Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems

by

Keiichirou KUSAKARI

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 2000

Copyright © 2000 by Keiichirou KUSAKARI

Abstract

Recently, Arts and Giesl introduced the notion of dependency pairs, which gives effective methods for proving termination of term rewriting systems (TRSs). In this thesis, we extend the notion of dependency pairs to AC-TRSs, and introduce new methods for effectively proving AC-termination. Since it is impossible to directly apply the notion of dependency pairs to AC-TRSs, we introduce the head parts in terms and show an analogy between the root positions in infinite reduction sequences by TRSs and the head positions in those by AC-TRSs. Indeed, this analogy is essential for the extension of dependency pairs to AC-TRSs. Based on this analogy, we define AC-dependency pairs.

To simplify the task of proving termination and AC-termination, several elimination transformations such as the dummy elimination, the distribution elimination, the general dummy elimination and the improved general dummy elimination, have been proposed. In this thesis, we show that the argument filtering method combined with the AC-dependency pair technique is essential in all the elimination transformations above. We present remarkable simple proofs for the soundness of these elimination transformations based on this observation. Moreover, we propose a new elimination transformation, called the argument filtering transformation, which is not only more powerful than all the other elimination transformations but also especially useful to make clear an essential relationship among them.

Acknowledgments

I wish to express my sincere gratitude to my principal advisor Professor Yoshihito Toyama of Japan Advanced Institute of Science and Technology for his constant encouragement and kind guidance during this work. I would like to thank my advisor Associate Professor Masahiko Sakai of Nagoya University for his helpful discussions and suggestions. I would like to thank my advisor Associate Professor Aart Middeldorp of University of Tsukuba, Professor Hiroakira Ono and Associate Professor Hajime Ishihara of Japan Advanced Institute of Science and Technology for their useful comments and suggestions.

I am grateful to Assistant Professor Claude Marché for his helpful discussions. I also wish to express my thanks to Research Associate Taro Suzuki, Research Associate Munehiro Iwami and Mr. Masaki Nakamura for their valuable discussions, suggestions and continuous encouragements.

I devote my sincere thanks and appreciation to all of them, and my colleagues.

Contents

Abstract

Acknowledgments

1 Introduction **1**

2 Preliminaries **5**
2.1 Binary Relations . . . 5
2.1.1 Binary Relations and Orders . . . 5
2.1.2 Multiset Extension . . . 6
2.1.3 Lexicographic Extension . . . 9
2.2 Term Rewriting Systems . . . 9
2.3 AC-Term Rewriting Systems . . . 10
2.4 Reduction Order and AC-Reduction Order . . . 11
2.4.1 Polynomial Interpretation . . . 11
2.4.2 Simplification Order . . . 12

3 Dependency Pairs **15**
3.1 Dependency Pair and Dependency Chain . . . 15
3.2 Proving Termination by Dependency Pairs . . . 16
3.2.1 Weak Reduction Order . . . 16
3.2.2 Weak Reduction Pair . . . 18
3.2.3 Argument Filtering Method . . . 19
3.2.4 Polynomial Interpretation . . . 21
3.3 Hierarchy of Dependency Pairs . . . 22
3.4 Dependency Graph . . . 24

4 AC-Dependency Pairs **28**
4.1 AC-Dependency Pair and AC-Dependency Chain . . . 29
4.1.1 Unmarked AC-Dependency Pairs and Chains . . . 29
4.1.2 AC-Dependency Pairs and Chains . . . 32
4.1.3 Another AC-dependency Pair . . . 33
4.2 Proving AC-Termination by AC-Dependency Pairs . . . 34
4.2.1 Weak AC-Reduction Order . . . 34
4.2.2 Weak AC-Reduction Pair . . . 35
4.2.3 Argument Filtering Method . . . 36
4.2.4 Lexicographic Argument Filtering Method . . . 40
4.2.5 AC-Multisets Extension . . . 42

4.2.6 Polynomial Interpretation 47
4.3 AC-Dependency Graph 47

5 Argument Filtering Transformation **52**
5.1 Soundness Condition for Transformation 52
5.2 Argument Filtering Transformation 54
5.3 Comparison with Other Eliminations 56
5.3.1 Dummy Elimination 56
5.3.2 Distribution Elimination 57
5.3.3 General Dummy Elimination 60
5.3.4 Improved General Dummy Elimination 62
5.4 Comparison with Argument Filtering Method 64

6 Conclusion **66**

References **67**

Publications **72**

Chapter 1

Introduction

Computability (Church's Thesis): The notion of computable function was essentially born in the proof of Gödel's Incompleteness Theorems [24] and was formalized by Church, Kleene, Turing and others in the 1930's. Intuitively, a computable function is a function whose values can be calculated in some kind of automatic or effective way. It is well known that there exist functions that are not computable. From the point of view of computer science, this fact means that there exist functions that we can not program. As rapid advances of computer technology, the study of computability greatly gains its importance in various fields.

In order to give a rigorously mathematical definition of the intuitive notion of computable function, several computation models were proposed in 1936: recursive functions by Kleene [25, 36, 37][1], λ-calculus by Church [10], and Turing machine by Turing [65]. It is well-known that these computation models characterize the same class for computable function[2]. On the basis of these evidence, most mathematicians have accepted the claim that these characterizations give a satisfactory formalization for computable function, though these definitions of computable function have no generality separated from a particular computation model. This claim is often referred to as Church's Thesis.

(AC-)Term Rewriting Systems: Term rewriting systems (TRSs) can be regarded as a computation model, that is, all computable functions are definable by term rewriting systems. In term rewriting systems, terms are reduced by using a set of directed equations, called rewrite rules. The most striking feature is that term rewriting systems themselves can be regarded as functional programming languages. For example, in term rewriting systems with the constant 0 and the successor function s, addition of natural numbers is defined as follows:

$$R = \left\{ \begin{array}{rcl} x + 0 & \to & x \\ x + s(y) & \to & s(x + y) \end{array} \right.$$

[1]The notion of recursive functions was proposed by Gödel in 1934 based on a suggestion by Herbrand [25]. After that Kleene improved the notion by introducing a minimalization operator, called μ-operator, and formalized the current form in 1936 [36, 37].

[2]Kleene proved the equivalence between recursive functions and computable functions by λ-calculus [38], and Turing proved the equivalence between computable functions by λ-calculus and by Turing machine [66].

By applying rules in this term rewriting system, we can evaluate the sum of 1 and 2 as follows:

$$\underline{s(0) + s(s(0))} \underset{R}{\rightarrow} s(\underline{s(0) + s(0)}) \underset{R}{\rightarrow} s(s(\underline{s(0) + 0})) \underset{R}{\rightarrow} s(s(s(0)))$$

Term rewriting systems have various applications in many fields of computer science and mathematics, that is, to represent abstract interpreters of functional programming languages and to model formal manipulating systems used in various applications, such as program optimization, program verification and automatic theorem proving [6, 18, 28, 39].

The effectiveness of term rewriting systems to theorem proving is worth mentioning. In fact, the field of term rewriting got a decisive impact by the pioneering paper by Knuth and Bendix, in which they designed completion procedures, called Knuth-Bendix procedure, that automatically solves word problems in universal algebra [40]. For example, on Knuth-Bendix procedure, starting with the axiom of group

$$\begin{cases} 0 + x &= x \\ (-x) + x &= 0 \\ (x + y) + z &= x + (y + z) \end{cases}$$

produces a terminating and confluence term rewriting system as follows:

$$\begin{cases} 0 + x &\rightarrow x \\ (-x) + x &\rightarrow 0 \\ (x + y) + z &\rightarrow x + (y + z) \\ (-x) + (x + y) &\rightarrow y \\ x + 0 &\rightarrow x \\ -0 &\rightarrow 0 \\ -(-x) &\rightarrow x \\ x + (-x) &\rightarrow 0 \\ x + ((-x) + y) &\rightarrow y \\ -(x + y) &\rightarrow (-y) + (-x) \end{cases}$$

In a confluent term rewriting system, the answer of a given term is unique, that is, the final result does not depend on computation procedures. Moreover, if every computation procedure always terminates, the system solves the word problem for the corresponding equational theory. Hence, using the obtained term rewriting system, we can automatically solve the word problems for group theory.

Knuth-Bendix procedure has the limitation that can not handle a commutativity equation, say, $x + y = y + x$, because the term rewriting system $\{x + y \rightarrow y + x\}$ induced from the equation is essentially not terminating. To avoid this difficulty, associative-commutative term rewriting systems (AC-TRSs) were introduced, in which associative and commutative equations are not represented as rewrite rules, instead we take them into account when applying some other rewrite rules [48, 49, 60].

Proving (AC-)Termination Termination of TRSs is in general an undecidable property. Nevertheless, it is often necessary to prove the termination for a particular system. For example, the termination property is essentially important in Knuth-Bendix procedure. To prove termination, we commonly design a reduction order by which all rules are ordered.

The most important study for designing a reduction order is the notion of simplification order introduced by Dershowitz [15]. Based on the notion several reduction orders are introduced. The recursive path order was introduced by Dershowitz [15, 16]. The lexicographic path order was introduced by Kamin and Lévy [32]. The recursive decomposition order was introduced by Jouannaud, Lescanne and Reinig [31]. An overview and comparison of simplification orders have been given by Steinbach [63, 64]. Furthermore, several AC-compatible simplification orders have been proposed based on simplification orders. The order introduced by Cherifa and Lescanne is based on polynomial interpretation [9]. The order introduced by Bachmair and Plaisted based on the recursive path order with flattening [7], and it has been extended by Bachmair [8], by Delor and Puel [14], by Kapur, Sivakumar and Zhang [34, 35], by Rubio and Nieuwenhuis [61, 62], and so on.

On the other hand, proving termination by simplification orders has a theoretical limitation. In fact, there exist terminating TRSs that can not be essentially proved by simplification orders. In order to prove termination of such TRSs, we have two methods. One is a transformation method that transforms a given TRS into a TRS whose termination is easier to prove than the original one. The representative transformations are elimination transformations. The dummy elimination [20], the distribution elimination [53, 67], the general dummy elimination [21] and the improved general dummy elimination [57] are examples of elimination transformations. Furthermore, the dummy elimination and the distribution elimination are extended to AC-TRSs in [22] and [58], respectively. Another one is the dependency pair method, introduced by Arts and Giesl, that is a method to check a dependency of function call sequences in evaluating processes of TRSs as programs [1, 2]. Dependency pairs are useful not only proving termination but also analyzing an infinite reduction sequence. Furthermore, separate extensions of the dependency pair to AC-TRSs were independently done by us in [43] and by Marché and Urbain in [52].

Structure of the Thesis: The next chapter gives the preliminaries needed later on. In Chapter 3, we review results about dependency pairs. First, we recall basic notions and fundamental results on dependency pairs. The method of dependency pairs compares rewrite rules and dependency pairs by a weak reduction order or by a weak reduction pair, which play an important role on the method of dependency pair, instead of a reduction order. In Section 3.2, we introduce the notion of weak reduction order and its application for proving termination. We explain two methods to design weak reduction orders. One is the argument filtering method, which allows us to make a weak reduction order from an arbitrary reduction order. Another one is the polynomial interpretation method. A set of dependency pairs itself makes a TRS. However, a TRS, the set of its dependency pairs and the union of them do not accurately agree on the termination property. In Section 3.3, we present a hierarchy for the termination property among these systems. In Section 3.4, we introduce the dependency graph, that gives a more powerful method for analyzing an infinite reduction sequence.

In Chapter 4, we extend the notion of dependency pairs to AC-TRSs. It is impossible to directly apply the notion of the dependency pair to AC-TRSs. In Section 4.1, we show this difficulty through an example. To avoid this difficulty we introduce the head parts in terms and show an analogy between the root positions in infinite reduction sequences by TRSs and the head positions in those by AC-TRSs. Indeed, this analogy is essential

for the extension of dependency pairs to AC-TRSs. Based on this analogy, we define AC-dependency pairs and AC-dependency chains. In Section 4.2, we introduce the argument filtering method, which generates a weak AC-reduction order and a weak AC-reduction pair. The original idea of the argument filtering method for TRSs without AC-function symbols was first proposed by Arts and Giesl [5, 23]. To analyze other proving methods for termination, the method was slightly improved by combining the subterm relation [46]. We extend these methods to AC-TRSs. Our extension gives a design of a weak AC-reduction order and a weak AC-reduction pair from an arbitrary AC-reduction order. Moreover, in order to strengthen the power of the argument filtering method, we improve the method in two directions. One is the lexicographic argument filtering method, in which argument filtering functions are lexicographically combined to compare AC-dependency pairs. Another one is an extension over multisets. In the argument filtering method on AC-TRSs, any argument filtering function must be compatible to AC-equations. We relax this restriction using the extension over multisets. These methods are effective for proving not only AC-termination but also termination of TRSs.

In Chapter 5, we study the relation between the argument filtering method and various elimination transformations. The key of our result is the observation that the argument filtering method combined with the dependency pair technique is essential in all elimination transformations. Indeed, we present remarkable simple proofs for the soundness of these elimination transformations based on this observation, though the original proofs presented in the literatures [20, 21, 22, 53, 57, 58, 67] developed rather different methods respectively. This observation also leads us to a new powerful elimination transformations, called the argument filtering transformation, which is not only more powerful than all the other elimination transformations but also especially useful to make clear an essential relation hidden behind these methods.

Chapter 2

Preliminaries

In this chapter, we present the basic notions used in this thesis about orders, term rewriting systems and AC-term rewriting systems. More details can be found in Baader and Nipkow [6], Dershowitz and Jouannaud [18], and Klop [39].

2.1 Binary Relations

First let us remark that we assume familiarity with basic mathematical no(ta)tions and terminologies about functions, sets, pairs and so on.

2.1.1 Binary Relations and Orders

Definition 2.1.1 *A binary relation on a set A is a subset of $A \times A$. For any binary relation Υ on a set A, we write $a_1 \Upsilon a_2$ instead of $(a_1, a_2) \in \Upsilon$. The composition of binary relations Υ_1 and Υ_2 on a set A, denoted by $\Upsilon_1 \circ \Upsilon_2$, is defined as follows:*

$$\Upsilon_1 \circ \Upsilon_2 = \{(a, a'') \mid \exists a'.\ a \Upsilon_1 a' \wedge a' \Upsilon_2 a''\}$$

For any binary relation Υ on a set A and for any natural number n, we define Υ^n as follows:

$$\begin{cases} \Upsilon^0 & = & \{(a, a) \mid a \in A\} \\ \Upsilon^{n+1} & = & \Upsilon^n \circ \Upsilon \end{cases}$$

The inverse relation of a binary relation Υ, denoted by Υ^{-1}, is defined by $\{(a', a) \mid a \Upsilon a'\}$. A binary relation Υ is compatible with a set R of pairs if $a \Upsilon a'$ for all $(a, a') \in R$.

Definition 2.1.2 *A binary relation Υ on a set A is said to be*

- *transitive if it satisfies $a_1 \Upsilon a_2 \wedge a_2 \Upsilon a_3 \Rightarrow a_1 \Upsilon a_3$ for all $a_1, a_2, a_3 \in A$,*
- *reflexive if it satisfies $a \Upsilon a$ for all $a \in A$,*
- *irreflexive if it satisfies $\neg(a \Upsilon a)$ for all $a \in A$,*
- *symmetric if it satisfies $a_1 \Upsilon a_2 \Rightarrow a_2 \Upsilon a_1$ for all $a_1, a_2 \in A$, and*
- *antisymmetric if it satisfies $a_1 \Upsilon a_2 \wedge a_2 \Upsilon a_1 \Rightarrow a_1 = a_2$ for all $a_1, a_2 \in A$.*

Definition 2.1.3 *The closure of a binary relation Υ with respect to a property P is the smallest binary relation containing Υ and satisfying P. We write the reflexive closure, the transitive closure and the reflexive-transitive closure of Υ as $\Upsilon^=$, Υ^+ and Υ^*, respectively.*

Proposition 2.1.4 *Let Υ be a binary relation on a set A. Then, $\Upsilon^= = \Upsilon^0 \cup \Upsilon$, $\Upsilon^+ = \bigcup_{n=1}^{\infty} \Upsilon^n$ and $\Upsilon^* = \bigcup_{n=0}^{\infty} \Upsilon^n$.*

Definition 2.1.5 *A binary relation is an equivalence relation if it is a reflexive, transitive and symmetric. Let $\sim$ be an equivalence relation on a set A. For any $x \in A$, the set $\{a \in A \mid x \sim a\}$ is called the equivalence class of x modulo $\sim$ and is denoted by $[\![x]\!]$. The quotient set of A modulo $\sim$, denoted by $A/\sim$, is defined by $\{[\![x]\!] \mid x \in A\}$.*

Note that any two distinct equivalence classes on a set A are non-empty and disjoint, and the union of all equivalence classes is equivalent to A.

