omitting its sort information, has the property P. Usual many-sorted TRS was extended with ordered sorts by Aoto and Toyama [3]. And it was shown that the persistency of confluence [2] is preserved for this extension in [3]. Zantema [66] showed that termination is persistent for TRSs without collapsing or duplicating rules. Ohsaki and Middeldorp [46] studied the persistence of termination, acyclicity and non-loopingness on equational many-sorted TRSs. In this chapter, we show that the above Zantema's result is preserved for Aoto and Toyama's extension in the subclass of order sorted term rewriting systems. This research was first appeared in [22]. Further, Ohsaki [47] studied the more general case of equational order-sorted TRSs.

Given an arbitrary STRS $\mathcal{R}$, by identifying each sort with $*$, we obtain an unsorted TRS $\Theta(\mathcal{R})$ called the underlying TRS of $\mathcal{R}$, i.e., $\Theta(\mathcal{R})$ is obtained by omitting sort information of $\mathcal{R}$. In this case $f \in \mathcal{F}_n$ is regarded as an abbreviation for $f : \underbrace{* \times \ldots \times *}_{n} \to *$.

The following definition of persistence was given by Zantema [66].

Definition 6.1.1 A property P of TRSs is said to be *persistent* if for any many-sorted TRS $\mathcal{R}$, $\mathcal{R}$ has the property P if and only if its underlying TRS $\Theta(\mathcal{R})$ has the property P.

Also, Zantema showed that termination is not persistent for TRSs in general but it is persistent for the particular class of TRSs.

Theorem 6.1.2 ([66]) Termination is persistent for TRSs without collapsing or duplicating rules.

6.2 Sorting of Term Rewriting Systems

Aoto and Toyama [2, 3] defined the notion of *sort attachment* and formulated the notion of persistence using sort attachment. In this section we introduce the sort attachment and persistence defined by Aoto and Toyama [3]. We mainly follow basic definitions and basic lemmas in Aoto and Toyama [3] in this chapter.

Let $\mathcal{S}$ be a set of sorts with well-founded partial ordering $\succ$ on it. $\mathcal{F}$ is a set of arity fixed function symbols.

A *sort attachment* μ of $\mathcal{F}$ on $\mathcal{S}$ is a family of mapping $\cup_n(\mathcal{F}_n \to \mathcal{S}^{n+1}) \cup (\mathcal{V} \to \mathcal{S})$.

We can assume that there are countably infinite variables x with $\mu(x) = b$ for each $b \in \mathcal{S}$. $f \in \mathcal{F}$ with $\mu(f) = (b_1,\ldots,b_n,b')$ is written as $f{:}b_1 \times \ldots \times b_n \to b'$. The set of *$\mu$-sorted function symbols* from $\mathcal{F}$ and that of *μ-sorted variables* from $\mathcal{V}$ are denoted by $\mathcal{F}^\mu$ and $\mathcal{V}^\mu$, respectively.

A term t is said to be *well-sorted* under μ with sort b (written as $\mu(t) = b$) if and only if $t \in \mathcal{T}(\mathcal{F}^\mu, \mathcal{V}^\mu)^b$. A term t is said to be *strict well-sorted* under μ with sort b if and only if $t \in \mathcal{T}(\mathcal{F}^\mu, \mathcal{V}^\mu)^b$ is strict. *Well-sorted contexts* are defined by special constants $\Box^{b_1}, \ldots$. Well-sorted terms and contexts are often treated as sorted terms and contexts, respectively.

Definition 6.2.1 Let $(\mathcal{F}, \mathcal{R})$ be a TRS. A sort attachment μ of $\mathcal{F}$ on $\mathcal{S}$ is said to be *consistent* with $\mathcal{R}$ if and only if for any rewrite rule $l \to r \in \mathcal{R}$, l and r are strictly well-sorted under μ and $\mu(l) \succeq \mu(r)$. The *set of μ-sorted rewrite rules* of $\mathcal{R}$ is denoted by $\mathcal{R}^\mu$.

It is obvious that for given $(\mathcal{F}, \mathcal{R})$, a sort attachment μ of $\mathcal{F}$ on $\mathcal{S}$ that is consistent with $\mathcal{R}$ uniquely induces a STRS $(\mathcal{F}^\mu, \mathcal{R}^\mu)$ called the μ-sorted STRS of $(\mathcal{F}, \mathcal{R})$.

It is trivial that the following lemma holds.

Lemma 6.2.2 Let $\mathcal{R}$ be a TRS and μ a sort attachment that is consistent with $\mathcal{R}$. For any term s which is well-sorted under μ, if $s \to_\mathcal{R} t$ then t is well-sorted under μ and $s \to_{\mathcal{R}^\mu} t$.

Using the sort attachment, persistence can be alternatively formulated as follows. It is clear that definition 6.1.1 and the following definition are equivalent when set of sorts with empty relation on it. Aoto and Toyama [3] introduced the following definition.

Definition 6.2.3 A property P of TRSs is *persistent* if and only if for any TRS $(\mathcal{F}, \mathcal{R})$ and any sort attachment μ that is consistent with $\mathcal{R}$,

$(\mathcal{F}^\mu, \mathcal{R}^\mu)$ has property P $\Leftrightarrow$ $(\mathcal{F}, \mathcal{R})$ has property P.

We consider the persistence of termination for TRSs on ordered sorts using definition 6.2.3 in this chapter instead of Zantema's definition. We assume that a set $\mathcal{S}$ of sorts and a TRS $\mathcal{R}$ are given and that an attachment μ on $\mathcal{S}$ that is consistent with $\mathcal{R}$ is fixed.