Definition 2.1.6 *A binary relation on a set A is a strict order, usually denoted by $>$, if it is transitive and irreflexive. A binary relation on a set A is a partial order, usually denoted by $\geq$, if it is reflexive, transitive and antisymmetric. A binary relation on a set A is a quasi-order, usually denoted by $\gtrsim$, if it is transitive and reflexive.*

Definition 2.1.7 *For any quasi-order $\gtrsim$, partial order $\geq$ and strict order $>$, we write their inverse relations $\gtrsim^{-1}$, $\geq^{-1}$ and $>^{-1}$ by $\lesssim$, $\leq$ and $<$, respectively. The strict part of a quasi-order $\gtrsim$, written by $\gnsim$, is defined as $\gtrsim \setminus \lesssim$. The equivalence part of a quasi-order $\gtrsim$, written by $\sim$, is defined as $\gtrsim \cap \lesssim$.*

Proposition 2.1.8 *For any quasi-order $\gtrsim$, its strict part $\gnsim$ is a strict order and its equivalence part $\sim$ is an equivalence relation.*

Definition 2.1.9 *A binary relation $>$ on a set A is said to be*

- *well-founded if every non-empty subset of A has a minimal element,*
- *terminating or strongly normalizing if there exist no infinite decreasing sequences of the form $a_1 > a_2 > a_3 > \cdots$.*

Here, a subset $B \subseteq A$ has a minimal element if there exists $b \in B$ such that $b > a$ implies $a \notin B$.

It is well known that well-foundedness and termination are equivalent concepts by the Axiom of Choice.

2.1.2 Multiset Extension

In this subsection, we consider useful extensions for strict order and quasi-order, called the multiset extension. First we define multisets.

Definition 2.1.10 *Let A be a set. A multiset on a set A is a function M from A to natural numbers. A multiset on a set A is a finite multiset if $M(a) \neq 0$ only for finitely many $a \in A$. The set of all finite multisets on a set A is denoted by $M(A)$.*

Definition 2.1.11 *Let M_1 and M_2 be multisets on a set A. The relations $\in$, $=$, $\subseteq$ and $\subset$ are defined as follows:*

$$\begin{array}{rcl} a \in M_1 & \overset{\text{def}}{\Longleftrightarrow} & 0 < M_1(a) \\ M_1 = M_2 & \overset{\text{def}}{\Longleftrightarrow} & M_1(a) = M_2(a) \quad \textit{for all} \;\; a \in A \\ M_1 \subseteq M_2 & \overset{\text{def}}{\Longleftrightarrow} & M_1(a) \leq M_2(a) \quad \textit{for all} \;\; a \in A \\ M_1 \subset M_2 & \overset{\text{def}}{\Longleftrightarrow} & M_1 \neq M_2 \wedge M_1 \subseteq M_2 \end{array}$$

The operations $\cup$, $\cap$ and $-$ on multisets are defined as follows:

$$\begin{array}{rcl} M_1 \cup M_2 & \overset{\text{def}}{=\!=} & (M_1 \cup M_2)(a) = M_1(a) + M_2(a) \\ M_1 \cap M_2 & \overset{\text{def}}{=\!=} & (M_1 \cap M_2)(a) = min\{M_1(a), M_2(a)\} \\ M_1 - M_2 & \overset{\text{def}}{=\!=} & (M_1 - M_2)(a) = max\{M_1(a) - M_2(a), 0\} \end{array}$$

Intuitively, a multiset on a set A is a set of elements of A in which elements may have multiple occurrences. We use standard set notation like $\{a, a, b\}$ as an abbreviation of the multisets M such that $M = \{a \mapsto 2, b \mapsto 1, c \mapsto 0\}$ on the set $A = \{a, b, c\}$. It will be obvious from the context if we refer to a set or a multiset.

We now consider the multiset extension of given strict order on a set A, which is a binary relation on $M(A)$. There exist several definitions of the multiset extension [6, 19, 29, 30]. Here, we give four definitions of the multiset extension, denoted by $\gg$, for given strict order $>$. We have known that these definitions are equivalent [6, 30].

Definition 2.1.12 *Let $>$ be a strict order on a set A. The multiset extension $\gg$ is defined as follows:*

$$\begin{array}{rcl} M \gg N & \overset{\text{def}}{\Longleftrightarrow} & \exists X, Y \in M(A) \\ & & [\emptyset \neq X \subseteq M \wedge N = (M - X) \cup Y \wedge \forall y \in Y.\ \exists x \in X.\ x > y] \end{array}$$

Definition 2.1.13 *Let $>$ be a strict order on a set A. The multiset extension $\gg$ is defined as follows:*

$$M \gg N \overset{\text{def}}{\Longleftrightarrow} M \neq N \wedge \forall n \in N - M.\ \exists m \in M - N.\ m > n$$

Definition 2.1.14 *Let $>$ be a strict order on a set A. The multiset extension $\gg$ is defined as follows:*

$$M \gg N \overset{\text{def}}{\Longleftrightarrow} M \neq N \wedge [\exists a \in A.\ N(a) >_{\mathcal{N}} M(a) \Rightarrow \exists a' \in A.\ a' > a \wedge M(a') >_{\mathcal{N}} N(a')],$$

where $>_{\mathcal{N}}$ is usual order on natural numbers.

Definition 2.1.15 *Let $>$ be a strict order on a set A. We define the single-step relation on $M(A)$ as follows:*

$$M \gg^1 N \overset{\text{def}}{\Longleftrightarrow} \exists x \in M.\ \exists Y \in M(A).\ N = (M - \{x\}) \cup Y \wedge \forall y \in Y.\ x > y$$

The multiset extension $\gg$ is defined by the transitive closure of $\gg^1$.

Proposition 2.1.16 *[19] $>$ is a strict order on a set A if and only if its multiset extension $\gg$ is a strict order on $M(A)$. Furthermore, $>$ is well-founded if and only if $\gg$ is well-founded.*

Next, we define the multiset extension for quasi-orders based on that for strict orders. Here we should prepare several notions to treat the equivalence classes.

Definition 2.1.17 *Let $\sim$ be an equivalence relation on a set A. For any multiset $M = \{a_1, a_2, \ldots, a_n\}$, we define $M_E = \{[\![a_1]\!], [\![a_2]\!], \ldots, [\![a_n]\!]\}$, where $[\![a_i]\!]$ is the equivalence class of a_i modulo $\sim$. Let M' and M'' be multisets on a set A. The relations $\in_E$, $=_E$, $\subseteq_E$ and $\subset_E$ are defined as follows:*

$$\begin{array}{rcl} a \in_E M' & \overset{\text{def}}{\Longleftrightarrow} & [\![a]\!] \in M'_E \\ M' =_E M'' & \overset{\text{def}}{\Longleftrightarrow} & M'_E = M''_E \\ M' \subseteq_E M'' & \overset{\text{def}}{\Longleftrightarrow} & M'_E \subseteq M''_E \\ M' \subset_E M'' & \overset{\text{def}}{\Longleftrightarrow} & M'_E \subset M''_E \end{array}$$

The operations $\cup_E$, $\cap_E$ and $-_E$ on multisets are defined as follows:

$$\begin{array}{rcl} M' \cup_E M'' & \overset{\text{def}}{=\!=} & M'_E \cup M''_E \\ M' \cap_E M'' & \overset{\text{def}}{=\!=} & M'_E \cap M''_E \\ M' -_E M'' & \overset{\text{def}}{=\!=} & M'_E - M''_E \end{array}$$

In this thesis, we treat only AC-equation $\underset{AC}{\sim}$ as equation. Hence, it is enough to treat $_{AC}$ as the subscript $_E$. We often omit the subscript $_E$ or $_{AC}$ whenever no confusion arises.

Lemma 2.1.18 *Let $\gtrsim$ be a quasi-order on a set A and $\sim$ be its equivalence part. We define the binary relation $\gtrsim_E$ on $A/\sim$ by $[\![a_1]\!] \gtrsim_E [\![a_2]\!] \overset{\text{def}}{\Longleftrightarrow} a_1 \gtrsim a_2$. Then, $\gtrsim_E$ is a quasi-order on $A/\sim$.*

Proof. Let $[\![a_1]\!] = [\![a_2]\!]$. Since $a_1 \sim a_2$, it follows that $a_1 \gtrsim a_2$. Thus $\gtrsim_E$ is reflexive. Let $[\![a_1]\!] \gtrsim_E [\![a_2]\!]$ and $[\![a_2]\!] \gtrsim_E [\![a_3]\!]$. Since $a_1 \gtrsim a_2 \gtrsim a_3$, it follows that $a_1 \gtrsim a_3$. Thus, $[\![a_1]\!] \gtrsim_E [\![a_3]\!]$. Therefore $\gtrsim_E$ is transitive. $\square$

Definition 2.1.19 *Let $\gtrsim$ be a quasi-order on a set A, $\sim$ the equivalence part of $\gtrsim$, $\gtrsim_E$ the binary relation on $A/\sim$ defined in Lemma 2.1.18, $\underset{\not\sim}{\gtrsim}_E$ the strict part of $\gtrsim_E$, and $\gg$ the multiset extension of $\underset{\not\sim}{\gtrsim}_E$ for strict orders. We define the multiset extension $\underline{\gg}$ for quasi-orders by $M' \underline{\gg} M'' \overset{\text{def}}{\Longleftrightarrow} M'_E \gg M''_E \vee M'_E = M''_E$.*

Proposition 2.1.20 *[21] $\gtrsim$ is a quasi-order on a set A if and only if its multiset extension $\underline{\gg}$ is a quasi-order on $M(A)$. Furthermore, the strict part $\underset{\not\sim}{\gtrsim}$ of $\gtrsim$ is well-founded if and only if the strict part $\gg$ of $\underline{\gg}$ is well-founded.*

2.1.3 Lexicographic Extension

In this subsection, we consider the lexicographic extension.

Definition 2.1.21 *Let $>$ be a strict order on a set A. The lexicographic extension $>_{lex}$ on lists of A is recursively defined as follows:*

$$\begin{cases} [a_1, a_2, \ldots, a_n] & >_{lex} & [] & \text{if } n > 0 \\ [a_1, a_2, \ldots, a_n] & >_{lex} & [a'_1, a'_2, \ldots, a'_m] & \text{if } a_1 > a'_1 \\ [a_1, a_2, \ldots, a_n] & >_{lex} & [a'_1, a'_2, \ldots, a'_m] & \text{if } a_1 = a'_1 \wedge [a_2, \ldots, a_n] >_{lex} [a'_2, \ldots, a'_m] \end{cases}$$

Unfortunately, the well-foundedness of $>$ does not guarantee that of $>_{lex}$. In fact, for the set $A = \{a, b\}$ with $a > b$, there exists an infinite decreasing sequence $[a] >_{lex} [b, a] >_{lex} [b, b, a] >_{lex} \cdots$. We can avoid this problem by restricting the max length of lists.

Proposition 2.1.22 *$>$ is a strict order on a set A if and only if its lexicographic extension $>_{lex}$ is a strict order on lists of A. Furthermore, for arbitrary positive number n, $>$ is well-founded if and only if $>_{lex}$ is well-founded on lists of A, whose length are less than or equal to n.*

2.2 Term Rewriting Systems

We introduce the basic notions of term rewriting systems.

Definition 2.2.1 *A signature Σ is a finite set of function symbols, where each $f \in \Sigma$ is associated with natural number n, written by $arity(f)$. A set $\mathcal{V}$ is an enumerable set of variables with $\Sigma \cap \mathcal{V} = \emptyset$. The set of terms, written by $\mathcal{T}(\Sigma, \mathcal{V})$, is the smallest set containing $\mathcal{V}$ such that $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$ whenever $f \in \Sigma$, $arity(f) = n$ and $t_i \in \mathcal{T}(\Sigma, \mathcal{V})$ for $i = 1, \ldots, n$. For a function symbol e with $arity(e) = 0$, we write e instead of $e()$. Identity of terms is denoted by $\equiv$. $Var(t)$ is the set of variables in t. The size $|t|$ of a term t is the number of function symbols and variables in t. A term t is linear if every variable in t occurs only once.*

Definition 2.2.2 *A term position is a sequence of positive integers. We denote the empty sequence by ε. The prefix order $\prec$ on term positions is defined by $p \prec q$ iff $pw = q$ for some w ($\neq \varepsilon$). We recursively define $(t)_p$ the symbol at position p in t, and $t|_p$ the subterm of t at position p as follows:*

$$\begin{cases} (x)_\varepsilon & = & x \\ (f(t_1, \ldots, t_n))_\varepsilon & = & f \\ (f(t_1, \ldots, t_n))_{i \cdot p} & = & (t_i)_p \end{cases} \qquad \begin{cases} x|_\varepsilon & = & x \\ f(t_1, \ldots, t_n)|_\varepsilon & = & f(t_1, \ldots, t_n) \\ f(t_1, \ldots, t_n)|_{i \cdot p} & = & t_i|_p \end{cases}$$

Definition 2.2.3 *A substitution $\theta : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is a mapping. A substitution over terms is defined as the homomorphic extension through $\theta(f(t_1, \ldots, t_n)) = f(\theta(t_1), \ldots, \theta(t_n))$ for $f \in \Sigma$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$. Two terms s and t are unifiable if there exists a substitution θ such that $\theta(s) \equiv \theta(t)$. We write $t\theta$ instead of $\theta(t)$.*

Definition 2.2.4 *A context C is a term with occurrences of a special constant $\Box$, called a hole. $C[t_1, \ldots, t_n]$ denotes the result of placing $t_1, \ldots, t_n$ in the n holes of C from left to right. In particular $C[\ ]$ denotes a context containing precisely one hole and we sometimes write $C[t]_p$ to indicate at which position the replacement takes place.*

Definition 2.2.5 *A term s is called a subterm of t if $t \equiv C[s]$ for some context C. A subterm s of t is called a proper subterm if $s \not\equiv t$.*

Definition 2.2.6 *A binary relation Υ on $\mathcal{T}(\Sigma, \mathcal{V})$ is said to be*

- *monotonic if it satisfies $s\Upsilon t \Rightarrow C[s]\Upsilon C[t]$ for all contexts $C[\ ]$,*
- *stable if it satisfies $s\Upsilon t \Rightarrow s\theta\Upsilon t\theta$ for all substitutions θ.*

A congruence relation is an equivalence, monotonic and stable relation.

Definition 2.2.7 *A rewrite rule is a pair of terms, written by $l \to r$, with $l \notin \mathcal{V}$ and $Var(l) \supseteq Var(r)$. A term rewriting system (TRS) is a finite set of rules. The set of defined symbols in R is $DF(R) = \{(l)_\varepsilon \mid l \to r \in R\}$. A reduction relation $\underset{R}{\to}$ is defined as follows:*

$$s \underset{R}{\to} t \iff^{\text{def}} s \equiv C[l\theta] \wedge t \equiv C[r\theta] \quad \textit{for some } l \to r \in R,\ C[\ ] \textit{ and } \theta$$

When we want to specify the position p of $C[l\theta]_p$ in the above reductions, we write $s \underset{R}{\to}_p t$. A step of the form $s \underset{R}{\to}_\varepsilon t$ is called a root reduction step. We often omit the subscripts R whenever no confusion arises.

Definition 2.2.8 *A TRS R is terminating if its reduction relation $\underset{R}{\to}$ is terminating.*

2.3 AC-Term Rewriting Systems

We introduce the basic notions of AC-term rewriting systems.

Definition 2.3.1 *The set Σ_{AC} of AC-function symbols, which have fixed arity 2, is a subset of Σ. The binary relation $\underset{AC}{\sim}$ is the congruence relation generated by $f(f(x, y), z) =_A f(x, f(y, z))$ and $f(x, y) =_C f(y, x)$ for each $f \in \Sigma_{AC}$.*

Definition 2.3.2 *Two terms s and t are AC-unifiable if there exists a substitution θ such that $\theta(s) \underset{AC}{\sim} \theta(t)$. A set of terms T is AC-unifiable if there exists a substitution θ such that $\theta(s) \underset{AC}{\sim} \theta(t)$ for all $s, t \in T$.*

Definition 2.3.3 *An AC-term rewriting system (AC-TRS) is a TRS with AC-function symbols Σ_{AC}. An AC-reduction relation[1] $\underset{R/AC}{\to}$ is defined as follows:*

$$s \underset{R/AC}{\to} t \iff^{\text{def}} s \underset{AC}{\sim} C[l\theta] \wedge t \equiv C[r\theta] \quad \textit{for some } l \to r \in R,\ C[\ ] \textit{ and } \theta$$

We often omit the subscripts R/AC whenever no confusion arises.

[1] In this thesis, we introduce AC-TRSs as TRSs with AC-matching. On the other hand, in usual way, AC-TRSs have been introduced as rewriting systems over equivalence classes of terms modulo $\underset{AC}{\sim}$, that is, $t \underset{AC}{\sim} C[r\theta]$ is used instead of $t \equiv C[r\theta]$ in the above definition. For studying AC-termination, these two systems are essentially same. Refer to [52].

Definition 2.3.4 *An AC-TRS R is AC-terminating if its AC-reduction relation $\underset{R/AC}{\rightarrow}$ is terminating.*

2.4 Reduction Order and AC-Reduction Order

The termination property of TRSs is undecidable [27], even for one-rule systems [13]. Nevertheless, it is often necessary to prove the termination for a particular system. To prove (AC-)termination, we commonly design a (AC-)reduction order by which all rules are ordered. Firstly we introduce the notion of (AC-)reduction orders.

Definition 2.4.1 *A rewrite order $>$ is a strict order that is monotonic and stable. A reduction order $>$ is a well-founded rewrite order. An AC-reduction order $>$ is a reduction order that is AC-compatible, i.e., $s \underset{AC}{\sim} s' > t \Rightarrow s > t$.*

Theorem 2.4.2 *A TRS R is terminating iff there exists a reduction order $>$ that is compatible with R. An AC-TRS R is AC-terminating iff there exists an AC-reduction order that is compatible with R.*

Proof. It suffices to show the cases for AC-TRSs, because each TRS is a special form of AC-TRSs, i.e., $\Sigma_{AC} = \emptyset$. Let R be an AC-TRS. From the definition, it is trivial that R is AC-terminating iff the transitive closure of its AC-reduction relation $\overset{+}{\underset{R/AC}{\rightarrow}}$ is well-founded. Moreover, $\overset{+}{\underset{R/AC}{\rightarrow}}$ is well-founded iff $\overset{+}{\underset{R/AC}{\rightarrow}}$ is an AC-reduction order, which obviously satisfies $l \overset{+}{\underset{R/AC}{\rightarrow}} r$ for all $l \rightarrow r \in R$. □

In order to prove termination of TRSs, several reduction orders have been proposed. In this section, we introduce reduction orders based on polynomial interpretations [47, 50, 51] and simplification orders [15]. We also introduce several AC-reduction orders, which are designed based on reduction orders.

2.4.1 Polynomial Interpretation

In this subsection, we introduce methods for designing reduction orders and AC-reduction orders based on polynomial interpretations, in which function symbols are interpreted as polynomials over natural numbers.

Definition 2.4.3 *Let $A \subseteq \mathcal{N}$. We denote a polynomial $f_A : A^n \rightarrow A$ for any $f \in \Sigma$ with arity n. We call a polynomial f_A a monotone polynomial if it depends on all its indeterminates, i.e., for all i $(1 \leq i \leq n)$, it contains a monomial (with non-zero coefficient) in which x_i occurs with an exponent at least 1. For any $\sigma : \mathcal{V} \rightarrow A$, a polynomial interpretation $[\![t]\!]_\sigma$ is defined as follows:*

$$[\![x]\!]_\sigma = \sigma(x) \text{ and } [\![f(t_1, \ldots, t_n)]\!]_\sigma = f_A([\![t_1]\!]_\sigma, \ldots, [\![t_n]\!]_\sigma)$$

The polynomial strict order $>_A$ is defined as follows:

$$s >_A t \overset{\text{def}}{\Longleftrightarrow} \forall\sigma([\![s]\!]_\sigma > [\![t]\!]_\sigma)$$

A monotone polynomial strict order is a polynomial strict order in which all f_A is a monotone polynomial.

Proposition 2.4.4 *[47, 50, 51] Let $A \subseteq \mathcal{N}\backslash\{0\}$. A monotone polynomial strict order is a reduction order.*

Proposition 2.4.5 *[9] Let $A \subseteq \mathcal{N}\backslash\{0\}$. If for any $f \in \Sigma_{AC}$ we associate the polynomial $f_A = f_2XY + f_1(X+Y) + f_0$ with $f_2f_0 = f_1(f_1 - 1)$ for some $f_0, f_1, f_2 \in A$ then the monotone polynomial strict order is an AC-reduction order.*

2.4.2 Simplification Order

The most important study for designing reduction orders is the notion of simplification orders introduced by Dershowitz [15]. Based on the notion several reduction orders are introduced [15, 16, 17, 31, 32, 33].

Definition 2.4.6 *A simplification order $>$ is a rewrite order on $\mathcal{T}(\Sigma, \mathcal{V})$ with the subterm property, i.e., $C[t] > t$ for all term t and non-empty context C ($\not\equiv \Box$). An AC-compatible simplification order $>$ is a simplification order with the AC-compatibility i.e., $s \underset{AC}{\sim} s' > t \Rightarrow s > t$.*

Note that simplification orders require the subterm property instead of the well-foundedness. This simplifies the design of reduction orders, because mostly the subterm property is easier to prove than the well-foundedness. We should mention that this simplification does not lose the well-foundedness.

Proposition 2.4.7 *[15, 16] Any simplification order is well-founded. Hence any simplification order is a reduction order[2].*

The proof of this proposition is based on Kruskal's Embedding Theorem [42].

Definition 2.4.8 *A TRS R is simply terminating if there exists a simplification order compatible with R.*

We define the embedding TRS $Emb = \{f(x_1, \ldots, x_i, \ldots, x_n) \to x_i \mid f \in \Sigma,\ 1 \leq i \leq n = arity(f)\}$.

Proposition 2.4.9 *[41] A TRS R is simply terminating if and only if $R \cup Emb$ is terminating.*

As representative simplification orders, we introduce the recursive path order and the lexicographic path order.

Definition 2.4.10 *(Recursive Path Order) Let $\rhd$ be a strict order on Σ, called precedence. We define $s >_{rpo} t$ as follows:*

- *$t \in Var(s)$ and $s \not\equiv t$, or*
- *$s \equiv f(s_1, \ldots, s_n)$ and $t \equiv g(t_1, \ldots, t_m)$, and*
  - *$f \rhd g$ and $s >_{rpo} t_j$ for all j,*

[2]In this thesis, we restrict that the signature Σ is finite. In infinite signatures case, this proposition does not hold. Refer to [54].

- $f = g$ *and* $\{s_1, \ldots, s_n\} \gg_{rpo} \{t_1, \ldots, t_m\}$, *or*
- $s_i \geq_{rpo} t$ *for some* i.

Proposition 2.4.11 *[15, 16] The recursive path order* $>_{rpo}$ *is a simplification order.*

Definition 2.4.12 *(Lexicographic Path Order) Let* $\rhd$ *be a strict order on* Σ*, called precedence. We define* $s >_{lpo} t$ *as follows:*

- $t \in Var(s)$ *and* $s \not\equiv t$, *or*
- $s \equiv f(s_1, \ldots, s_n)$ *and* $t \equiv g(t_1, \ldots, t_m)$, *and*
  - $f \rhd g$ *and* $s >_{lpo} t_j$ *for all* j,
  - $f = g$, $[s_1, \ldots, s_n] >_{lpo}^{lex} [t_1, \ldots, t_m]$ *and* $s >_{lpo} t_j$ *for all* j, *or*
  - $s_i \geq_{lpo} t$ *for some* i.

Proposition 2.4.13 *[17, 32] The lexicographic path order* $>_{lpo}$ *is a simplification order.*

To extend the recursive order to an AC-reduction order, flattening terms were introduced. The flattening term of a term t, denoted $\overline{t}$, is the normal form of t for the rules $f(\vec{x_i}, f(\vec{y_i}), \vec{z_i}) \to f(\vec{x_i}, \vec{y_i}, \vec{z_i})$ for each AC-symbol f, where $x_1, \ldots, x_n$ is abbreviated to $\vec{x}$. Notice that we allow that the arity of AC-function symbols does not fix. Using flattening terms and the recursive path order $>_{rpo}$, we define $s >_{rpo}^{flat} t$ by $\overline{s} >_{rpo} \overline{t}$[3]. However, this order $>_{rpo}^{flat}$ is not always monotonic. Therefore, we need a suitable restriction on the precedence, as shown by the following proposition.

Proposition 2.4.14 *[7] If all AC-symbols are minimal in a precedence* $\rhd$*, then the order* $>_{rpo}^{flat}$ *is an AC-reduction order.*

Based on this work a lot of AC-reduction orders have been proposed by Bachmair [8], by Delor and Puel [14], by Kapur, Sivakumar and Zhang [34, 35], by Rubio and Nieuwenhuis [61], and so on. Finally, we explain the newest AC-reduction order, introduced by Rubio, which does not require any restriction on the precedence [62].

Definition 2.4.15 *We suppose that each term is a flattening term. Let* $>$ *be a binary relation on terms,* $\rhd$ *be a precedence and* f *be an AC-function symbol. The multiset extension of* $>$ *w.r.t.* f*, denoted by* $\underline{\succ}_f$*, is defined as the smallest transitive relation including* $=_{AC}$ *and satisfying the following property:*

$$X \cup \{s\} \underline{\succ}_f Y \cup \{t_1, \ldots, t_n\} \qquad \text{if} \begin{cases} X =_{AC} Y,\ s > t_i,\ \text{and} \\ \text{if } (s)_\varepsilon \not\rhd f \text{ then } (s)_\varepsilon \trianglerighteq (t_i)_\varepsilon, \\ \text{for all } i \in \{1, \ldots, n\} \end{cases}$$

For any s, $\#(s)$ *is an expression with variables on the positive integers, defined as* $\#(f(s_i, \ldots, s_n)) = \Sigma_{i=1}^{n} \#_v(s_i)$, *where* $\#_v(x) = x$ *and* $\#_v(t) = 1$ *if* $t \notin \mathcal{V}$. *We define* $\#(s) >_{poly} \#(t)$ *by* $\sigma(\#(s)) > \sigma(\#(t))$ *for all* $\sigma : \mathcal{V} \to \mathcal{N} \backslash \{0\}$, *and* $\#(s) \geq_{poly} \#(t)$ *by*

[3]Strictly speaking, the binary relation $>_{rpo}$ in $\overline{s} >_{rpo} \overline{t}$ is different from Definition 2.4.10: $s_i \geq_{rpo} t$ is interpreted as $s_i >_{rpo} t$ or $s_i \underset{AC}{\sim} t$.

$\sigma(\#(s)) \geq \sigma(\#(t))$ *for all* $\sigma : \mathcal{V} \to \mathcal{N}\backslash\{0\}$. *The top-flattening of a term* s *w.r.t.* f *is a string of terms defined as follows:*

$$tf_f(s) = \begin{cases} s_1, \ldots, s_n & \text{if} \quad s \equiv f(s_1, \ldots, s_n) \\ s & \text{if} \quad (s)_\varepsilon \neq f \end{cases}$$

For any $s \equiv f(s_1, \ldots, s_n)$, *we define* $BigHead(s)$, $NoSmallHead(s)$ *and* $EmbNoBig(s)$ *as follows:*

$$\begin{aligned} BigHead(s) &= \{s_i \mid (s_i)_\varepsilon \rhd f\} \\ NoSmallHead(s) &= \{s_i \mid f \not\rhd (s_i)_\varepsilon\} \\ EmbNoBig(s) &= \{f(s_1, \ldots, tf_f(t_{ij}), \ldots, s_n) \mid s_i \equiv h(t_{i1}, \ldots, t_{im}), h \not\rhd f\} \end{aligned}$$

We define $s > t$ *as follows:*

- $t \in Var(s)$ *and* $s \not\equiv t$, *or*
- $s \equiv f(s_1, \ldots, s_n)$ *and* $t \equiv g(t_1, \ldots, t_m)$, *and*
  - $f \rhd g$ *and* $s > t_j$ *for all* j,
  - $f = g \notin \Sigma_{AC}$, $[s_1, \ldots, s_n] >^{lex} [t_1, \ldots, t_m]$ *and* $s > t_j$ *for all* j,
  - $s_i \geq t$ *for some* i,
  - $f = g \in \Sigma_{AC}$ *and* $s' \geq t$ *for some* $s' \in EmbNoBig(s)$, *or*
  - $f = g \in \Sigma_{AC}$, $s \geq t'$ *for all* $t' \in EmbNoBig(t)$, $NoSmallHead(s) \not\geq_f NoSmallHead(t)$, *and either*
    - $BigHead(s) \gg BigHead(t)$,
    - $\#(s) >_{poly} \#(t)$, *or*
    - $\#(s) \geq_{poly} \#(t)$ *and* $\{s_1, \ldots, s_n\} \gg \{t_1, \ldots, t_m\}$,

where $\geq$ *is the union of* $>$ *and* $\underset{AC}{\sim}$.

Proposition 2.4.16 *[62] The binary relation* $>$ *in Definition 2.4.15 is an AC-compatible simplification order. Hence it is an AC-reduction order.*

Chapter 3

Dependency Pairs

Recently, Arts and Giesl proposed the notion of dependency pairs, which can offer effective methods for analyzing an infinite reduction sequence [1, 2]. Intuitively, the dependency pair is a method to check a dependency of function call sequences in evaluating processes of term rewriting systems as programs. Using dependency pairs, we can easily show the termination property of TRSs to which traditional techniques cannot be applied directly. In Section 3.1, we introduce the notion and basic results.

The method of dependency pairs compares rewrite rules and dependency pairs by a weak reduction order, which plays an important role on this method, instead of a reduction order. In Section 3.2, we introduce the notion of weak reduction orders and its application for proving termination. We introduce two methods to design weak reduction orders. One is the argument filtering method, which designs a weak reduction order from an arbitrary reduction order. Another one is the polynomial interpretation methods.

A set of dependency pairs itself is regarded as a term rewriting system. However, a term rewriting system, the set of its dependency pairs and the union of them do not accurately agree on the termination property. In Section 3.3, we present a hierarchy for the termination property between these systems.

In Section 3.4, we state the dependency graph, which gives a more powerful method for analyzing an infinite reduction sequence.

3.1 Dependency Pair and Dependency Chain

In this section, we explain dependency pairs and dependency chains.

Definition 3.1.1 *Let $\Sigma^\# = \Sigma \uplus \{f^\# \mid f \in \Sigma\}$ be a set of function symbols. We define the marking function $\# : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{T}(\Sigma^\#, \mathcal{V})$ by $\#(x) = x$ and $\#(f(t_1, \ldots, t_n)) = f^\#(t_1, \ldots, t_n)$. We write $t^\#$ instead of $\#(t)$.*

Definition 3.1.2 *Let R be a TRS. A pair $\langle u^\#, v^\# \rangle$ of terms is a dependency pair of R if there exists a rule $u \to C[v] \in R$ for some C such that $(v)_\varepsilon \in DF(R)$. The set of dependency pairs of R is written by $DP^\#(R)$. The set of unmarked dependency pairs of R, written by $DP(R)$, is obtained by erasing marks of symbols in $DP^\#(R)$.*

Example 3.1.3 *Let $R = \{add(x, 0) \to x,\ add(x, s(y)) \to s(add(x, y))\}$. Then,*

$$DP(R) = \{\langle add(x, s(y)), add(x, y) \rangle\}, \qquad DP^\#(R) = \{\langle add^\#(x, s(y)), add^\#(x, y) \rangle\}.$$

Definition 3.1.4 *A sequence of dependency pairs $\langle u_0^\#, v_0^\#\rangle\langle u_1^\#, v_1^\#\rangle\langle u_2^\#, v_2^\#\rangle\cdots$ is a dependency chain if there exists a substitution θ over $\mathcal{T}(\Sigma,\mathcal{V})$ such that $v_i^\#\theta \xrightarrow[R]{*} u_{i+1}^\#\theta$ for all i. We assume that different dependency pair occurrences do not own jointly variables without loss of generality.*

Theorem 3.1.5 *[1, 2] A TRS R is not terminating iff there exists an infinite dependency chain of R.*

Proof.

($\Leftarrow$) Let $\langle u_0^\#, v_0^\#\rangle\langle u_1^\#, v_1^\#\rangle\langle u_2^\#, v_2^\#\rangle\cdots$ be an infinite dependency chain with a substitution θ such that $v_i^\#\theta \xrightarrow{*} u_{i+1}^\#\theta$ for all i. Since there exist C_i $(i = 0, 1, 2, \ldots)$ such that $u_i \to C_i[v_i] \in R$, there exists an infinite reduction sequence $u_0\theta \to C_0[v_0]\theta \xrightarrow{*} C_0[u_1]\theta \to C_0[C_1[v_1]]\theta \xrightarrow{*} C_0[C_1[u_2]]\theta \to C_0[C_1[C_2[v_2]]]\theta \xrightarrow{*} \cdots$.

($\Rightarrow$) Let t be a minimal size counterexample, i.e., t is a minimal size in all non-terminating terms. Since any proper subterm of t is terminating, there is a rule $l_0 \to C_0[r_0] \in R$ and a substitution θ such that $t^\# \xrightarrow{*} l_0^\#\theta$ and $r_0\theta$ is a minimal size counterexample in all subterms of $C_0[r_0]\theta$. From the minimality of t, $x\theta$ is terminating for all $x \in Var(r_0) \subseteq Var(l_0)$. From the minimality of $r_0\theta$, it follows that $(r_0)_\varepsilon \in DF(R)$. Thus, $\langle l_0^\#, r_0^\#\rangle$ is a dependency pair. Applying similar procedure to $r_0\theta$, we get a rule $l_1 \to C_1[r_1] \in R$ and a dependency pair $\langle l_1^\#, r_1^\#\rangle$ such that $r_0^\#\theta \xrightarrow{*} l_1^\#\theta$ and $r_1\theta$ is a minimal size counterexample in all subterms of $C_1[r_1]\theta$. Applying this procedure repeatedly, we obtain an infinite dependency chain $\langle l_0^\#, r_0^\#\rangle\langle l_1^\#, r_1^\#\rangle\langle l_2^\#, r_2^\#\rangle\cdots$. □

This theorem is the key in the dependency pair method. All results of dependency pairs are essentially based on this theorem. In the next section, we will introduce proving methods for termination using dependency pairs.

3.2 Proving Termination by Dependency Pairs

The method of dependency pairs compares rewrite rules and dependency pairs by a weak reduction order or by a weak reduction pair instead of a reduction order. In this section, we firstly introduce the notion of weak reduction orders and its application for proving termination. Next, we introduce the notion of weak reduction pairs, which is a generalization of weak reduction orders. Lastly, we introduce two methods to design weak reduction orders and weak reduction pairs. One is the argument filtering method[5, 23, 46]. Another one is the polynomial interpretation method [23, 43, 52].