6.3 Characterization of Unsorted Terms

In this section we give a characterization of unsorted terms by ordered sorts. The proofs of the following basic lemmas were given by Aoto and Toyama [3].

Definition 6.3.1 The *top sort* (under μ) of an unsorted term t is defined by

$$top(t) = \begin{cases} \mu(t) & \text{if } t \in \mathcal{V} \\ b' & \text{if } t = f(t_1, \ldots, t_n) \text{ with } f : b_1 \times \ldots \times b_n \to b' \end{cases}$$

Definition 6.3.2 Let $t = C[t_1, \ldots, t_n]$ $(n \geq 0)$ be an unsorted terms with $C[, \ldots,] \neq \Box$. We write $t = C[\![t_1, \ldots, t_n]\!]$ if and only if

1. $C{:}b_1 \times \ldots \times b_n \to b'$ is a context that is well-sorted under μ.
2. $top(t_i) \not\preceq b_i$ for $i = 1, \ldots, n$.

The t_1,...,t_n are said to be the *principal subterms* of t.

We denote $t = C\,\langle\langle\, t_1, \ldots, t_n \,\rangle\rangle$ if either $t = C[\![t_1, \ldots, t_n]\!]$ or $C = \Box$ and $t \in \{t_1, \ldots, t_n\}$. Multiset $S(t)$ consists of all principal subterms of $t = C[\![t_1, \ldots, t_n]\!]$.

Definition 6.3.3 Let t be an unsorted term. *Rank* of t is defined by

$$rank(t) = \begin{cases} 1 & \text{if } t \text{ is well-sorted term} \\ 1 + max\{rank(t_1), \ldots, rank(t_n)\} & \text{if } t = C[\![t_1, \ldots, t_n]\!] \end{cases}$$

Definition 6.3.4 Let t be an unsorted term. *Cap* of t is defined by

$$cap(t) = \begin{cases} t & \text{if } t \text{ is well-sorted term} \\ C[, \ldots,] & \text{if } t = C[\![t_1, \ldots, t_n]\!] \end{cases}$$

Definition 6.3.5 A rewrite step $s \to_{\mathcal{R}} t$ is said to be *inner* (written as $s \to^i_{\mathcal{R}} t$) if and only if $s = C\,[\![s_1,\ldots,C'\,[l\theta],\ldots,s_n]\!] \to_{\mathcal{R}} C\,[\![t_1,\ldots,C'\,[r\theta],\ldots,t_n]\!] = t$ for some $s_1,\ldots,s_n$, $l \to r \in \mathcal{R}$, θ and C', otherwise *outer* (written as $s \to^o_{\mathcal{R}} t$).

Definition 6.3.6 A rewrite step $s \rightarrow^o_{\mathcal{R}} t$ is said to be *vanishing* if and only if $s = C\ [\![s_1,\ldots,\theta(x),\ldots,s_n]\!] \rightarrow^o_{\mathcal{R}} \theta(x) = t$ for some $s_1,\ldots,s_n$, C and θ such that $C'\theta = C\ [s_1,\ldots,\Box,\ldots,s_n]$ and $C'[x] \rightarrow x \in \mathcal{R}$.

Definition 6.3.7 A rewrite rule $l \rightarrow r$ in $\mathcal{R}$ is said to be *decreasing* if and only if $top(l) \succ top(r)$. A rewrite step $s \rightarrow^o_{\mathcal{R}} t$ is said to be *decreasing* if and only if $top(s) \succ top(t)$.

Definition 6.3.8 A rewrite step $s \rightarrow^o_{\mathcal{R}} t$ is said to be *destructive at level 1* if and only if $s \rightarrow^o t$ is either vanishing or decreasing. The rewrite step $s \rightarrow_{\mathcal{R}} t$ is said to be *destructive at level $k+1$* if and only if $C[\![s_1,\ldots,s_j,\ldots,s_n]\!] \rightarrow^i_{\mathcal{R}} C[s_1,\ldots,t_j,\ldots,s_n] = t$ with $s_j \rightarrow_{\mathcal{R}} t_j$ *destructive at level k* .

Lemma 6.3.9 The following statements hold.

1. If a rewrite step $s \rightarrow^o_{\mathcal{R}} t$ is not vanishing then $top(s) \succeq top(t)$.
2. If a rewrite step $s \rightarrow^o_{\mathcal{R}} t$ is not destructive at level 1 then $top(s) = top(t)$.

Lemma 6.3.10 The following statements hold.

1. If $s \rightarrow_{\mathcal{R}} t$ then $rank(s) \geq rank(t)$.
2. If a rewrite step $s \rightarrow_{\mathcal{R}} t$ is vanishing then $rank(s) > rank(t)$.

Definition 6.3.11 The *grade* ($grade(t) \in N \times \mathcal{S}$) of a term t is defined by $grade(t) = \langle rank(t), top(t)\rangle$ where N is the set of all natural numbers.

Let $>$ be the lexicographic ordering on $N \times \mathcal{S}$ induced from $>$ on N and $\succ$ on $\mathcal{S}$. The lexicographic ordering $>$ on $N \times \mathcal{S}$ is well-founded since orderings $>$ on N and $\succ$ on $\mathcal{S}$ are well-founded.

Lemma 6.3.12 The following statements hold.

1. If $s \rightarrow_{\mathcal{R}} t$ then $grade(s) \geq grade(t)$.
2. If a rewrite step $s \rightarrow_{\mathcal{R}} t$ is destructive at level 1 then $grade(s) > grade(t)$.

6.4 Persistence of Termination

In this section we discuss the persistence of termination for TRSs on ordered sorts and the examples as applications. We mainly follow the Ohlebusch's argument in [45].