3.2.1 Weak Reduction Order

In this subsection, we introduce the notion of weak reduction orders and methods for proving termination using dependency pairs.

Definition 3.2.1 *A quasi-order $\gtrsim$ is a weak reduction order if $\gtrsim$ is monotonic and stable, and its strict part $\gtrsim$ is well-founded and stable.*

The set of dependency pairs itself is a term rewriting system. Thus, the union of a term rewriting system and its dependency pairs gives an extended term rewriting system.

Lemma 3.2.2 *[5, 45] R is terminating if and only if $R \cup DP^{\#}(R)$ is terminating.*

Proof. ($\Leftarrow$) Trivial. ($\Rightarrow$) We denote by $|t|_{\#}$ the maximal nesting number of marked symbols in t. We write $C[\![t_1,\ldots,t_n]\!]$ if each root symbol of t_i is a marked symbol and C ($\not\equiv \Box$) has no marked symbol. We denote $(C[\![t_1,\ldots,t_n]\!])^{\#}$ by $C^{\#}[\![t_1,\ldots,t_n]\!]$. Let $R' = R \cup DP^{\#}(R)$. We prove that any t is terminating in R' by induction on $|t|_{\#}$. In the case $|t|_{\#} = 0$, it is trivial from termination of R. Suppose that $|t|_{\#} > 0$. We assume that t is not terminating in R'.

(i) $(t)_{\varepsilon}$ is a marked symbol.
Let $t \equiv C_1^{\#}[\![t_{11},\ldots,t_{1n_1}]\!]$. From induction hypothesis, each t_{1j} is terminating in R'. Since each marked symbol occurs only at the root positions of paired terms in $DP^{\#}(R)$, any infinite reduction sequence starting from t is expressed by $t \equiv C_1^{\#}[\![t_{11},\ldots,t_{1n_1}]\!] \xrightarrow[R']{*} C_1^{\#}[\![t'_{11},\ldots,t'_{1n_1}]\!] \xrightarrow[R']{} C_2^{\#}[\![t_{21},\ldots,t_{2n_2}]\!] \xrightarrow[R']{*} C_2^{\#}[\![t'_{21},\ldots,t'_{2n_2}]\!] \xrightarrow[R']{} \cdots$ such that $\{t'_{i1},\ldots,t'_{in_i}\} \supseteq \{t_{i+1,1},\ldots,t_{i+1,n_{i+1}}\}$ and $t_{ij} \xrightarrow[R']{*} t'_{ij}$ for all i and j. From the distribution of occurrences of marked symbols, we obtain an infinite reduction sequence $C_1^{\#} \xrightarrow[R']{} C_2^{\#} \xrightarrow[R']{} C_3^{\#} \xrightarrow[R']{} \cdots$. Since R is terminating, there are some $i_1, i_2, \ldots$ such that $C_{i_1}^{\#} \xrightarrow[R]{*} C_{i_2}^{\#} \xrightarrow[DP^{\#}(R)]{} C_{i_3}^{\#} \xrightarrow[R]{*} C_{i_4}^{\#} \xrightarrow[DP^{\#}(R)]{} C_{i_5}^{\#} \xrightarrow[R]{*} C_{i_6}^{\#} \xrightarrow[DP^{\#}(R)]{} \cdots$. Since rules of $DP^{\#}(R)$ are applied at only the root position in this infinite reduction sequence, there are some $\check{C}_i$ such that $C_{i_1} \xrightarrow[R]{*} C_{i_2} \xrightarrow[R]{} \check{C}_3[C_{i_3}] \xrightarrow[R]{*} \check{C}_3[C_{i_4}] \xrightarrow[R]{} \check{C}_3[\check{C}_5[C_{i_5}]] \xrightarrow[R]{*} \check{C}_3[\check{C}_5[C_{i_6}]] \xrightarrow[R]{} \cdots$. It is a contradiction to the termination of R.

(ii) $(t)_{\varepsilon}$ is an unmarked symbol.
Let $t \equiv C_1[\![t_{11},\ldots,t_{1n_1}]\!]$. From $|t_{1j}|_{\#} \leq |t|_{\#}$ and (i), each t_{1j} is terminating in R'. Therefore, it is a contradiction as similar to (i). □

Theorem 3.2.3 *[5] For any TRS R, the following two properties are equivalent.*

1. R is terminating.
2. There exists a weak reduction order $\gtrsim$ such that $\gtrsim$ is compatible with R and its strict part $\gnsim$ is compatible with $DP^{\#}(R)$.

Proof.

(1 $\Rightarrow$ 2) From Lemma 3.2.2, $R \cup DP^{\#}(R)$ is terminating. Thus, there exists a reduction order $>$ compatible with $R \cup DP^{\#}(R)$. Let $\geq$ be $> \cup \equiv$. Then it is trivial that $\geq$ is a weak reduction order such that it is compatible with R and its strict part is compatible with $DP^{\#}(R)$.

(2 $\Rightarrow$ 1) We assume that R is not terminating. From Theorem 3.1.5, there is an infinite dependency chain $\langle u_0^{\#}, v_0^{\#}\rangle\langle u_1^{\#}, v_1^{\#}\rangle\langle u_2^{\#}, v_2^{\#}\rangle \cdots$ with a substitution θ such that $v_i^{\#}\theta \xrightarrow[R]{*} u_{i+1}^{\#}\theta$ for all i. It follows that $\xrightarrow[R]{*} \subseteq \gtrsim$ from the assumption, the stability, the transitivity and the monotonicity of $\gtrsim$. For any i, we have $u_i^{\#}\theta \gnsim v_i^{\#}\theta$ from the assumption and the stability of $\gnsim$. Therefore, we get an infinite decreasing sequence $u_0^{\#}\theta \gnsim v_0^{\#}\theta \gtrsim u_1^{\#}\theta \gnsim v_1^{\#}\theta \gtrsim u_2^{\#}\theta \gnsim v_2^{\#}\theta \cdots$. It is a contradiction. □

3.2.2 Weak Reduction Pair

The method of dependency pairs is useful for not only proving termination but also analyzing other proving methods for termination. In order to analyze transformation methods called elimination transformations, we need extend the notion of weak reduction order to that of weak reduction pair. In this subsection, we introduce the notion of weak reduction pair. About analyzing elimination transformations, we will discuss in Chapter 5.

Definition 3.2.4 *A pair $(\gtrsim, >)$ of binary relations on terms is a weak reduction pair*[1] *if it satisfies the following three conditions:*

- $\gtrsim$ is monotonic and stable.
- $>$ is stable and well-founded.
- $(\gtrsim, >)$ is compatible, i.e., $\gtrsim \circ > \subseteq >$ or $> \circ \gtrsim \subseteq >$.

In the above definition, we do not assume that $\gtrsim$ is a quasi-order or $>$ is a strict order. This simplifies the design of a weak reduction pair. We should mention that this simplification does not lose the generality of our definition, because for a given weak reduction pair $(\gtrsim, >)$ we can make a weak reduction pair $(\gtrsim^*, >^+)$ in which $\gtrsim^*$ is a quasi-order and $>^+$ is a strict order. Note that $>$ is a reduction order if and only if $(>, >)$ is a weak reduction pair, and $(\gtrsim, \gtrsim)$ is a weak reduction pair for all weak reduction order $\gtrsim$.

Theorem 3.2.5 *[46] For any TRS R, the following properties are equivalent.*

1. TRS R is terminating.
2. There exists a weak reduction pair $(\gtrsim, >)$ such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP(R)$.
3. There exists a weak reduction pair $(\gtrsim, >)$ such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP^{\#}(R)$.

Proof. For the case $(1 \Rightarrow 2)$, we define $\gtrsim$ by $\xrightarrow[R]{*}$, and $s > t$ by $s \not\equiv t$ and $s \xrightarrow[R]{*} C[t]$ for some C. Then, it is easily shown that $(\gtrsim, >)$ is a weak reduction pair satisfying the conditions. For the case $(2 \Rightarrow 3)$, it is easily shown by identifying $f^{\#}$ with f. For the case $(3 \Rightarrow 1)$, as similar to the proof of Theorem 3.2.3 $(2 \Rightarrow 1)$. $\square$

In general for any terminating TRS R it is still open whether there exists a weak reduction order $\gtrsim$ such that $\gtrsim$ is compatible with R and its strict part $\gtrsim$ is compatible with unmarked dependency pairs $DP(R)$. The proof for $(1 \Rightarrow 2)$ in Theorem 3.2.3 is based on Lemma 3.2.2. However, the same proof method can not work well for unmarked dependency pairs, because the termination of R does not ensure that of $R \cup DP(R)$. In fact, for the terminating TRS $R = \{f(f(x)) \to f(g(x)),\ g(x) \to h(f(x))\}$, $R \cup DP(R)$ is not terminating. The fact maintains the usefulness of marking technique. On the other hand, the above theorem guarantees the existence of a weak reduction pair $(\gtrsim, >)$ such that $\gtrsim$ is compatible with R and its strict part $>$ is compatible with unmarked dependency pairs $DP(R)$. This fact indicates that weak reduction pairs have an extra power as compared with weak reduction orders.

[1] The notion of weak reduction pairs is a generalization of the stable-strict relation $\succ^{ss}$ in [23] and the irreflexive order $>_{lift}$ lifted from ground terms to non-ground terms in [5].

3.2.3 Argument Filtering Method

The argument filtering method was first proposed by Arts and Giesl [5, 23]. To analyze other proving methods for termination, the method was slightly improved by combining the subterm relation [46]. We firstly explain the notion of argument filtering function.

Definition 3.2.6 *An argument filtering function is a function π such that for any $f \in \Sigma^{\#}$, $\pi(f)$ is either an integer i or a list of integers $[i_1, \ldots, i_m]$ $(m \geq 0)$, where those integers $i, i_1, \ldots, i_m$ are positive and not more than $arity(f)$. Suppose that $\Sigma^{\#}_{\pi} = \{f \in \Sigma^{\#} \mid \pi(f)$ is a list $\}$. We can naturally extend π from $\mathcal{T}(\Sigma^{\#}, \mathcal{V})$ to $\mathcal{T}(\Sigma^{\#}_{\pi}, \mathcal{V})$ as follows:*

$$\begin{cases} \pi(x) & = & x & \\ \pi(f(t_1, \ldots, t_n)) & = & \pi(t_i) & \textit{if } \pi(f) = i \\ \pi(f(t_1, \ldots, t_n)) & = & f(\pi(t_{i_1}), \ldots, \pi(t_{i_m})) & \textit{if } \pi(f) = [i_1, \ldots, i_m] \end{cases}$$

We denote by $\pi(\theta)$ the substitution defined as $\pi(\theta)(x) = \pi(\theta(x))$ for all $x \in \mathcal{V}$.

We hereafter assume that if $\pi(f)$ is not defined explicitly then it is intended to be $[1, \ldots, arity(f)]$.

Definition 3.2.7 *Let $>$ be a binary relation on terms. We define $s >^{sub} t$ by $s \not\equiv t$ and $s \geq C[t]$ for some C.*

Lemma 3.2.8 *Let $>$ be a well-founded and monotonic binary relation on terms. Then the binary relation $>^{sub}$ is well-founded.*

Proof. We assume that there exists an infinite decreasing sequence $t_0 >^{sub} t_1 >^{sub} t_2 >^{sub} \cdots$. Then there exist contexts C_i $(i = 1, 2, \ldots)$ such that $t_i \geq C_{i+1}[t_{i+1}]$ for any i. From the monotonicity, $t_0 \geq C_1[t_1] \geq C_1[C_2[t_2]] \cdots$. From the well-foundedness, there is some k such that $C_1[\cdots C_k[t_k] \cdots] \equiv C_1[\cdots C_{k+1}[t_{k+1}] \cdots] \equiv C_1[\cdots C_{k+2}[t_{k+2}] \cdots] \equiv \cdots$. Thus, there is some m such that $C_m \equiv \Box$. Hence, it follows that $t_{m-1} \equiv t_m$. It is a contradiction. $\Box$

Definition 3.2.9 *Let $>$ be a reduction order and π an argument filtering function. We define $s \gtrsim_{\pi} t$ by $\pi(s) \geq \pi(t)$, and $s >_{\pi} t$ by $\pi(s) >^{sub} \pi(t)$.*

Note that $s \gtrsim_{\pi} t$ iff $\pi(s) > \pi(t)$, and $\gtrsim_{\pi} = >_{\pi}$ if $>$ is an simplification order.

Lemma 3.2.10 $\pi(\theta)(\pi(t)) \equiv \pi(t\theta)$.

Proof. We prove this lemma by induction on t. The case $t \equiv x \in \mathcal{V}$ is trivial. Suppose that $t \equiv f(t_1, \ldots, t_n)$. If $\pi(f) = i$ then

$$\pi(\theta)(\pi(f(t_1, \ldots, t_n))) \equiv \pi(\theta)(\pi(t_i)) \equiv \pi(t_i\theta) \equiv \pi(f(t_1\theta, \ldots, t_n\theta)) \equiv \pi(f(t_1, \ldots, t_n)\theta).$$

If $\pi(f) = [i_1, \ldots, i_m]$ then

$$\begin{aligned} \pi(\theta)(\pi(f(t_1, \ldots, t_n))) & \equiv \pi(\theta)(f(\pi(t_{i_1}), \ldots, \pi(t_{i_m}))) \\ & \equiv f(\pi(\theta)(\pi(t_{i_1})), \ldots, \pi(\theta)(\pi(t_{i_m}))) \\ & \equiv f(\pi(t_{i_1}\theta), \ldots, \pi(t_{i_m}\theta)) \\ & \equiv \pi(f(t_1\theta, \ldots, t_n\theta)) \\ & \equiv \pi(f(t_1, \ldots, t_n)\theta). \end{aligned}$$

$\Box$

Lemma 3.2.11 $\pi(C)[\pi(t)] \equiv \pi(C[t])$.

Proof. We prove this lemma by induction on C. The case $C \equiv \Box$ is trivial. Suppose that $C \equiv f(\ldots, t_{i-1}, C', t_{i+1}, \ldots)$. We have the following two cases:

- $\pi(f) = j$: The case $i \neq j$ is trivial. Suppose that $i = j$. Then,

$$\begin{array}{rcl} \pi(C)[\pi(t)] & \equiv & \pi(f(\ldots, t_{i-1}, C', t_{i+1}, \ldots))[\pi(t)] \\ & \equiv & \pi(C')[\pi(t)] \\ & \equiv & \pi(C'[t]) \\ & \equiv & \pi(f(\ldots, t_{i-1}, C'[t], t_{i+1}, \ldots)) \\ & \equiv & \pi(C[t]). \end{array}$$

- $\pi(f) = [i_1, \ldots, i_m]$: The case $i \notin \pi(f)$ is trivial. Suppose that $i \in \pi(f)$. Then,

$$\begin{array}{rcl} \pi(C)[\pi(t)] & \equiv & \pi(f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_n)[\pi(t)] \\ & \equiv & f(t_{i_1}, \ldots, \pi(C'), \ldots, t_{i_m})[\pi(t)] \\ & \equiv & f(t_{i_1}, \ldots, \pi(C')[\pi(t)], \ldots, t_{i_m}) \\ & \equiv & f(t_{i_1}, \ldots, \pi(C'[\pi(t)]), \ldots, t_{i_m}) \\ & \equiv & \pi(f(t_1, \ldots, t_{i-1}, C'[t], t_{i+1}, \ldots, t_n)) \\ & \equiv & \pi(C[t]). \end{array}$$

$\Box$

Theorem 3.2.12 *[5, 23] Let $>$ be a reduction order and π an argument filtering function. Then $\gtrsim_\pi$ is a weak reduction order*[2]*.*

Proof.

- ($\gtrsim_\pi$ is a quasi-order): It is trivial.
- (The monotonicity of $\gtrsim_\pi$): From Lemma 3.2.11, $s \gtrsim_\pi t \Rightarrow \pi(s) \geq \pi(t) \Rightarrow \pi(C)[\pi(s)] \geq \pi(C)[\pi(t)] \Rightarrow \pi(C[s]) \geq \pi(C[t]) \Rightarrow C[s] \gtrsim_\pi C[t]$.
- (The stability of $\gtrsim_\pi$): From Lemma 3.2.10, $s \gtrsim_\pi t \Rightarrow \pi(s) \geq \pi(t) \Rightarrow \pi(\theta)(\pi(s)) \geq \pi(\theta)(\pi(t)) \Rightarrow \pi(s\theta) \geq \pi(t\theta) \Rightarrow s\theta \gtrsim_\pi t\theta$.
- (The well-foundedness of $\gtrsim_\pi$): We assume that there exists an infinite decreasing sequence $t_0 \gtrsim_\pi t_1 \gtrsim_\pi t_2 \gtrsim_\pi \cdots$. Then, it follows that $\pi(t_0) > \pi(t_1) > \pi(t_2) > \cdots$. It is a contradiction to the well-foundedness of $>$.
- (The stability of $\gtrsim_\pi$): From Lemma 3.2.10, $s \gtrsim_\pi t \Rightarrow \pi(s) > \pi(t) \Rightarrow \pi(\theta)(\pi(s)) > \pi(\theta)(\pi(t)) \Rightarrow \pi(s\theta) > \pi(t\theta) \Rightarrow s\theta \gtrsim_\pi t\theta$. $\Box$

Theorem 3.2.13 *[46] Let $>$ be a reduction order and π an argument filtering function. Then $(\gtrsim_\pi, >_\pi)$ is a weak reduction pair.*

[2]For designing a weak reduction order, the argument filtering method is essentially a special form of recursive program schema (RPS) [11].