Lemma 6.4.1 Let t be a strictly well-sorted term. Suppose $t = C[x_1, \ldots, x_n]$ with all variables displayed and $C : b_1 \times \ldots \times b_n \to b'$. Then for any substitution θ, if $x_i = x_j$ and $\theta(x_i)$ is a principal subterm of $t\theta$ then $\theta(x_j)$ is also a principal subterm of $t\theta$.

Let $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ be terms. We write $\langle\, s_1, \ldots, s_n \,\rangle \propto \langle\, t_1, \ldots, t_n \,\rangle$ if and only if for any $1 \leq i, j \leq n$, $s_i = s_j$ implies $t_i = t_j$.

Lemma 6.4.2 If $C[\![s_1, \ldots, s_n]\!] \to^o_{\mathcal{R}} C' \langle\, \langle\, s_{i_1}, \ldots, s_{i_m} \,\rangle \,\rangle$ and $\langle\, s_1, \ldots, s_n \,\rangle \propto \langle\, t_1, \ldots, t_n \,\rangle$ then $C[t_1, \ldots, t_n] \to^o_{\mathcal{R}} C'[t_{i_1}, \ldots, t_{i_m}]$.

Lemma 6.4.3 If $s \to^o_{\mathcal{R}} t$ is a non-destructive rewrite step and applied rewrite rule is not duplicating, the multiset inclusion $S(t) \subseteq S(s)$ holds.

Lemma 6.4.4

1. If $s = C[\![s_1, \ldots, s_j, \ldots, s_n]\!] \to^i_{\mathcal{R}} C[s_1, \ldots, t_j, \ldots, s_n] = t$ is destructive at level 2 and $cap(s) \neq cap(t)$ then $S(t) = S(s) \setminus \{s_j\} \cup S(t_j)$.
2. If $s = C[\![s_1, \ldots, s_j, \ldots, s_n]\!] \to^i_{\mathcal{R}} C[s_1, \ldots, t_j, \ldots, s_n] = t$ is destructive at level 2 and $cap(s) = cap(t)$ then $S(t) = S(s) \setminus \{s_j\} \cup \{t_j\}$.

Lemma 6.4.5 The following statements hold.

1. If $s \to^o_{\mathcal{R}} t$ is not destructive at level 1 then $cap(s) \to_{\mathcal{R}^\mu} cap(t)$.
2. If $s \to^i_{\mathcal{R}} t$ is not destructive at level 2 then $cap(s) = cap(t)$.

Proof.

1. Let $s = C[\![s_1, \ldots, s_n]\!] \to^o_{\mathcal{R}} C' \langle\, \langle\, s_{i_1}, \ldots, s_{i_m} \,\rangle \,\rangle = t$. Since $\langle\, s_1, \ldots, s_n \,\rangle \propto \langle\, \Box, \ldots, \Box \,\rangle$ and lemma 6.4.2, $cap(s) = C[, \ldots,] \to_{\mathcal{R}^\mu} C'[, \ldots,] = cap(t)$.
2. Let $s = C[\![s_1, \ldots, s_j, \ldots, s_n]\!] \to^i_{\mathcal{R}} C[\![s_1, \ldots, t_j, \ldots, s_n]\!] = t$ and $s_j \to_{\mathcal{R}} t_j$. Since $cap(s) = C[, \ldots,]$ and $cap(t) = C[, \ldots,]$, it is trivial. □

Definition 6.4.6 For any infinite rewrite sequence $D : s_0 \to_{\mathcal{R}} s_1 \to_{\mathcal{R}} s_2 \to_{\mathcal{R}} \ldots$, we define the *grade* of D to be $grade(D) = grade(s_0)$.

Definition 6.4.7 Let $>$ is the lexicographic ordering on $N \times \mathcal{S}$ and $>^{mult}$ is the multiset extension of it. We define $\sharp s = \{grade(t) \mid t \in S(s)\}$, i.e., $\sharp s$ denote the multiset of the grades of the principal subterms of s.

Multiset ordering $>^{mult}$ is well-founded since lexicographic ordering $>$ on $N \times \mathcal{S}$ is well-founded [10].

Lemma 6.4.8 If $s \rightarrow^{i}_{\mathcal{R}} t$ is destructive at level 2 then $\sharp s >^{mult} \sharp t$.

Proof. Let s and t be $C[\![s_1, \ldots, s_j, \ldots, s_n]\!]$ and $C[s_1, \ldots, t_j, \ldots, s_n]$, respectively.

1. $cap(s) \neq cap(t)$. By lemma 6.4.4, $S(t) = S(s) \setminus \{s_j\} \cup S(t_j)$.

   - $s_j \rightarrow_{\mathcal{R}} t_j$ is vanishing. Since $rank(s_j) > rank(t_j)$ and $rank(t_j) > rank(u)$ for any $u \in S(t_j)$, $rank(s_j) > rank(u)$ holds. So $grade(s_j) > grade(u)$ for any $u \in S(t_j)$ holds. Hence, $\sharp s >^{mult} \sharp t$ holds.
   - $s_j \rightarrow_{\mathcal{R}} t_j$ is decreasing. Since $rank(s_j) \geq rank(t_j)$ and $rank(t_j) > rank(u)$ for any $u \in S(t_j)$, $rank(s_j) > rank(u)$ holds. So $grade(s_j) > grade(u)$ for any $u \in S(t_j)$ holds. Hence, $\sharp s >^{mult} \sharp t$ holds.

2. $cap(s) = cap(t)$. By lemma 6.4.4, $S(t) = S(s) \setminus \{s_j\} \cup \{t_j\}$.

   - $s_j \rightarrow t_j$ is vanishing. Since $rank(s_j) > rank(t_j)$, $grade(s_j) > grade(t_j)$ holds.
   - $s_j \rightarrow t_j$ is decreasing. Since $rank(s_j) \geq rank(t_j)$ and $top(s_j) \succ top(t_j)$, $grade(s_j) > grade(t_j)$ holds.

Therefore, $\sharp s >^{mult} \sharp t$ holds. □

Lemma 6.4.9 Let $\mathcal{R}^{\mu}$ be a terminating STRS. Let $D : s_0 \rightarrow_{\mathcal{R}} s_1 \rightarrow_{\mathcal{R}} s_2 \rightarrow_{\mathcal{R}} \ldots$ be an infinite rewrite sequence of minimal grade with respect to $\mathcal{R}$. Then the following statements hold.

1. There are infinitely many outer rewrite steps in D.
2. There are infinitely many inner rewrite steps in D which are destructive at level 2.
3. There are infinitely many duplicating outer rewrite steps in D.

Proof. Since $grade(s_j) = grade(D)$ for any $j \in N$, there is no rewrite step which is destructive at level 1.

1. Suppose that there are only finitely many outer rewrite steps in D. Then we can assume that there is no outer rewrite step in D. If $s_0 = C[\![t_1, \ldots, t_n]\!]$ then there must be an infinite rewrite sequence starting from some $t_i \in S(s_0)$. Since $rank(t_i) < rank(s_0)$ this contradicts the minimality of $grade(D)$.

2. Suppose that there are only finitely many inner rewrite steps in D which are destructive at level 2. Then we can assume that there is no inner rewrite step in D which is destructive at level 2. By lemma 6.4.5, if $s \to^o_{\mathcal{R}} t$ in D then $cap(s) \to_{\mathcal{R}^\mu} cap(t)$ and if $s \to^i_{\mathcal{R}} t$ in D then $cap(s) = cap(t)$. By the case 1, R^μ is not terminating. This is contradiction by the assumption.

3. Suppose that there are only finitely many outer rewrite steps which are applied duplicating rules in D. We consider the following cases.

   - If $s_j \to^o_{\mathcal{R}} s_{j+1}$ then $S(s_{j+1}) \subseteq S(s_j)$ since the rewrite step is non-destructive and non-duplicating and lemma 6.4.3. Then $\sharp s_j \geq^{mult} \sharp s_{j+1}$ holds.
   - If $s_j \to^i_{\mathcal{R}} s_{j+1}$ is not destructive at level 2 then we have $s_j = C[\![t_1, \ldots, t_k, \ldots, t_n]\!] \to^i_{\mathcal{R}} C[\![t_1, \ldots, t'_k, \ldots, t_n]\!] = s_{j+1}$ where $t_k \to_{\mathcal{R}} t'_k$. Then $\sharp s_j \geq^{mult} \sharp s_{j+1}$ holds since $grade(t_k) \geq grade(t'_k)$.
   - If $s_j \to^i_{\mathcal{R}} s_{j+1}$ is destructive at level 2 then we have $s_j = C[\![t_1, \ldots, t_k, \ldots, t_n]\!] \to^i_{\mathcal{R}} C[t_1, \ldots, t'_k, \ldots, t_n] = s_{j+1}$ where $t_k \to_{\mathcal{R}} t'_k$ is destructive at level 1. By lemma 6.4.8, $\sharp s_j >^{mult} \sharp s_{j+1}$. By the well-foundedness of $>^{mult}$, there are only finitely many inner rewrite steps which are destructive at level 2 in D. This contradicts the case 2. □

Lemma 6.4.10 Let $\mathcal{R}^\mu$ be a terminating STRS. Assume that $\mathcal{R}$ is not terminating. Then $\mathcal{R}$ has duplicating rules and $\mathcal{R}$ has collapsing or decreasing rules.

Proof. By lemma 6.4.9, it is trivial. □

Theorem 6.4.11 The following statements hold.

1. Termination is persistent for TRSs on ordered sorts without collapsing or decreasing rules.
2. Termination is persistent for TRSs on ordered sorts without duplicating rules.

Proof. By lemma 6.4.10, it is trivial. □

In the following, we give the examples as applications of theorem 6.4.11. To show the termination of the following TRSs directly seems difficult from known results. The persistence of termination is useful to show termination of a given TRS from those of its subsystems.

Example 6.4.12 We show that the following TRS $\mathcal{R}$ is terminating using theorem 6.4.11.

$$\mathcal{R} = \begin{cases} f(x, f(x, A)) \to G(x, x) & (r1) \\ f(x, G(x, B)) \to f(x, G(x, C)) & (r2) \\ f(x, f(x, y)) \to y & (r3) \\ A \to B & (r4) \\ F(G(C, x), x) \to F(G(A, x), x) & (r5) \\ G(C, C) \to C & (r6) \end{cases}$$

Let $\mathcal{S} = \{0, 1, 2\}$, $1 \succ 0$ and $2 \succ 0$.

$$\mu = \begin{cases} f : 0 \times 1 \to 1 \\ F : 0 \times 0 \to 2 \\ A, B, C : 0 \\ G : 0 \times 0 \to 0 \end{cases}$$

Any term in $\mathcal{T}^0$, $\mathcal{T}^1$ or $\mathcal{T}^2$ is terminating, i.e., any term in $\mathcal{T}$ is terminating.