Proof. In the proof of Theorem 3.2.12, we have already shown the stability and the monotonicity of $\gtrsim_\pi$.

- (The well-foundedness of $>_\pi$): We assume that there exists an infinite decreasing sequence $t_0 >_\pi t_1 >_\pi t_2 >_\pi \cdots$. Then $\pi(t_0) >^{sub} \pi(t_1) >^{sub} \pi(t_2) >^{sub} \cdots$. It is a contradiction to Lemma 3.2.8.

- (The stability of $>_\pi$): From Lemmas 3.2.10 and 3.2.11, $s >_\pi t \Rightarrow \pi(s) \geq C[\pi(t)]$ $\Rightarrow \pi(\theta)(\pi(s)) \geq \pi(\theta)(C[\pi(t)]) \Rightarrow \pi(s\theta) \geq C'[\pi(t\theta)]$ where $C' \equiv \pi(C\theta)$. We assume that $\pi(s\theta) \equiv \pi(t\theta)$. Then, $\pi(s\theta) \geq C'[\pi(s\theta)] \geq C'[C'[\pi(s\theta)]] \cdots$. Thus, $C' \equiv \Box$ and $C \equiv \Box$. Hence, it follows that $\pi(s) \geq \pi(t)$. Since $\pi(s) \not\equiv \pi(t)$, $\pi(s) > \pi(t)$. From the stability of $>$, $\pi(\theta)(\pi(s)) > \pi(\theta)(\pi(t))$. From Lemma 3.2.10, $\pi(s\theta) > \pi(t\theta)$. It is a contradiction to $\pi(s\theta) \equiv \pi(t\theta)$.

- ($\gtrsim_\pi \circ >_\pi \subseteq >_\pi$): Let $t_0 \gtrsim_\pi t_1 >_\pi t_2$. Then $\pi(t_1) \not\equiv \pi(t_2)$ and $\pi(t_0) \geq \pi(t_1) \geq C[\pi(t_2)]$ for some C. It follows $\pi(t_0) \geq C[\pi(t_2)]$. It suffices to show that $\pi(t_0) \not\equiv \pi(t_2)$. Assume that $\pi(t_0) \equiv \pi(t_2)$. Then $\pi(t_2) \geq \pi(t_1) \geq C[\pi(t_2)]$. Since $\pi(t_2) \geq C[\pi(t_2)]$, it follows that $C \equiv \Box$. Moreover it follows that $\pi(t_1) \equiv \pi(t_2)$. It is a contradiction. $\Box$

3.2.4 Polynomial Interpretation

In this subsection, using polynomial interpretations, we introduce methods to design weak reduction orders and weak reduction pairs.

Theorem 3.2.14 *[43] Let $A \subseteq \mathcal{N}\backslash\{0\}$. We define the polynomial quasi-order $\gtrsim_A$ as follows:*

$$s \gtrsim_A t \iff^{\text{def}} \forall\sigma([\![s]\!]_\sigma > [\![t]\!]_\sigma) \text{ or } \forall\sigma([\![s]\!]_\sigma = [\![t]\!]_\sigma).$$

Then, the polynomial quasi-order $\gtrsim_A$ is a weak reduction order.

Proof. As similar to Theorem 3.2.12 based on Proposition 2.4.4. $\Box$

The above polynomial quasi-order cannot treat the value 0, because in the case $0 \in A$ the above polynomial quasi-order $\gtrsim_A$ is not monotonic. For example, let $a_A = 2$, $b_A = 1$, $f_A(X, Y) = XY$, $C \equiv f(\Box, x)$ and $0 \in A$. Then, it follows that $a \gtrsim_A b$ and $C[a] \not\gtrsim_A C[b]$. On the other hand, the following polynomial interpretation method can treat the value 0.

Proposition 3.2.15 *[23, 52] Let $A \subseteq \mathcal{N}$. We define the pair $(\gtrsim_A, >_A)$ as follows:*

$$s \gtrsim_A t \iff^{\text{def}} \forall\sigma([\![s]\!]_\sigma \geq [\![t]\!]_\sigma), \qquad s >_A t \iff^{\text{def}} \forall\sigma([\![s]\!]_\sigma > [\![t]\!]_\sigma).$$

Then, the pair $(\gtrsim_A, >_A)$ is a weak reduction pair.

Note that this binary relation $\gtrsim_A$ is not a weak reduction order, because its strict part $\gtrsim_A$ is not stable. For example, let $a_A = min(A)$, $f_A = X$ and $\theta(x) = a$. Then, it follows that $f(x) \gtrsim_A a$ and $f(x)\theta \equiv f(a) \not\gtrsim_A a$.

These method by polynomial interpretations can easily extend to methods by arbitrary algebraic interpretations.

3.3 Hierarchy of Dependency Pairs

A dependency pair can be regarded as a rewrite rule, because the pair of terms satisfies the variable condition. Hence, we can regard $DP(R)$ and $DP^{\#}(R)$ as term rewriting systems, respectively. However, TRS R, $DP(R)$, $DP^{\#}(R)$ and the union of them do not accurately coincide on the termination property. In fact, for the TRS $R_1 = \{f(a) \to f(b),\ b \to a\}$, it is trivial that R_1 is not terminating while $DP(R_1) = \{f(a) \to f(b),\ f(a) \to b\}$ is terminating. In this section, we investigate relations between TRSs, their dependency pairs and the union of them. Moreover, we discuss the effectiveness of the marking technique. As a result, we show a hierarchy of dependency pairs, which indicate a class of TRSs in which the marking technique effectively works for proving termination.

Theorem 3.3.1 *[45] The following inclusion relations hold:*

(1) R is simply terminating if and only if $R \cup DP(R)$ is simply terminating.

(2) If $R \cup DP(R)$ is simply terminating then $R \cup DP(R)$ is terminating.

(3) If $R \cup DP(R)$ is terminating then R and $DP(R)$ are terminating.

(4) R is terminating if and only if $R \cup DP^{\#}(R)$ is terminating.

(5) If $R \cup DP^{\#}(R)$ is terminating then $DP^{\#}(R)$ is terminating.

(6) If $DP(R)$ is terminating then $DP^{\#}(R)$ is terminating.

Proof. The cases (2,3,5) are trivial, because all sub-TRSs of a terminating TRS are terminating.

(1) ($\Leftarrow$) Trivial. ($\Rightarrow$) We define $s > t$ by $s \xrightarrow[R \cup Emb]{+} t$. Since R is simply terminating, $>$ is a simplification order. It is easily checked that $>$ is compatible with $R \cup DP(R)$. Therefore, $R \cup DP(R)$ is simply terminating.

(4) This case is Lemma 3.2.2.

(6) Assume that $DP^{\#}(R)$ is not terminating and $t_0 \xrightarrow[DP^{\#}(R)]{} t_1 \xrightarrow[DP^{\#}(R)]{} t_2 \xrightarrow[DP^{\#}(R)]{} \cdots$. Let t'_i be the term obtained by erasing marking in t_i. Then, it is clear that $t'_0 \xrightarrow[DP(R)]{} t'_1 \xrightarrow[DP(R)]{} t'_2 \xrightarrow[DP(R)]{} \cdots$. It is a contradiction to the termination of $DP(R)$. $\square$

Theorem 3.3.2 *[45] There exist TRSs R_i such the following:*

(1) $DP(R_1)$ is terminating but R_1 is not terminating.

(2) $DP^{\#}(R_2)$ is terminating but R_2 and $DP(R_2)$ are not terminating.

(3) R_3 is terminating but $DP(R_3)$ is not terminating.

(4) R_4 and $DP(R_4)$ are terminating but $R_4 \cup DP(R_4)$ is not terminating.

(5) $R_5 \cup DP(R_5)$ is terminating but R_5 is not simply terminating.

Proof. For each termination proof, we use the recursive path order $>$ as a reduction order, and we extend $>$ to a weak reduction order $\gtrsim_\pi$ by the argument filtering method.

(1) Consider the following TRSs R_1 and R_1':

$$R_1 = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ b & \to & a \end{array} \right. \qquad R_1' = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ f(a) & \to & b \end{array} \right.$$

Here, R_1' corresponds to $DP(R_1)$. Let $a \rhd b$. Then $>$ is compatible with R_1'. Hence, R_1' is terminating. However, R_1 is not terminating because of $f(a) \underset{R_1}{\to} f(b) \underset{R_1}{\to} f(a)$.

(2) Consider the following TRSs R_2, R_2' and R_2'':

$$R_2 = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ g(x) & \to & h(f(x)) \\ h(x) & \to & x \end{array} \right.$$

$$R_2' = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ f(f(x)) & \to & g(x) \\ g(x) & \to & h(f(x)) \\ g(x) & \to & f(x) \end{array} \right. \qquad R_2'' = \left\{ \begin{array}{rcl} \check{f}(f(x)) & \to & \check{f}(g(x)) \\ \check{f}(f(x)) & \to & \check{g}(x) \\ \check{g}(x) & \to & \check{h}(f(x)) \\ \check{g}(x) & \to & \check{f}(x) \end{array} \right.$$

Here, R_2' and R_2'' correspond to $DP(R_2)$ and $DP^\#(R_2)$, respectively. Let $\pi(\check{h}) = []$, $f \rhd g$, $f \rhd \check{g} \rhd \check{f}$ and $\check{g} \rhd \check{h}$. Then $\gtrsim_\pi$ is compatible with R_2'', and $\gtrsim_\pi$ is compatible with $DP(R_2'')$. Hence, R_2'' is terminating. However, R_2 and R_2' are not terminating because of $f(f(x)) \underset{R_2}{\to} f(g(x)) \underset{R_2}{\to} f(h(f(x))) \underset{R_2}{\to} f(f(x))$ and $f(f(x)) \underset{R_2'}{\to} f(g(x)) \underset{R_2'}{\to} f(f(x))$.

(3) Consider the following TRSs R_3 and R_3':

$$R_3 = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ g(x) & \to & h(f(x)) \end{array} \right. \qquad R_3' = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ f(f(x)) & \to & g(x) \\ g(x) & \to & f(x) \end{array} \right.$$

Here, R_3' corresponds to $DP(R_3)$. Let $\pi(h) = []$, $f \rhd g \rhd h$ and $f \rhd g^\# \rhd f^\#$. Then $\gtrsim_\pi$ is compatible with R_3, and $\gtrsim_\pi$ is compatible with $DP^\#(R_3)$. Hence, R_3 is terminating. However, R_3' is not terminating because of $f(f(x)) \underset{R_3'}{\to} f(g(x)) \underset{R_3'}{\to} f(f(x))$.

(4) Consider the following TRSs R_4, R_4' and R_4'':

$$R_4 = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ b & \to & g(h(a)) \\ h(x) & \to & x \end{array} \right.$$

$$R_4' = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ f(a) & \to & b \\ b & \to & h(a) \end{array} \right. \qquad R_4'' = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ b & \to & g(h(a)) \\ h(x) & \to & x \\ f(a) & \to & b \\ b & \to & h(a) \end{array} \right.$$

Here, R_4' and R_4'' correspond to $DP(R_4)$ and $R_4 \cup DP(R_4)$, respectively. Let $\pi(g) = \pi(h^\#) = []$, $a \rhd b \rhd g$ and $f^\# \rhd b^\# \rhd h^\#$. Then $\gtrsim_\pi$ is compatible with R_4, and $\gtrsim_\pi$ is compatible with $DP^\#(R_4)$. Hence, R_4 is terminating. Let $\pi(h) = []$ and $a \rhd b \rhd h$. Then $\gtrsim_\pi$ is compatible with R_4', and $\gtrsim_\pi$ is compatible with $DP(R_4')$. Hence, R_4' is terminating. However, R_4'' is not terminating because of $f(a) \underset{R_4''}{\to} f(b) \underset{R_4''}{\to} f(h(a)) \underset{R_4''}{\to} f(a)$.

(5) Consider the following TRSs R_5 and R_5':

$$R_5 = \left\{ \begin{array}{lcl} f(f(x)) & \to & f(g(f(x))) \end{array} \right. \qquad R_5' = \left\{ \begin{array}{lcl} f(f(x)) & \to & f(g(f(x))) \\ f(f(x)) & \to & f(x) \end{array} \right.$$

Here, R_5' corresponds to $R_5 \cup DP(R_5)$. Let $\pi(g) = []$ and $f \rhd g$. Then $\gtrsim_\pi$ is compatible with R_5', and $\gtrsim_\pi$ is compatible with $DP(R_5')$. Hence, R_5' is terminating. However, R_5 is not simply terminating because of $f(f(x)) \underset{R_5}{\to} f(g(f(x))) \underset{Emb}{\to} f(f(x))$.

□

From Theorems 3.3.1 and 3.3.2, we can demonstrate the hierarchy as follows: (Figure 3.1)

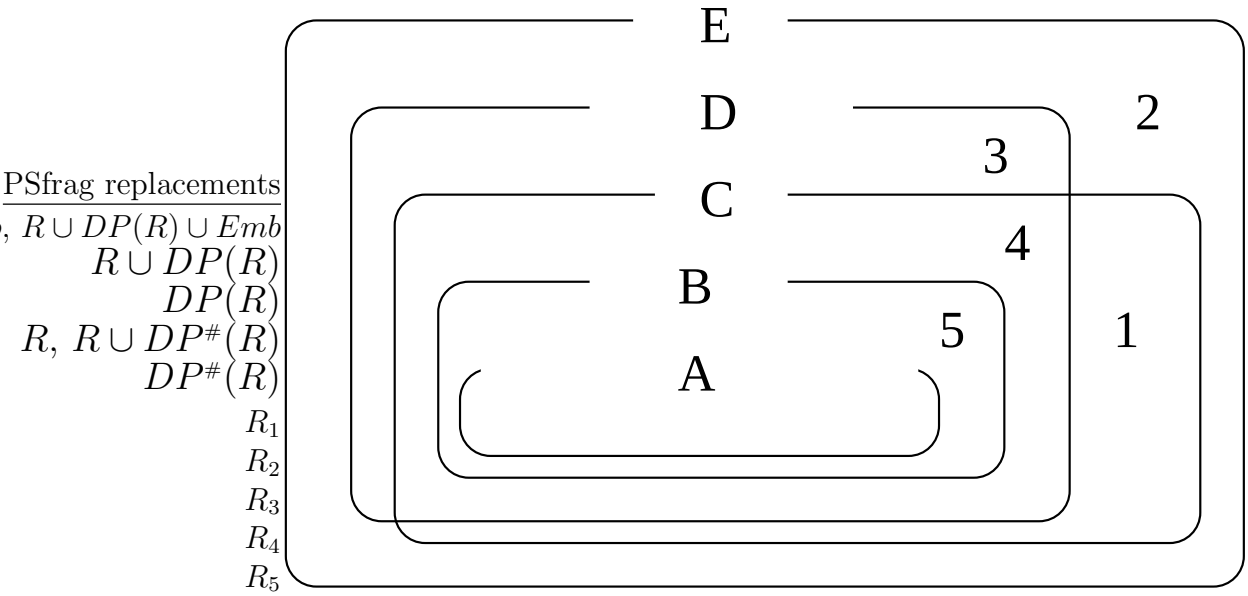

Figure 3.1: The Hierarchy of Dependency Pairs

This hierarchy makes clear the class of TRSs in which the marking technique works effectively for proving termination. Since R and $R \cup DP(R)$ make the difference class in this hierarchy, we conclude that marking is effective for the class that R is terminating but $R \cup DP(R)$ is not terminating. In fact, in Theorem 3.3.2, marking is only used in the termination proof of R_3 and R_4, which belong to this class.

3.4 Dependency Graph

For proving termination by Theorem 3.2.3, all dependency pairs must be checked. In order to removing unnecessary dependency pairs, the notion of dependency graphs introduced by Arts and Giesl is very useful [1, 2].

Definition 3.4.1 *We define a directed graph as a set of nodes and a set of arcs, each arc leading from a node to a node. A path is a sequence of nodes in which successive nodes are connected by arcs in the graph. A cycle is a path of length at least* 1 *in which no node is repeated except that the first and the last nodes are the same. We regard cycles* $n_1, \ldots, n_k$ *and* $n_i, \ldots, n_{k-1}, n_1, \ldots, n_i$ *as the same cycle. A non-empty set* N *of nodes is a cluster if for all nodes* $n, n' \in N$ *there exist* $n_1, n_2, \ldots, n_m \in N$ $(m \geq 0)$ *such that* $n, n_1, n_2, \ldots, n_m, n'$ *is a path in the graph.*

Notice that the length of a path $n, n_1, n_2, \ldots, n_m, n'$ in the above definition about cluster is at least 1.