Since rules applicable to terms in $\mathcal{T}^0$ are $(r4)$ and $(r6)$, any term in $\mathcal{T}^0$ is terminating. Since rules applicable to terms in $\mathcal{T}^1$ are $(r1)$, $(r2)$, $(r3)$, $(r4)$ and $(r6)$, any term in $\mathcal{T}^1$ is terminating. Since rules applicable to terms in $\mathcal{T}^2$ are $(r4)$, $(r5)$ and $(r6)$, any term in $\mathcal{T}^2$ is terminating. Termination of the subsystems $\{(r4), (r6)\}$, $\{(r1), (r2), (r3), (r4), (r6)\}$ and $\{(r4), (r5), (r6)\}$ can be shown using RPO.

Then $\mathcal{R}^\mu$ is terminating. Since $\mathcal{R}$ has no duplicating rules and theorem 6.4.11, $\mathcal{R}$ is terminating.

Example 6.4.13 Further, we show that the following TRS $\mathcal{R}$ is terminating using theorem 6.4.11.

$$\mathcal{R} = \begin{cases} x \supset F \to \neg x \vee F & (r1) \\ x \supset (x \wedge T) \to x \supset (x \wedge \neg\neg F) & (r2) \\ x \otimes (x \vee F) \to x \otimes (x \vee \neg\neg T) & (r3) \\ x \wedge T \to x & (r4) \\ x \vee F \to x & (r5) \\ T \to TRUE & (r6) \\ F \to FALSE & (r7) \end{cases}$$

Let $\mathcal{S} = \{0, 1, 2\}$, $1 \succ 0$ and $2 \succ 0$.

$$\mu = \begin{cases} \supset : 0 \times 0 \to 1 \\ \otimes : 0 \times 0 \to 2 \\ \wedge, \vee : 0 \times 0 \to 0 \\ \neg : 0 \to 0 \\ T, F, TRUE, FALSE : 0 \end{cases}$$

Any term in $\mathcal{T}^0$, $\mathcal{T}^1$ or $\mathcal{T}^2$ is terminating, i.e., any term in $\mathcal{T}$ is terminating.

Since rules applicable to terms in $\mathcal{T}^0$ are $(r4)$, $(r5)$, $(r6)$ and $(r7)$, any term in $\mathcal{T}^0$ is terminating. Since rules applicable to terms in $\mathcal{T}^1$ are $(r1)$, $(r2)$, $(r4)$, $(r5)$, $(r6)$ and $(r7)$, any term in $\mathcal{T}^1$ is terminating. Since rules applicable to terms in $\mathcal{T}^2$ are $(r3)$, $(r4)$, $(r5)$, $(r6)$ and $(r7)$, any term in $\mathcal{T}^2$ is terminating. Termination of the subsystems $\{(r4), (r5), (r6), (r7)\}$, $\{(r1), (r2), (r4), (r5), (r6), (r7)\}$ and $\{(r3), (r4), (r5), (r6), (r7)\}$ can be shown using RPO.

Then $\mathcal{R}^\mu$ is terminating. Since $\mathcal{R}$ has no duplicating rules and theorem 6.4.11, $\mathcal{R}$ is terminating.

6.5 Related Works

It is possible that theorem 6.4.11 is extended to equational rewriting. An equational rewriting $s \rightarrow_{\mathcal{R}/\mathcal{E}} t$ if and only if there are terms u, v such that $s =_{\mathcal{E}} u \rightarrow_{\mathcal{R}} v =_{\mathcal{E}} t$ where $\mathcal{E}$ is an equational system [13]. After our research in [22], Ohsaki [47] studied the persistence of termination for order-sorted equational TRSs. The following theorem in equational order-sorted TRSs is more general case of theorem 6.4.11.

Theorem 6.5.1 ([47]) Termination is persistent for $\mathcal{S}$-sorted equational TRS $\mathcal{R}/\mathcal{E}$ such that $\mathcal{E}$ is a non-collapsing, variable-preserving, non-decreasing equational system and $\mathcal{R}$ is either a non-collapsing and non-decreasing TRS or a non-duplicating TRS.

The following theorem was given as an application of theorem 6.5.1 in [47].

Theorem 6.5.2 ([47]) Let $(\mathcal{F} \cup \mathcal{G}, \mathcal{R}_1)$ be a non-duplicating TRS and let $(\mathcal{G}, \mathcal{R}_2)$ be a non-duplicating confluent TRS. If $\mathcal{R}_1$ and $\mathcal{R}_2$ are terminating and for any rewrite rule $l \rightarrow r$ in $\mathcal{R}_1$,

(1) $root(l) \notin \mathcal{G}$ and $root(l') \notin Fun(l) \cup Fun(r)$ for any $l' \rightarrow r' \in \mathcal{R}_2$,

(2) Every subterm s of l (and r) satisfies $root(s) \in \mathcal{F}$ or $s \in \mathcal{T}(\mathcal{G}, \mathcal{V})$ and

(3) Every subterm $f(s_1, \ldots, s_n)$ of l and subterm $g(t_1, \ldots, t_m)$ of l or r such that $s_i = t_j$ and $s_i, t_j \in \mathcal{V}$ imply $f, g \in \mathcal{F}$ or $f, g \in \mathcal{G}$,

then $(\mathcal{F} \cup \mathcal{G}, \mathcal{R}_1 \cup \mathcal{R}_2)$ is terminating.

6.6 Conclusion

In this chapter, we have shown that the Zantema's result [66] is preserved for Aoto and Toyama's extension [3] in the subclass of order sorted term rewriting systems. That is, we have shown that termination is persistent for TRSs on ordered sorts without collapsing, decreasing or duplicating rules. Further, we have given some examples as applications of our results. Usual TRSs are special cases of HRSs. In order to study the basic property of termination of HRSs, we have tried to analyze the termination of usual TRSs using the notion of persistence.