Definition 3.4.2 *A dependency graph of* R *is a directed graph of which the nodes are dependency pairs, and there is an arc from* $\langle u^\#, v^\# \rangle$ *to* $\langle u'^\#, v'^\# \rangle$ *if* $\langle u^\#, v^\# \rangle \langle u'^\#, v'^\# \rangle$ *is a dependency chain.*

Theorem 3.4.3 *[1, 2] Let* R *be a TRS. If there exists a weak reduction order* $\gtrsim$ *such that*

- $l \gtrsim r$ *for all* $l \to r \in R$,
- $u^\# \gtrsim v^\#$ *for all* $\langle u^\#, v^\# \rangle$ *on a cycle in the dependency graph of* R*, and*
- $u^\# \gtrsim v^\#$ *for at least one* $\langle u^\#, v^\# \rangle$ *on each cycle in the dependency graph of* R*,*

then R *is terminating.*

Proof. Assume that R is not terminating. From Theorem 3.1.5, there exists an infinite dependency chain $\langle u_0^\#, v_0^\# \rangle \langle u_1^\#, v_1^\# \rangle \langle u_2^\#, v_2^\# \rangle \cdots$ with θ such that $v_i^\# \theta \xrightarrow{*} u_{i+1}^\# \theta$ for all i. Since the number of dependency pairs is finite, there exists a tail of this infinite dependency chain $\langle u_m^\#, v_m^\# \rangle \langle u_{m+1}^\#, v_{m+1}^\# \rangle \langle u_{m+2}^\#, v_{m+2}^\# \rangle \cdots$ in which all occurring dependency pairs occur infinitely often up to variable renaming. Here, $v_i^\# \theta \gtrsim u_{i+1}^\# \theta$ for all i. Since any $\langle u_i^\#, v_i^\# \rangle$ $(i \geq m)$ is on a cycle, $u_i^\# \theta \gtrsim v_i^\# \theta$. Since this tail is an infinite length path, infinite number cycles occur. Hence cases like $u_i^\# \theta \gtrsim v_i^\# \theta$ occur infinitely often. It is a contradiction. □

The reader might think that this theorem can be generalized by dividing for treating each cycle, that is, for each cycle C we allow to use different weak reduction order $\gtrsim_C$. Unfortunately this generalization is not sound. For example, consider the following TRS:

$$R = \{f(c, x, c) \to f(a, b, x),\ f(a, b, x) \to f(c, d, x),\ f(x, d, d) \to f(a, b, x)\}$$

Then, its dependency graph is Figure 3.2. In this graph, there are two cycles, that is, (1) does not construct a cycle. For each cycle (2) and (3), it is easy to design a weak reduction order by argument filtering method based on the recursive path order. However, R is not terminating because $f(a, b, c) \to f(c, d, c) \to f(a, b, d) \to f(c, d, d) \to f(a, b, c)$. To lead this conjecture to be sound, we need the notion of clusters. In fact, the cluster (1) is essential for existence of infinite dependency chains in the above examples. Note that similar problem is also caused even if we change the definition of cycles such repeating no arcs instead of nodes.

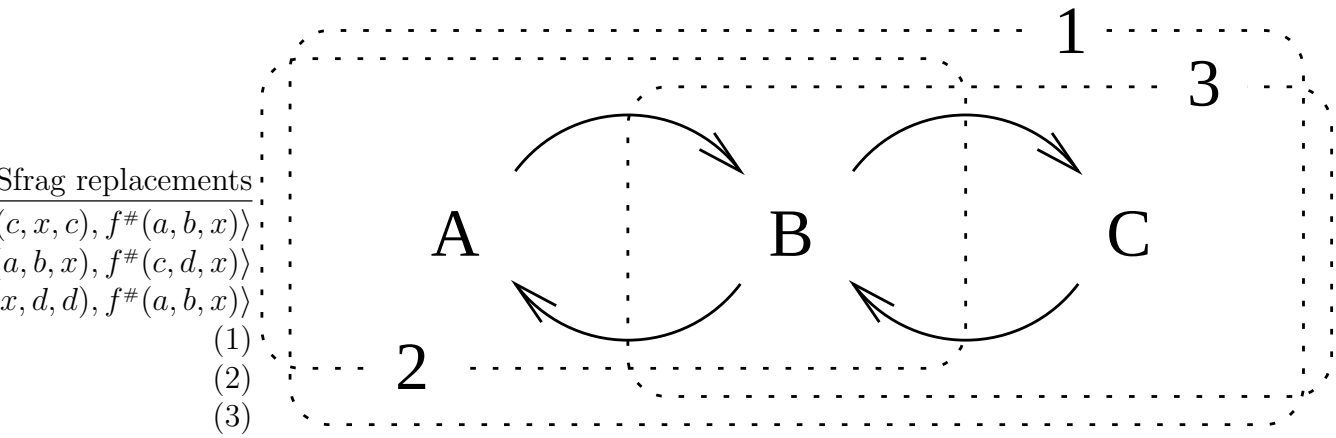

Figure 3.2: Dependency Graph

Theorem 3.4.4 *[4] Let R be a TRS. If for each cluster N in the dependency graph there exists a weak reduction order $\gtrsim_N$ such that*

- $l \gtrsim_N r$ *for all* $l \to r \in R$,
- $u^{\#} \gtrsim_N v^{\#}$ *for all* $\langle u^{\#}, v^{\#}\rangle$ *in* N, *and*
- $u^{\#} \gtrsim_N v^{\#}$ *for at least one* $\langle u^{\#}, v^{\#}\rangle$ *in* N,

then R is terminating.

Proof. Assume that R is not terminating. From Theorem 3.1.5, there exists an infinite dependency chain $\langle u_0^{\#}, v_0^{\#}\rangle\langle u_1^{\#}, v_1^{\#}\rangle\langle u_2^{\#}, v_2^{\#}\rangle\cdots$. Since the number of dependency pairs is finite, there exist dependency pairs occuring infinitely often, up to variable renaming. Let $N = \{\langle u_{k_1}^{\#}, v_{k_1}^{\#}\rangle, \ldots, \langle u_{k_n}^{\#}, v_{k_n}^{\#}\rangle\}$ be the set of all dependency pairs that occur infinitely often in this dependency chain. From the construction of N, N is a cluster. Moreover, there exists a number m such that $\langle u_m^{\#}, v_m^{\#}\rangle\langle u_{m+1}^{\#}, v_{m+1}^{\#}\rangle\cdots$ is constructed from dependency pairs in N, and any dependency pair in N occurs infinitely often in the chain. From the assumption, there exists a decreasing sequence $v_m^{\#}\theta \gtrsim_N u_{m+1}^{\#}\theta \gtrsim_N v_{m+1}^{\#}\theta \gtrsim_N u_{m+2}^{\#}\theta \gtrsim_N \cdots$, in which cases like $u_i^{\#}\theta \gtrsim_N v_i^{\#}\theta$ occur infinitely often. It leads to a contradiction. $\square$

Unfortunately, dependency graphs in general are not computable, because it is undecidable whether there is some substitution θ such that $v^{\#}\theta \xrightarrow{*} u'^{\#}\theta$ for two dependency pairs $\langle u^{\#}, v^{\#}\rangle$ and $\langle u'^{\#}, v'^{\#}\rangle$. To generate approximated dependency graphs, Arts and Giesl introduced the following algorithm.

Definition 3.4.5 *Let R be a TRS, t a term and $z_1, z_2, \ldots$ an infinite sequence of fresh variables. The function CAP and REN from terms to terms are inductively defined as follows:*

$$\begin{aligned} CAP(x) &= x \\ CAP(f(t_1, \ldots, t_n)) &= \begin{cases} z & \text{if } f \in DF(R) \\ f(CAP(t_1), \ldots, CAP(t_n)) & \text{if } f \notin DF(R) \end{cases} \\ REN(x) &= z \end{aligned}$$

$$REN(f(t_1, \ldots, t_n)) \quad = \quad f(REN(t_1), \ldots, REN(t_n))$$

where z represents the next fresh variable in the list of free variables.

Definition 3.4.6 *Let R be a TRS. The approximated dependency graph of R is a directed graph of which the nodes are dependency pairs, and there is an arc from $\langle u^\#, v^\# \rangle$ to $\langle u'^\#, v'^\# \rangle$ if $REN(CAP(v^\#))$ and $u'^\#$ are unifiable.*

Proposition 3.4.7 *[1, 2] Let R be a TRS. The approximated dependency graph of R is a subgraph of the dependency graph of R.*

Arts and Giesl proposed a more powerful approximation algorithm using narrowing technique [1, 3].

Chapter 4

AC-Dependency Pairs

In this chapter, we extend the notion of dependency pairs to AC-TRSs. It is impossible to directly apply the notion of dependency pairs to AC-TRSs. We show this difficulty through an example. To avoid this difficulty we introduce the notion of head parts in terms and show an analogy between the root positions in infinite reduction sequences by TRSs and the head positions in those by AC-TRSs. Indeed, this analogy is essential for the extension of dependency pairs to AC-TRSs. Based on this analogy, we define AC-dependency pairs and AC-dependency chains.

The method of AC-dependency pairs compares rewrite rules and AC-dependency pairs by a weak AC-reduction order or by a weak AC-reduction pair, which play an important role on this method, instead of AC-reduction orders. We introduce the notion of weak AC-reduction orders and weak AC-reduction pairs. We show their application for proving AC-termination. Before introducing AC-dependency pairs, we have studied no designing method for weak AC-reduction orders and weak AC-reduction pairs. Hence we introduce the argument filtering method and the polynomial interpretation method. The original idea of the argument filtering method for TRSs without AC-function symbols was first proposed by Arts and Giesl [5, 23]. To analyze other proving methods for termination, this method was slightly improved by combining the subterm relation [46]. We extend these methods to AC-TRSs. Our extension designs a weak AC-reduction order and a weak AC-reduction pair from an arbitrary AC-reduction order. Moreover, in order to strengthen the power of the argument filtering method, we improve the method in two directions. One is the lexicographic argument filtering method, which lexicographically combines argument filtering functions to compare AC-dependency pairs. Another one is an extension over multisets modulo AC.

Lastly, we introduce the notion of AC-dependency graphs. Since dependency graphs in general are not computable, some algorithms for generating approximated dependency graphs in TRSs were introduced [1, 2]. We also propose another algorithm for generating an approximated AC-dependency graph, using the techniques of Ω-reduction and Ω_V-reduction, which are introduced to analyze decidable call-by-need computations in TRSs [28, 55, 59]. Of course, our algorithm can also apply to TRSs, because TRSs are AC-TRSs without AC-symbols.

4.1 AC-Dependency Pair and AC-Dependency Chain

This section presents the notions of AC-dependency pairs and AC-dependency chains for AC-TRSs. In order to simplify the discussion, we first treat AC-dependency pairs without marked symbols, though the following discussion can be easily extended to AC-dependency pairs with marked symbols.

4.1.1 Unmarked AC-Dependency Pairs and Chains

The notion of dependency pairs cannot be directly applied to AC-TRSs. Consider the AC-TRS $R = \{f(x,x) \to f(0,1)\}$ with $\Sigma_{AC} = \{f\}$. Here, $DP^{\#}(R) = \{\langle f^{\#}(x,x), f^{\#}(0,1)\rangle\}$. The AC-TRS R is not AC-terminating, because

$$f(\underline{f(0,0)},1) \underset{R}{\to} f(f(0,1),1) \underset{AC}{\sim} f(0,\underline{f(1,1)}) \underset{R}{\to} f(0,f(0,1)) \underset{AC}{\sim} f(f(0,0),1).$$

However, there is no infinite dependency chain, i.e., $f^{\#}(0,1)\theta \underset{R/AC}{\overset{*}{\not\to}} f^{\#}(x,x)\theta$ for all θ. Thus the equivalency between the existence of infinite dependency chains and non-termination (Theorem 3.1.5) does not hold for AC-TRSs. Theorem 3.1.5 is proved from the fact that any infinite reduction sequence from a minimal size counterexample must include a reduction at the root position. However, this property does not hold for AC-TRSs. In fact, the infinite AC-reduction sequence in the above example does not include such the reduction though $f(f(0,0),1)$ is a minimal size counterexample in R. To avoid this difficulty, we introduce the notion of head positions, which behaves like the root position in a minimal size counterexample in TRSs.

Definition 4.1.1

$$\begin{array}{rcl} \mathcal{O}_{hd}(t) & = & \begin{cases} \{p \mid \forall q \preceq p,\ (t)_q = (t)_\varepsilon\} & \textit{if } (t)_\varepsilon \in \Sigma_{AC} \\ \{\varepsilon\} & \textit{if } (t)_\varepsilon \notin \Sigma_{AC} \end{cases} \\ T_{bd}(t) & = & \begin{cases} \{t|_p \mid (t)_p \neq (t)_\varepsilon,\ \forall q \prec p\ [(t)_q = (t)_\varepsilon]\} & \textit{if } (t)_\varepsilon \in \Sigma_{AC} \\ \{t|_i \mid 1 \leq i \leq arity((t)_\varepsilon)\} & \textit{if } (t)_\varepsilon \notin \Sigma_{AC} \end{cases} \end{array}$$

Definition 4.1.2

$$\begin{array}{rcll} s \overset{hd}{\to} t & \overset{\text{def}}{\Longleftrightarrow} & s \underset{AC}{\sim} s' \underset{R}{\to}_p t & \textit{for some } s' \textit{ and } p \in \mathcal{O}_{hd}(s') \\ s \overset{bd}{\to} t & \overset{\text{def}}{\Longleftrightarrow} & s \underset{AC}{\sim} s' \underset{R}{\to}_p t & \textit{for some } s' \textit{ and } p \notin \mathcal{O}_{hd}(s') \\ s \trianglerighteq_{hd} t & \overset{\text{def}}{\Longleftrightarrow} & s \underset{AC}{\sim} C[t]_p & \textit{for some } C[\ ]_p \textit{ and } p \in \mathcal{O}_{hd}(C[t]_p) \end{array}$$

For example, let $t \equiv f(f(0,1), g(f(2,3)))$ and $\Sigma_{AC} = \{f\}$. Then $\mathcal{O}_{hd}(t) = \{\varepsilon, 1\}$, $T_{bd}(t) = \{0, 1, g(f(2,3))\}$ and $f(f(0,1), g(f(2,3))) \trianglerighteq_{hd} f(0,1)$.

Definition 4.1.3 *Let R be an AC-TRS. The set $DP_{AC}(R)$ of unmarked AC-dependency pairs in R is defined by*

$$DP_{AC}(R) = DP(R) \cup \{\langle f(l,z), f(r,z)\rangle \mid l \to r \in R,\ (l)_\varepsilon = f \in \Sigma_{AC}\}$$

where z is a flesh variable. We call $\langle f(l,z), f(r,z)\rangle$ by unmarked extended dependency pair.

Example 4.1.4 *Let* $\Sigma_{AC} = \{add\}$ *and*

$$R = \left\{ \begin{array}{rcl} add(x,0) & \to & x \\ add(x,s(y)) & \to & s(add(x,y)) \end{array} \right.$$

Here, $DF(R) = \{add\}$*. There is one unmarked dependency pairs*

$$\langle add(x,s(y)), add(x,y)\rangle$$

and two unmarked extended dependency pairs

$$\langle add(add(x,0),z), add(x,z)\rangle, \quad \langle add(add(x,s(y)),z), add(s(add(x,y)),z)\rangle.$$

Definition 4.1.5 *A sequence of unmarked AC-dependency pairs* $\langle u_0, v_0\rangle\langle u_1, v_1\rangle\langle u_2, v_2\rangle \cdots$ *is an unmarked AC-dependency chain if there exists a substitution* θ *such that* $v_i\theta \xrightarrow{bd}{}^* \trianglerighteq_{hd} u_{i+1}\theta$ *for all* i*. We assume that different unmarked AC-dependency pair occurrences do not own jointly variables without loss of generality.*

Lemma 4.1.6 *Let* $s \equiv C[s_1, \ldots, s_n] \xrightarrow{bd} t$ *such that each* $s' \in T_{bd}(s)$ *is a subterm of some* s_i*. Then there are* $t_1, \ldots, t_n$ *such that* $C[t_1, \ldots, t_n] \underset{AC}{\sim} t$*, each* $t' \in T_{bd}(C[t_1, \ldots, t_n])$ *is a subterm of some* t_i*, and there is some* k *such that* $s_k \underset{R/AC}{\to} t_k$ *and* $s_j \equiv t_j$ *for all* $j \neq k$*.*

Proof. It is obvious from the definitions of $\xrightarrow{bd}$ and $\underset{AC}{\sim}$. □

Lemma 4.1.7 *Let* t *be an arbitrary term such that any term in* $T_{bd}(t)$ *is AC-terminating. Then there is no infinite* $\xrightarrow{bd}$ *sequence starting from* t*.*

Proof. Assume that there exists an infinite $\xrightarrow{bd}$ sequence $t \equiv t_0 \xrightarrow{bd} t_1 \xrightarrow{bd} \cdots$. Suppose that $t_0 \equiv C[t_{01}, \ldots, t_{0n}]$ where all body terms of t_0 are presented by $t_{01}, \ldots, t_{0n}$. Applying Lemma 4.1.6 repeatedly, for any i there are $t_{i1}, \ldots, t_{in}$ such that $t_i \underset{AC}{\sim} C[t_{i1}, \ldots, t_{in}]$, each term of $T_{bd}(t_i)$ is a subterm of some t_{ij}, and there is a k_i such that $s_{i-1,k_i} \underset{R/AC}{\to} t_{ik_i}$ and $s_{i-1,j} \equiv t_{ij}$ for all $j \neq k_i$. Thus, there is a t_{0j} that is not AC-terminating. It is a contradiction to AC-termination of $t_{0j} \in T_{bd}(t)$. □

This lemma means that "any infinite AC-reduction sequence from a minimal size counterexample must include a reduction at the head position". This property corresponds to the property "any infinite reduction sequence from a minimal size counterexample must include a reduction at the root position" in TRSs. Indeed, this analogy is essential in our extension of dependency pairs to AC-TRSs.

Lemma 4.1.8 *If each* t_i *is AC-terminating and* $g(t_1, \ldots, t_n)$ *is not AC-terminating, then* $g \in DF(R)$*.*

Proof. From Lemma 4.1.7, there exists a head reduction $\xrightarrow{hd}$ in any infinite AC-reduction sequence starting from $g(t_1, \ldots, t_n)$. Therefore, g is a defined symbol. □

Lemma 4.1.9 *Let t be a term such that t is not AC-terminating and each element of $T_{bd}(t)$ is AC-terminating. Then there exist t' and s such that $t \xrightarrow{bd}{}^* \trianglerighteq_{hd} t' \underset{R}{\xrightarrow{hd}} s$, each proper subterm of t' is AC-terminating, and s is not AC-terminating.*

Proof. Let $T_i = \{t' \mid t \xrightarrow{bd}{}^i \trianglerighteq_{hd} t' \underset{R}{\xrightarrow{hd}} s$ and s is not AC-terminating, for some $s\}$, where $\xrightarrow{bd}{}^i$ denotes a $\xrightarrow{bd}$ reduction of i steps. Since t is not AC-terminating and each element of $T_{bd}(t)$ is AC-terminating, there is an n such that $T_n \neq \emptyset$ and $T_j = \emptyset$ for all $j > n$ by Lemma 4.1.7. Let t' be a minimal size element in T_n. We assume that t' has a proper subterm t'' which is not AC-terminating. Since each element of $T_{bd}(t)$ is AC-terminating, so is each element of $T_{bd}(t')$. Thus, each element of $T_{bd}(t'')$ is AC-terminating and $t' \triangleright_{hd} t''$. From Lemma 4.1.7, there is an infinite AC-reduction sequence such that $t'' \xrightarrow{bd}{}^k \trianglerighteq_{hd} \hat{t} \underset{R}{\xrightarrow{hd}} \hat{s} \underset{R/AC}{\rightarrow} \cdots$ for some $\hat{t}$ and $\hat{s}$. Thus, $\hat{t} \in T_{n+k}$. From the maximality of n, $T_{n+k} = T_n$. Thus, $t'' \trianglerighteq_{hd} \hat{t}$. From the definition of $\trianglerighteq_{hd}$, we have $|t'| > |t''| \geq |\hat{t}|$. It is a contradiction to the minimality of t'. $\square$

Theorem 4.1.10 *[43] An AC-TRS R is not AC-terminating iff there is an infinite unmarked AC-dependency chain of R.*

Proof.

($\Leftarrow$) Let $\langle u_0, v_0\rangle\langle u_1, v_1\rangle\langle u_2, v_2\rangle \cdots$ be an infinite unmarked AC-dependency chain with a substitution θ such that $v_i\theta \xrightarrow{bd}{}^* \trianglerighteq_{hd} u_{i+1}\theta$ for all i. Thus, there exist C_i and C'_i such that $v_i\theta \underset{R/AC}{\overset{*}{\rightarrow}} \underset{AC}{\sim} C'_{i+1}[u_{i+1}\theta] \underset{R}{\rightarrow} C_{i+1}[v_{i+1}\theta]$ for all i. Therefore, there is an infinite AC-reduction sequence $v_0\theta \underset{R/AC}{\overset{+}{\rightarrow}} C_1[v_1\theta] \underset{R/AC}{\overset{+}{\rightarrow}} C_1[C_2[v_2\theta]] \underset{R/AC}{\overset{+}{\rightarrow}} \cdots$.

($\Rightarrow$) Let t_0 be a minimal size counterexample, i.e., each proper subterm of t_0 is AC-terminating and t_0 is not AC-terminating. From Lemma 4.1.9, we have $t_0 \xrightarrow{bd}{}^* \trianglerighteq_{hd} C[l\theta] \underset{R}{\xrightarrow{hd}} C[r\theta]$ for some C, $l \rightarrow r$ and θ such that each proper subterm of $C[l\theta]$ is AC-terminating and $C[r\theta]$ is not AC-terminating. We show that there exists a term t_1 and an unmarked AC-dependency pair $\langle u_0, v_0\rangle$ such that $t_0 \xrightarrow{bd}{}^* \trianglerighteq_{hd} u_0\theta_0$, $v_0\theta_0 \underset{AC}{\sim} t_1$, t_1 is not AC-terminating and each proper subterm of t_1 is AC-terminating. We have two following cases:

(a) $C \equiv \square$.
Let t_1 be a minimal size counterexample in $r\theta$. Since each proper subterm of $l\theta$ is AC-terminating, $x\theta$ is AC-terminating for all $x \in Var(r) \subseteq Var(l)$. Thus, $t_1 \equiv v_0\theta$ for some non-variable subterm v_0 of r. Therefore, there is an unmarked dependency pair $\langle u_0, v_0\rangle$ and substitution θ_0 such that $u_0\theta_0 \equiv l\theta$ and $v_0\theta_0 \equiv t_1$ from the minimality of t_1 and Lemma 4.1.8.

(b) $C \not\equiv \square$.
Since each proper subterm of $C[l\theta]$ is AC-terminating, so is each proper subterm of $C[r\theta]$. From the definition of head parts, $(l)_\varepsilon = f \in \Sigma_{AC}$. Thus, there is an unmarked extended dependency pair $\langle u_0, v_0\rangle$ such that $u_0 \equiv f(l, z)$, $v_0 \equiv f(r, z)$ and $f(l, z)\theta_0 \underset{AC}{\sim} C[l\theta] \underset{R}{\rightarrow} C[r\theta] \underset{AC}{\sim} f(r, z)\theta_0$. We obtain $t_1 \equiv C[r\theta]$.

Repeating this procedure to $t_0, t_1, t_2, \ldots$, we get AC-dependency pairs $\langle u_i, v_i \rangle$ ($i = 0, 1, 2, \ldots$) such that $v_i\theta_i \underset{AC}{\sim} t_{i+1} \xrightarrow{bd}{}^* \trianglerighteq_{hd} u_{i+1}\theta_{i+1}$ for each i, which makes an infinite unmarked AC-dependency chain $\langle u_0, v_0 \rangle \langle u_1, v_1 \rangle \langle u_2, v_2 \rangle \cdots$. $\square$

4.1.2 AC-Dependency Pairs and Chains

On the dependency pair method in TRSs, the notion of marked symbols is very useful for proving termination. In this subsection, we discuss the marking for AC-dependency pairs. The root position of a term in TRSs is corresponding to the head positions of a term in AC-TRSs in the behavior of a minimal size counterexample of infinite reduction sequences. Thus, we define a term $t^\#$ for AC-TRSs, which is the result of marking with # all the head positions instead of the root position of t.

Definition 4.1.11

$$\begin{cases} x^\# & \equiv & x \\ (f(t_1, t_2))^\# & \equiv & f^\#(t_1^{\#f}, t_2^{\#f}) \quad \textit{if} \;\; f \in \Sigma_{AC} \\ (f(t_1, \ldots, t_n))^\# & \equiv & f^\#(t_1, \ldots, t_n) \quad \textit{if} \;\; f \notin \Sigma_{AC} \end{cases}$$

$$\begin{cases} x^{\#f} & \equiv & x \\ (f(t_1, \ldots, t_n))^{\#f} & \equiv & f^\#(t_1^{\#f}, \ldots, t_n^{\#f}) \\ (g(t_1, \ldots, t_m))^{\#f} & \equiv & g(t_1, \ldots, t_m) \quad \textit{if} \;\; f \neq g \end{cases}$$

We regard $f^\#$ as an AC-function symbol for all $f \in \Sigma_{AC}$, i.e., $\Sigma_{AC}^\# = \Sigma_{AC} \cup \{f^\# \mid f \in \Sigma_{AC}\}$.

For example, suppose that $t \equiv f(f(0,1), g(f(2,3)))$ and $\Sigma_{AC} = \{f\}$. Then $\Sigma_{AC}^\# = \{f, f^\#\}$ and $t^\# \equiv f^\#(f^\#(0,1), g(f(2,3)))$.

Definition 4.1.12 *Let R be an AC-TRS. A pair $\langle u^\#, v^\# \rangle$ is an AC-dependency pair of R if $\langle u, v \rangle$ is an unmarked AC-dependency pair of R. We denote by $DP_{AC}^\#(R)$ all AC-dependency pairs of R.*

Unlike the marking for TRSs, the marking for AC-TRSs is not compatible with $\xrightarrow{bd}$, i.e., it does not hold that $s \xrightarrow{bd} t \Rightarrow s^\# \to t^\#$. For example, consider $R = \{g(x) \to f(x,x), h(x) \to x\}$ with $\Sigma_{AC} = \{f\}$. Then we have the follows:

$$\begin{array}{rcl} s & \equiv & f(g(0), h(f(1,1))) \xrightarrow{bd} f(f(0,0), h(f(1,1))) \xrightarrow{bd} f(f(0,0), f(1,1)) \equiv t, \\ s^\# & \equiv & f^\#(g(0), h(f(1,1))) \xrightarrow{bd} f^\#(f(0,0), h(f(1,1))) \xrightarrow{bd} f^\#(f(0,0), f(1,1)) \not\equiv t^\#. \end{array}$$

To avoid this problem we introduce the AC-TRS $R^\#$ defined as follows:

$$R^\# = \{f^\#(f(x,y), z) \to f^\#(f^\#(x,y), z) \mid f \in \Sigma_{AC}\}.$$

We denote by $t \downarrow_\#$ the normal form of t in $\underset{R^\#/AC}{\to}$. We write $s \xrightarrow{\#} t$ if $s \underset{R/AC}{\to} t' \wedge t' \downarrow_\# \equiv t$ for some t'. Note that $\xrightarrow{\#}$ is compatible for the above example:

$$\begin{array}{rcl} s^\# & \equiv & f^\#(g(0), h(f(1,1))) \xrightarrow{\#} f^\#(f^\#(0,0), h(f(1,1))) \xrightarrow{\#} f^\#(f^\#(0,0), f^\#(1,1)) \\ & & \equiv t^\#. \end{array}$$

Indeed we can show the compatibility of $\xrightarrow{\#}$ for non-variable terms.

Lemma 4.1.13 *The following properties hold for any terms $s, t \in \mathcal{T}(\Sigma, \mathcal{V})\backslash\mathcal{V}$.*

(i) $s \xrightarrow{bd} t \iff s^{\#} \xrightarrow{\#} t^{\#}$.

(ii) $s \trianglerighteq_{hd} t \iff s^{\#} \trianglerighteq_{hd} t^{\#}$.

Proof. Trivial. □

Definition 4.1.14 *A sequence of AC-dependency pairs $\langle u_0^{\#}, v_0^{\#}\rangle\langle u_1^{\#}, v_1^{\#}\rangle\langle u_2^{\#}, v_2^{\#}\rangle \cdots$ is an AC-dependency chain if there is some substitution θ over $\mathcal{T}(\Sigma, \mathcal{V})$ such that $(v_i\theta)^{\#} \xrightarrow{\#}{}^{*} \trianglerighteq_{hd} (u_{i+1}\theta)^{\#}$ for all i. We assume that different AC-dependency pair occurrences do not own jointly variables without loss of generality.*

Theorem 4.1.15 *[43] An AC-TRS R is not AC-terminating iff there exists an infinite AC-dependency chain of R.*

Proof. From Lemma 4.1.13, it follows that $v_i\theta \xrightarrow{bd}{}^{*} \trianglerighteq_{hd} u_{i+1}\theta$ iff $(v_i\theta)^{\#} \xrightarrow{\#}{}^{*} \trianglerighteq_{hd} (u_{i+1}\theta)^{\#}$, for any $v_i, u_{i+1} \in \mathcal{T}(\Sigma, \mathcal{V})\backslash\mathcal{V}$ and θ over $\mathcal{T}(\Sigma, \mathcal{V})$. Thus, there is some infinite unmarked AC-dependency chain iff there is some infinite AC-dependency chain. From Theorem 4.1.10, our proof is completed. □

4.1.3 Another AC-dependency Pair

Marché and Urbain recently proposed another idea of AC-dependency pairs in the framework of flattening terms [52], which was done independently of our work. In this subsection, we introduce their AC-dependency pairs. In order to compare our AC-dependency pairs and their ones in the same framework, the latter method is expressed with minor modification. Firstly, we define the AC-extended AC-TRS R^{AC} of R as follows:

$$R^{AC} = R \cup \{f(l, z) \to f(r, z) \mid l \to r \in R,\ (l)_{\varepsilon} = f \in \Sigma_{AC}\}$$

where z is a fresh variable.

Consider the AC-TRS $R = \{add(x, 0) \to x,\ add(x, s(y)) \to s(add(x, y))\}$ with $\Sigma_{AC} = \{add\}$. Then

$$R^{AC} = R \cup \left\{ \begin{array}{rcl} add(add(x,0),z) & \to & add(x,z) \\ add(add(x,s(y)),z) & \to & add(s(add(x,y)),z) \end{array} \right.$$

$$DP^{\#}(R^{AC}) = \left\{ \begin{array}{l} \langle add^{\#}(x,s(y)), add^{\#}(x,y)\rangle \\ \langle add^{\#}(add^{\#}(x,0),z), add^{\#}(x,z)\rangle \\ \langle add^{\#}(add^{\#}(x,s(y)),z), add^{\#}(s(add(x,y)),z)\rangle \\ \langle add^{\#}(add^{\#}(x,s(y)),z), add^{\#}(x,y)\rangle \end{array} \right.$$

$$DP^{\#}_{AC}(R) = \left\{ \begin{array}{l} \langle add^{\#}(x,s(y)), add^{\#}(x,y)\rangle \\ \langle add^{\#}(add^{\#}(x,0),z), add^{\#}(x,z)\rangle \\ \langle add^{\#}(add^{\#}(x,s(y)),z), add^{\#}(s(add(x,y)),z)\rangle \end{array} \right.$$

The set of AC-dependency pairs introduced by us in [43] corresponds to $DP^{\#}_{AC}(R)$ and another one introduced by Marché and Urbain in [52] corresponds to $DP^{\#}(R^{AC})$.

Proposition 4.1.16 *[52] An AC-TRS R is not AC-terminating iff there exists $\langle u_0^\#, v_0^\#\rangle \langle u_1^\#, v_1^\#\rangle\langle u_2^\#, v_2^\#\rangle \cdots$ such that $\langle u_i^\#, v_i^\#\rangle \in DP^\#(R^{AC})$ and $(v_i\theta)^\# \xrightarrow{\#}{}^* \underset{AC}{\sim} (u_{i+1}\theta)^\#$ for all i. We assume that different dependency pair occurrences do not own jointly variables without loss of generality.*

Note that our AC-dependency pairs do not require pairs $\langle f(l,z), r|_p\rangle$ such that $(l)_\varepsilon = f \in \Sigma_{AC}$ and $(r)_p \in DF(R)$, which is essential in their framework. Hence, we have smaller AC-dependency pairs in number for a given AC-TRS than they have. This fact is very useful when the AC-dependency pair method is efficiently applied to automatic theorem proving. On the other hand, their method is slightly powerful in theoretical than our one, because their AC-dependency chains do not require the head subterm relation $\trianglerighteq_{hd}$. In next section, we will discuss the effects.

4.2 Proving AC-Termination by AC-Dependency Pairs

On the AC-dependency pair method, weak AC-reduction orders and weak AC-reduction pairs play an important role. In this section, using AC-dependency pairs, we introduce new and powerful methods for effectively proving AC-termination.

4.2.1 Weak AC-Reduction Order

In this subsection, we introduce the notion of weak AC-reduction order.

Definition 4.2.1 *A weak reduction order $\gtrsim$ is a weak AC-reduction order if $\gtrsim$ is AC-compatible, i.e., $s \underset{AC}{\sim} t \Rightarrow s \gtrsim t$. A weak AC-reduction order $\gtrsim$ has the AC-deletion property if $f(f(x,y),z) \gtrsim f(x,y)$ for all AC-symbols f.*

Definition 4.2.2 *A quasi-order $\gtrsim$ satisfies the AC-marked condition if $\gtrsim$ satisfies the following two conditions:*

(i) $f^\#(f(x,y),z) \gtrsim f^\#(f^\#(x,y),z)$ *for all* $f \in \Sigma_{AC}$,
(ii) $f^\#(f^\#(x,y),z) \gtrsim f^\#(f(x,y),z)$ *for all* $f \in \Sigma_{AC}$.

Theorem 4.2.3 *[43] Let R be an AC-TRS. If there exists a weak AC-reduction order $\gtrsim$ with the AC-deletion property and the AC-marked condition such that*

(i) $l \gtrsim r$ *for all* $l \to r \in R$,
(ii) $u^\# \gtrsim v^\#$ *for all* $\langle u^\#, v^\#\rangle \in DP^\#_{AC}(R)$,

then R is AC-terminating.