Chapter 7

Conclusions

In this thesis, we have discussed the termination of higher-order rewrite systems by syntactic approaches. We summarize the main results in this thesis.

(1) We have extended the improved recursive decomposition ordering to higher-order rewrite systems for proving termination. Our extension method is inspired by Jouannaud and Rubio's idea [27, 28] and the particular properties of improved recursive decomposition ordering. First, we have introduced the typed improved recursive decomposition ordering on algebraic terms. In order to obtain the stability under ground substitutions, we have introduced the notion of pseudo-terminal positions. Next, we defined the higher-order improved recursive decomposition ordering based on the typed improved recursive decomposition ordering and interpretation function from normalized terms to algebraic terms. Further, we have compared our higher-order improved recursive decomposition ordering with the Jouannaud and Rubio's ordering [27, 28]. The improved recursive decomposition ordering is more powerful than recursive path ordering in the frame of term rewriting systems. The similar result holds in the frame of higher-order rewrite systems. We have shown that our ordering is more powerful than the ordering defined by Jouannaud and Rubio.

(2) We have introduced a natural framework of a simplification ordering for analyzing termination of higher-order rewriting, and based on this framework we have proposed a powerful recursive path ordering on higher-order terms, called the higher-order recursive path ordering. Though our approach has been inspired by the first-order interpretation method described in Jouannaud and Rubio [27, 28], we have developed the higher-order recursive path ordering within a more natural framework of a simplification ordering. Our key idea in higher-order recursive path ordering is the concept of envelopes for typed terms that allows us to treat higher-order variables as function symbols. We clarify that the priority between type and function symbol introduced in Jouannaud and Rubio [27, 28] is not essential and any partial ordering on function symbols can be used freely to define the higher-order recursive path ordering. Thus we could remove the type compatible term limitation in Jouannaud and Rubio's ordering. We have shown through several examples that our ordering can be applied to prove termination of higher-order rewrite systems to which Jouannaud and Rubio's ordering cannot be applied. We believe that our ordering provides very useful means of proving termination which arises in various higher-order formal systems like higher-order functional and logic programming

languages and higher-order theorem provers.

(3) Many-sorted TRSs are HRSs without higher-order functions or bound variables. Further usual TRSs are special cases of many-sorted TRSs. In order to study the basic property of termination of higher-order rewrite systems, we have tried to analyze the termination of term rewriting systems using the notion of persistence. Zantema [66] showed that termination is persistent for TRSs without collapsing or duplicating rules. Usual many-sorted TRS was extended with ordered sorts by Aoto and Toyama [3]. We have shown that the above Zantema's result is preserved for Aoto and Toyama's extension in the subclass of order sorted term rewriting systems. That is, we have shown that termination is persistent for term rewriting systems on ordered sorts without collapsing, decreasing or duplicating rules. Further, we have given some examples as applications of our results.

Appendix A

Property of Well-Partial Ordering

A.1 Lemma A.1.1

Lemma A.1.1 We can give a well-partial ordering $>$ on $\lambda\mathcal{C}$, i.e., $(\lambda\mathcal{C},>)$ is a well-partially ordered set.

Proof. $\lambda\mathcal{C} = \mathcal{C} \cup \mathcal{L} \cup \mathcal{B}$ where $\mathcal{C}$ is a finite set.

(1) As $\mathcal{C}$ is finite, $(\mathcal{C}, \emptyset)$ is a well-partially ordered set where $\emptyset$ is the empty relation.

(2) We show that a well-partial ordering $>_\lambda$ can be defined on $\mathcal{L}$. Let $\mathcal{S}$ be $\{b_1,\ldots,b_n\}$ where b_i is a basic type $(i = 1,\ldots,n)$. The total precedence on basic types and the constructor $\rightarrow$ is given by $b_0 < b_1 < \ldots < b_n < \rightarrow$. Then, for any τ, $\sigma \in \mathcal{T}_\mathcal{S}$, either $\tau >_{LPO} \sigma$ or $\tau <_{LPO} \sigma$ $(\tau \neq \sigma)$ holds where $>_{LPO}$ is the *lexicographic path ordering* [4] on $\mathcal{T}_\mathcal{S}$. Since $>_{LPO}$ is total and well-founded, it is a well-partial ordering. Hence, $(\mathcal{T}_\mathcal{S},>_{LPO})$ is a well-partially ordered set. The partial ordering $\lambda_\sigma >_\lambda \lambda_\tau$ is defined by $\sigma >_{LPO} \tau$. Thus, $(\mathcal{L},>_\lambda)$ is a well-partially ordered set.

(3) By the same argument as that in case (2), we can give a well-partial ordering $c_\sigma >_c c_\tau$ on $\mathcal{B}$ by $\sigma >_{LPO} \tau$. Thus, $(\mathcal{B},>_c)$ is a well-partially ordered set.

Therefore, $(\lambda\mathcal{C},>)$ is a well-partially ordered set where $> = >_\lambda \cup >_c$ by cases (1), (2) and (3). □

Bibliography

[1] P. B. Andrews, A. Issar, D. Nesmith and F. Pfenning, "The TPS theorem proving system," Proc. 10th International Conf. on Automated Deduction, LNCS, 449, pp.614–642, 1990.

[2] T. Aoto and Y. Toyama, "Persistency of confluence," J. Universal Computer Science, 3, pp.1134–1147, 1997.

[3] T. Aoto and Y. Toyama, "Extending persistency of confluence with ordered sorts," Research report, IS-RR-96-0025F, School of Information Science, JAIST, 1996.

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

[5] A. Ben-Cherifa and P. Lescanne, "Termination of rewriting systems by polynomial interpretations and its implementation," Science of Computing Programming, 9(2), pp.137–159, 1987.

[6] M. Broy, "Equational specification of partial higher order algebras," Theoretical Computer Science, 57, pp.3–45, 1988.

[7] R. Constable, S. Allen and H. Bromly, "Implementing mathematics with Nuprl proof development system," Prentice-Hall, New Jersey, 1986.

[8] M. Dauchet, "Simulation of Turing machines by a regular rewrite rules," Theoretical Computer Science, 103, pp.109–120, 1992.

[9] N. Dershowitz, "Orderings for term-rewriting systems," Theoretical Computer Science, 17, pp.279–301, 1982.

[10] N. Dershowitz and Z. Manna, "Proving termination with multiset orderings," Commun. ACM, 22 (8), pp.465–476, 1979.

[11] N. Dershowitz and J. P. Jouannaud, "Rewrite systems," In Handbook of Theoretical Computer Science, vol.B, ed. J.van Leeuwen, pp.243–320, The MIT Press/Elsevier, 1990.

[12] N. Dershowitz, "Termination of rewriting," J. Symbolic Computation, 3, pp.69–116, 1987.

[13] M. C. F. Ferreira, "Termination of term rewriting: well-foundedness, totality and transformations," Ph.D. thesis, Utrecht University, 1995.

[14] J. H. Gallier, "What's so special about Kruskal's theorem and the ordinal Γ_0 ? A survey of some results in proof theory," Annals of Pure and Applied Logic, 53, pp.199–260, 1991.

[15] J. H. Goldfarb, "The undecidability of the second-order unification problem," Theoretical Computer Science, 13, pp.225–230, 1981.

[16] J. R. Hindley and J. P. Seldin, "Introduction to combinators and λ-calculus," Cambridge University Press.

[17] P. Hudak, S. P. Jones and P. Walder, "Report on the programming language Haskell: A non-strict, purely functional language," ACM SIGPLAN Notices, 27(5), 1997.

[18] G. Huet and D. Lankford, "On the uniform halting problem for term rewriting systems," Report 238, INRIA, 1978.

[19] M. Iwami, M. Sakai and Y. Toyama, "On the termination of higher order rewrite systems," Technical Report of IEICE, COMP95-85, pp.113–121, 1996 (in Japanese).

[20] M. Iwami, "Termination of higher-order rewrite systems," Master thesis, School of Information Science, JAIST, 1996 (in Japanese).

[21] M. Iwami, M. Sakai and Y. Toyama, "An improved recursive decomposition ordering for higher-order rewrite systems," Technical Report of IEICE, COMP96-73, pp.17–24, 1997.

[22] M. Iwami and Y. Toyama, "On the persistency of termination of term rewriting systems with ordered sorts," Proc. 14th Conf. on Japan Society for Software Science and Technology, pp.357–360, 1997 (in Japanese).

[23] M. Iwami, M. Sakai and Y. Toyama, "An improved recursive decomposition ordering for higher-order rewrite systems," IEICE Transactions on Information and Systems, E81-D, pp.988–996, 1998.

[24] M. Iwami and Y. Toyama, "Simplification ordering for higher-order rewrite systems," Research Report, IS-RR-98-0024F, School of Information Science, JAIST, 1998.

[25] M. Iwami and Y. Toyama, "Simplification ordering for higher-order rewrite systems," To appear in IPSJ Transactions on Programming.

[26] J. P. Jouannaud, P. Lescanne and F. Reinig, "Recursive decomposition ordering," Working Conf. on Formal Description of Programming Concepts 2 (IFIP), North-Holland Publishing Company, pp.331–353, 1982.

[27] J. P. Jouannaud and A. Rubio, "A recursive path ordering for higher-order terms in η-long β-normal form," Proc. 7th International Conf. on Rewriting Techniques and Applications, LNCS, 1103, pp.108–122, 1996.

[28] 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, 208, pp.33–58, 1998.

[29] S. Kahrs, "Towards a domain theory of termination proofs," Proc. 6th International Conf. on Rewriting Techniques and Applications, LNCS, 914, pp.241– 255, 1995.

[30] S. Kamin and J. J. Levy, "Attempts for generalizing the recursive path orderings," University of Illinois, 1980.

[31] S. Kapur, P. Narendran and G. Sivakumar, "A path ordering for proving termination of term rewriting systems," Proc. 10th Colloquium on Trees in Algebra and Programming, LNCS, 185, pp.173–187, 1985.

[32] J. W. Klop, "Term rewriting systems," In Handbook of Logic in Computer Science, vol.2, pp.1–112, ed. S. Abramsky, D. Gabbay and T. Mabiaum, Oxford University Press, 1992.

[33] J. W. Klop, "Combinatory reduction systems," Ph.D. thesis, Mathematical Centre Tracts 127, CWI, Amsterdam, 1980.

[34] J. B. Kruskal, "Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture," Transactions of the American Mathematical Society, 95, pp.210–225, 1960.

[35] D. S. Lankford, On proving term rewriting systems are noetherian," Technical Report MTP-3, Louisiana Technical University, Ruston, 1979.

[36] C. Loría-Sáenz and J. Steinbach, "Termination of combined (rewrite and λ-calculus) systems, " Proc. 3rd International Conf. on Rewriting Techniques and Applications, LNCS, 656, pp.143–147, 1992.

[37] O. Lysne and J. Piris, "A termination ordering for higher order rewrite systems," Proc. 6th International Conf. on Rewriting Techniques and Applications, LNCS, 914, pp.26–40, 1995.

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

[39] A. Middeldorp and H. Zantema, "Simple termination of rewrite systems," Theoretical Computer Science, 175, pp.127–158, 1997.

[40] D. Miller, "A logic programming language with lambda-abstraction, function variables, and simple unification," J. Logic and Computation, 1 (4), pp.497–536, 1991.

[41] R. Milner, M. Tofte and R. Harper "The definition of standard ML," MIT Press, 1990.

[42] B. Möller, "Algebraic specification with higher-order operators," IFIPTC2 Working Conf. on Program Specification and Transformation, pp.367–392, 1986.

[43] T. Nipkow, "Higher-order critical pairs," Proc. IEEE Symp. on Logic in Computer Science, pp.342–349, 1991.

[44] T. Nipkow, "Orthogonal higher-order rewrite systems are confluent," Proc. 1st International Conf. on Typed Lambda Calculi and Applications, LNCS, 664, pp.306–317, 1993.

[45] E. Ohlebusch, "A simple proof of sufficient conditions for the termination of the disjoint union of term rewriting systems," Bullen of the EATCS, 49, pp.178–183, 1993.

[46] H. Ohsaki and A. Middeldorp, "Type introduction for equational rewriting," Proc. 4th Symp. on Logical Foundations of Computer Science, LNCS, 1234, pp.283–293, 1997.

[47] H. Ohsaki, "Termination of term rewriting systems: transformation and persistence," Ph.D. thesis, Tsukuba University, 1998.

[48] V. van Oostrom, "Confluence for abstract and higher-order rewriting," Ph.D. thesis, Vrije Universiteit, 1994.

[49] L. C. Paulson, "Isabelle: The next 700 theorem provers," Proc. IEEE Symp. on Logic in Computer Science, pp.361–385, 1990.

[50] L. C. Paulson, "ML for the working programmer," Cambridge University Press, 1991.

[51] L. C. Paulson, "Isabelle: A generic theorem prover," LNCS, 828, 1994.

[52] D. A. Plaisted, "A recursively defined ordering for proving termination of term rewriting systems," Technical Report, UIUCDS-R-78-943, Department of Computer Science, University of Illinois at Uranbana-Champaign, 1978.

[53] D. A. Plaisted, "Well-founded orderings for proving termination of rewrite rules," Technical Report, UIUCDS-R-78-932, Department of Computer Science, University of Illinois at Uranbana-Champaign, 1978.

[54] J. van de Pol, "Termination proofs of higher-order rewrite systems," Proc. 1st International Workshop on Higher-Order Algebra, Logic and Rewriting, LNCS, 816, pp.305–325, 1993.

[55] J. van de Pol, "Termination of higher-order rewrite systems," Ph.D. thesis, Utrecht University, 1996.

[56] C. Prehofer, "Solving higher-order equations from logic to programming," Progress in theoretical computer science, Birkhäuser, 1998.

[57] C. Prehofer, "Solving higher-order equations from logic to programming," Ph.D. thesis, Technische Universität München, 1995.

[58] F. van Raamsdonk, "Confluence and normalisation for higher-order rewriting," Ph.D. thesis, Vrije Universiteit, 1996.

[59] M. Rusinowitch, "Path of subterms ordering and recursive decomposition ordering revisited," J. Symbolic Computation, 3, pp.117–131, 1987.

[60] J. Steinbach, "Termination of rewriting-extension, comparison and automatic generation of simplification orderings," Ph.D. Thesis, University of Kaiserslautern, 1994.

[61] J. Steinbach, "Term orderings with status," SEKI Report SR-88-12, University of Kaiserslautern, 1988.

[62] J. Steinbach, "Extensions and comparison of simplification orderings," Proc. 3rd International Conf. on Rewriting Techniques and Applications, LNCS, 335, pp.434–448, 1989.

[63] J. Steinbach, "Simplification ordering: history of results," Fundamenta Informaticae, 24, pp.44–87, 1995.

[64] D. Walukiewicz, "A total AC-reduction ordering on higher-order terms," Proc. 25th International Conf. on Automata, Languages and Programming, LNCS, 1443, pp.530–542, 1998.

[65] D. A. Wolfram, "The clausal theory of types," Cambridge University Press, 1993.

[66] H. Zantema, "Termination of term rewriting: interpretation and type elimination," J. Symbolic Computation, 17, pp.23–50, 1994.

Publications

[1] M. Iwami, M. Sakai and Y. Toyama, "Termination of higher-order rewrite systems," Proc. the Joint Conf. of Hokuriku Chapters of Institutes of Electrical Engineers, Japan, E-28, pp.314, 1995 (in Japanese).

[2] M. Iwami, M. Sakai and Y. Toyama, "On the termination of higher order rewrite systems," Technical Report of IEICE, COMP95-85, pp.113–122, 1996 (in Japanese).

[3] M. Iwami, M. Sakai and Y. Toyama, "Termination of higher-order rewrite systems," Proc. LA Symp., Summer, pp.55–60, 1996 (in Japanese).

[4] M. Iwami, M. Sakai and Y. Toyama, "An improved recursive decomposition ordering for higher-order rewrite systems," Technical Report of IEICE, COMP96-73, pp.17–24, 1997.

[5] M. Iwami and Y. Toyama, "On the persistency of termination of term rewriting systems with ordered sorts," Proc. 14th Conf. on Japan Society for Software Science and Technology, pp.357–360, 1997 (in Japanese).

[6] M. Iwami, M. Sakai and Y. Toyama, "An improved recursive decomposition ordering for higher-order rewrite systems," IEICE Transactions on Information and Systems, E81-D, pp.988–996, 1998.

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

[8] M. Iwami and Y. Toyama, "Simplification ordering for higher-order rewrite systems," To appear in IPSJ Transactions on Programming.