Proof. We assume that R is not AC-terminating. From Theorem 4.1.15, there is some infinite AC-dependency chain $\langle u_0^\#, v_0^\#\rangle\langle u_1^\#, v_1^\#\rangle\langle u_2^\#, v_2^\#\rangle \cdots$ with a substitution θ such that $(v_i\theta)^\# \xrightarrow{\#}{}^* \trianglerighteq_{hd} (u_{i+1}\theta)^\#$ for all i. It follows that $\xrightarrow{\#}{}^* \subseteq \gtrsim$ from the assumption (i), the AC-compatibility, the transitivity, the stability, the monotonicity and the AC-marked condition (i). It follows that $\trianglerighteq_{hd} \subseteq \gtrsim$ from the AC-compatibility, the AC-deletion and the stability. It follows that $(u_i\theta)^\# \gtrsim u_i^\#\theta$ from the stability, the monotonicity, and the

AC-marked condition (ii). It follows that $u_i^\# \theta \gtrsim\!\!\!\!\!/\;\; v_i^\# \theta$ from the assumption (ii) and the stability of $\gtrsim\!\!\!\!\!/\;\;$. It follows that $v_i^\# \theta \gtrsim (v_i\theta)^\#$ from the stability, the monotonicity and the AC-marked condition (i). Therefore, we get an infinite decreasing sequence $(u_0\theta)^\# \gtrsim u_0^\#\theta \gtrsim\!\!\!\!\!/\;\; v_0^\#\theta \gtrsim (v_0\theta)^\# \gtrsim (u_1\theta)^\# \gtrsim u_1^\#\theta \gtrsim\!\!\!\!\!/\;\; v_1^\#\theta \gtrsim (v_1\theta)^\# \cdots$. It is a contradiction. □

For another AC-dependency pair $DP(R^{AC})$ by Marché and Urbain, we obtain the theorem corresponds to the above one.

Proposition 4.2.4 *[52] Let R be an AC-TRS. If there exists a weak AC-reduction order $\gtrsim$ with the AC-marked condition*[1] *such that*

(i)	$l \gtrsim r$	*for all* $l \to r \in R$,
(ii)	$u^\# \gtrsim\!\!\!\!\!/\;\; v^\#$	*for all* $\langle u^\#, v^\# \rangle \in DP^\#(R^{AC})$,

then R is AC-terminating.

Though our AC-dependency pairs $DP^\#_{AC}(R)$ is smaller than theirs $DP^\#(R^{AC})$, their method is slightly powerful than our one in theoretical. In fact, suppose that our method prove AC-termination of an AC-TRS R by weak AC-reduction order $\gtrsim$ with the AC-deletion property and Theorem 4.2.3, i.e., $\gtrsim\!\!\!\!\!/\;\;$ is compatible with $DP^\#_{AC}(R)$. Then $\gtrsim\!\!\!\!\!/\;\;$ is also compatible with $DP^\#(R^{AC})$. Therefore, their method also prove the AC-termination. Conversely, our approach does not always work well whenever their method prove AC-termination, because we additionally require the AC-deletion property. However, this requirement is not strong in practice, because the AC-deletion property automatically holds for any AC-reduction orders that we have known. So with AC-deletion property, both methods are equally powerful.

4.2.2 Weak AC-Reduction Pair

In order to analyze the transformation methods for proving termination, we slightly extended the notion of weak reduction order to that of weak reduction pair [46]. In this subsection, we extend the notion to AC-TRSs.

Definition 4.2.5 *A weak reduction pair $(\gtrsim, >)$ is a weak AC-reduction pair if $\gtrsim$ is AC-compatible ($s \underset{AC}{\sim} t \Rightarrow s \gtrsim t$). A weak AC-reduction pair $(\gtrsim, >)$ has the AC-deletion property if for all $f \in \Sigma_{AC}$, $f(f(x,y),z) \gtrsim f(x,y)$ or $f(f(x,y),z) > f(x,y)$. A weak AC-reduction pair $(\gtrsim, >)$ satisfies the AC-marked condition if for all $f \in \Sigma_{AC}$, $f^\#(f(x,y),z) \gtrsim f^\#(f^\#(x,y),z)$ and $f^\#(f^\#(x,y),z) \gtrsim f^\#(f(x,y),z)$.*

Note that $(\gtrsim, \gtrsim\!\!\!\!\!/\;\;)$ is a weak AC-reduction pair for all weak AC-reduction order $\gtrsim$.

Theorem 4.2.6 *For any AC-TRS R, the following properties are equivalent.*

1. AC-TRS R is AC-terminating.

2. There exists a weak AC-reduction pair $(\gtrsim, >)$ with the AC-deletion property such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP_{AC}(R)$.

[1]This AC-marked condition is slightly modified, because their original definition can not handle collapsing rules, i.e., the rules whose right hand sides are variables.

3. There exists a weak AC-reduction pair $(\gtrsim, >)$ such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP(R^{AC})$.

4. There exists a weak AC-reduction pair $(\gtrsim, >)$ with the AC-marked condition and the AC-deletion property such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP^{\#}_{AC}(R)$.

5. There exists a weak AC-reduction pair $(\gtrsim, >)$ with the AC-marked condition such that $\gtrsim$ is compatible with R and $>$ is compatible with $DP^{\#}(R^{AC})$.

Proof. For the cases $(1 \Rightarrow 2)$ and $(1 \Rightarrow 3)$, we define $\gtrsim$ by $(\underset{R/AC}{\rightarrow} \cup \underset{AC}{\sim})^*$, and $s > t$ by $s \underset{AC}{\not\sim} t$ and $s \gtrsim C[t]$ for some C. Then it is easily shown that $(\gtrsim, >)$ is a weak AC-reduction pair satisfying the conditions. For the cases $(2 \Rightarrow 4)$ and $(3 \Rightarrow 5)$, it is easily shown by identifying $f^{\#}$ with f. For the case $(4 \Rightarrow 1)$, as similar to the proof of Theorem 4.2.3. For the case $(5 \Rightarrow 1)$, as similar to the proof of Proposition 4.2.4. □

For a given AC-terminating AC-TRS R, it is still open whether there exists a weak AC-reduction order $\gtrsim$ such that $\gtrsim$ is compatible with R and its strict part $\gtrsim\!\!\!\!/$ is compatible with unmarked AC-dependency pairs $DP_{AC}(R)$. On the other hand, the above theorem guarantees the existence of such weak AC-reduction pair. This fact indicates that weak AC-reduction pairs have an extra power as compared with weak AC-reduction orders.

4.2.3 Argument Filtering Method

In this subsection, to design weak AC-reduction orders and weak AC-reduction pairs, we extend the argument filtering method to AC-TRSs.

Definition 4.2.7 *An argument filtering function π satisfies the AC-condition if for all $f \in \Sigma^{\#}_{AC}$, $\pi(f)$ is either $[]$ or $[1,2]$.*

The above restriction is essential in AC-TRSs, because it guarantees that the image of associative and commutative axiom for $f \in \Sigma_{AC}$ are either $f = f$ or themselves. For example, for a commutative axiom $f(x,y) =_C f(y,x)$ of $f \in \Sigma_{AC}$, $\pi(f(x,y) =_C f(y,x))$ produces the following equations:

$$f(x,y) =_C f(y,x) \quad \overset{\pi}{\Longrightarrow} \quad \begin{cases} f =_C f & \text{if } \pi(f) = [] \\ x =_C y & \text{if } \pi(f) = 1 \\ f(x) =_C f(y) & \text{if } \pi(f) = [1] \\ f(x,y) =_C f(y,x) & \text{if } \pi(f) = [1,2] \end{cases}$$

Based on this observation we define AC-function symbols after argument filtering by $\Sigma^{\#}_{AC,\pi} = \{f \in \Sigma^{\#}_{AC} \mid \pi(f) = [1,2]\}$. We also write by $\underset{AC}{\sim}$ the AC-equation generated by $\Sigma^{\#}_{AC,\pi}$. Then it follows that $s \underset{AC}{\sim} t$ implies $\pi(s) \underset{AC}{\sim} \pi(t)$.

Definition 4.2.8 *We define the AC-extension $\gtrsim_{AC}$ of a strict order $>$ by $\gtrsim_{AC} = (> \cup \underset{AC}{\sim})^*$. We define $s \gtrsim^{sub}_{AC} t$ by $s \gtrsim_{AC} C[t]$ for some C, and $\gtrsim\!\!\!\!/^{sub}_{AC}$ by its strict part.*

Note that if $>$ be an AC-reduction order then the strict part $\gtrsim\!\!\!\!/_{AC}$ of its AC-extension $\gtrsim_{AC}$ is also AC-reduction order.

Lemma 4.2.9 *If a strict order $>$ is AC-compatible then $\gtrsim_{AC} = \; >^{=} \circ \underset{AC}{\sim}$.*

Proof. It is trivial. $\square$

Lemma 4.2.10 *Let $>$ be an AC-reduction order. Then $\gtrsim^{sub}_{AC}$ is well-founded.*

Proof. We assume that there exists an infinite decreasing sequence $t_0 \gtrsim^{sub}_{AC} t_1 \gtrsim^{sub}_{AC} t_2 \gtrsim^{sub}_{AC} \cdots$. Then there exist C_i $(i = 1, 2, \ldots)$ such that $t_i \gtrsim_{AC} C_{i+1}[t_{i+1}]$. Here, $\gtrsim_{AC}$ has the monotonicity, because $>$ and $\underset{AC}{\sim}$ have the monotonicity. Thus, $t_0 \gtrsim_{AC} C_1[t_1] \gtrsim_{AC} C_1[C_2[t_2]] \cdots$. From the well-foundedness of $>$ and Lemma 4.2.9, there is some k such that $C_1[\cdots C_k[t_k] \cdots] \underset{AC}{\sim} C_1[\cdots C_{k+1}[t_{k+1}] \cdots] \underset{AC}{\sim} C_1[\cdots C_{k+2}[t_{k+2}] \cdots] \underset{AC}{\sim} \cdots$. Since $\underset{AC}{\sim}$ preserves the size of terms, there is some m such that $C_m \equiv \square$. Hence, it follows that $t_{m-1} \underset{AC}{\sim} t_m$. It is a contradiction to $t_{m-1} \gtrsim^{sub}_{AC} t_m$. $\square$

Definition 4.2.11 *Let $>$ be a strict order and π an argument filtering function. We define $s \gtrsim_\pi t$ by $\pi(s) \gtrsim_{AC} \pi(t)$, and $s >_\pi t$ by $\pi(s) \gtrsim^{sub}_{AC} \pi(t)$.*

Lemma 4.2.12 *Let $>$ be an AC-reduction order. Then the following properties hold:*

- $s \gtrsim_\pi t \iff \pi(s) >^{=} \circ \underset{AC}{\sim} \pi(t)$,
- $s \gtrsim_\pi t \iff \pi(s) > \circ \underset{AC}{\sim} \pi(t)$,
- $s \gtrsim_\pi t \wedge t \gtrsim_\pi s \iff \pi(s) \underset{AC}{\sim} \pi(t)$,
- $s >_\pi t \iff \exists C.\ \pi(s) > \circ \underset{AC}{\sim} C[\pi(t)]$ or $\exists C \not\equiv \square.\ \pi(s) \underset{AC}{\sim} C[\pi(t)]$.

Proof. It suffices to show implications from left to right. The first property is a direct consequence of Lemma 4.2.9, and the second property is a direct consequence of the first property.

Let $s \gtrsim_\pi t \wedge t \gtrsim_\pi s$. From Lemma 4.2.9, $\pi(s) >^{=} \circ \underset{AC}{\sim} \pi(t) >^{=} \circ \underset{AC}{\sim} \pi(s)$. If $\pi(s) \underset{AC}{\not\sim} \pi(t)$ then $\pi(s) > \circ \underset{AC}{\sim} \pi(s)$. It is a contradiction to the well-foundedness of $>$. Hence the third property holds.

Let $s >_\pi t$. Then $\pi(s) \gtrsim_{AC} C[\pi(t)]$. From Lemma 4.2.9, $\pi(s) >^{=} \circ \underset{AC}{\sim} C[\pi(t)]$. Hence, $\pi(s) \underset{AC}{\sim} C[\pi(t)]$ or $\pi(s) > \circ \underset{AC}{\sim} C[\pi(t)]$. In the former case, if $C \equiv \square$ then $\pi(s) \underset{AC}{\sim} \pi(t)$. It is a contradiction to $\pi(s) \gtrsim^{sub}_{AC} \pi(t)$. Hence $C \not\equiv \square$. Therefore the fourth property holds. $\square$

Theorem 4.2.13 *If $>$ is an AC-reduction order and π is an argument filtering function with the AC-condition then $\gtrsim_\pi$ is a weak AC-reduction order[2]. Furthermore, if $>$ has the AC-deletion property then so is $\gtrsim_\pi$.*

Proof.

[2]For designing a weak AC-reduction order, the argument filtering method is essentially a special form of recursive program schema (RPS). Indeed, Marché and Urbain proved a similar result in a general framework of AC-RPS [52].

- ($\gtrsim_\pi$ is an AC-compatible quasi-order): It is a direct consequence of the first property of Lemma 4.2.12.

- (The monotonicity of $\gtrsim_\pi$): From Lemma 3.2.11, $s \gtrsim_\pi t \Rightarrow \pi(s) \gtrsim_{AC} \pi(t) \Rightarrow \pi(C)[\pi(s)] \gtrsim_{AC} \pi(C)[\pi(t)] \Rightarrow \pi(C[s]) \gtrsim_{AC} \pi(C[t]) \Rightarrow C[s] \gtrsim_\pi C[t]$.

- (The stability of $\gtrsim_\pi$): From Lemma 3.2.10, $s \gtrsim_\pi t \Rightarrow \pi(s) \gtrsim_{AC} \pi(t) \Rightarrow \pi(\theta)(\pi(s)) \gtrsim_{AC} \pi(\theta)(\pi(t)) \Rightarrow \pi(s\theta) \gtrsim_{AC} \pi(t\theta) \Rightarrow s\theta \gtrsim_\pi t\theta$.

- (The well-foundedness of $\gtrsim_\pi$): We assume that there exists an infinite decreasing sequence $t_0 \gtrsim_\pi t_1 \gtrsim_\pi t_2 \gtrsim_\pi \cdots$. Then $\pi(t_0) > \circ \underset{AC}{\sim} \pi(t_1) > \circ \underset{AC}{\sim} \pi(t_2) > \circ \underset{AC}{\sim} \cdots$. It is a contradiction.

- (The stability of $\gtrsim_\pi$): From Lemma 3.2.10, $s \gtrsim_\pi t \Rightarrow \pi(s) > \circ \underset{AC}{\sim} \pi(t) \Rightarrow \pi(\theta)(\pi(s)) > \circ \underset{AC}{\sim} \pi(\theta)(\pi(t)) \Rightarrow \pi(s\theta) > \circ \underset{AC}{\sim} \pi(t\theta) \Rightarrow s\theta \gtrsim_\pi t\theta$.

- (The AC-deletion property): Let $f \in \Sigma_{AC}$. If $\pi(f) = []$ then $\pi(f(f(x,y),z)) \equiv f \equiv \pi(f(x,y))$. Hence, it follows that $f(f(x,y),z) \gtrsim_\pi f(x,y)$. If $\pi(f) = [1,2]$ then $\pi(f(f(x,y),z)) \equiv f(f(x,y),z) > f(x,y) \equiv \pi(f(x,y))$. Hence, it follows that $f(f(x,y),z) \gtrsim_\pi f(x,y)$. $\square$

Theorem 4.2.14 *If $>$ is an AC-reduction order and π is an argument filtering function with the AC-condition then $(\gtrsim_\pi, >_\pi)$ is a weak AC-reduction pair with the AC-deletion property.*

Proof. In the proof of Theorem 4.2.13, we have already shown the AC-compatibility, the stability and the monotonicity of $\gtrsim_\pi$.

- (The stability of $>_\pi$): Thanks to Lemmas 4.2.12 and 3.2.10, if $\pi(s) \underset{AC}{\sim} C[\pi(t)]$ then $\pi(s\theta) \underset{AC}{\sim} C'[\pi(t\theta)]$ is trivial, where $C' \equiv \pi(\theta)(C)$. Suppose that $\pi(s) > \circ \underset{AC}{\sim} C[\pi(t)]$.

$$\begin{aligned}
s >_\pi t &\Rightarrow \pi(s) > \circ \underset{AC}{\sim} C[\pi(t)] \\
&\Rightarrow \pi(\theta)(\pi(s)) > \circ \underset{AC}{\sim} \pi(\theta)(C[\pi(t)]) \\
&\Rightarrow \pi(\theta)(\pi(s)) > \circ \underset{AC}{\sim} C'[\pi(\theta)(\pi(t))] \quad \text{where} \quad C' \equiv \pi(\theta)(C) \\
&\Rightarrow \pi(s\theta) > \circ \underset{AC}{\sim} C'[\pi(t\theta)] \\
&\Rightarrow s\theta >_\pi t\theta
\end{aligned}$$

- (The well-foundedness of $>_\pi$): Assuming that the existence of an infinite decreasing sequence $t_0 >_\pi t_1 >_\pi t_2 >_\pi \cdots$, it follows that $\pi(t_0) \gtrsim^{sub}_{AC} \pi(t_1) \gtrsim^{sub}_{AC} \pi(t_2) \gtrsim^{sub}_{AC} \cdots$. It is a contradiction to Lemma 4.2.10.

- ($\gtrsim_\pi \circ >_\pi \subseteq >_\pi$): Let $t_0 \gtrsim_\pi t_1 >_\pi t_2$. From Lemma 4.2.12, either $\pi(t_0) >^= \circ \underset{AC}{\sim} \pi(t_1) > \circ \underset{AC}{\sim} C[\pi(t_2)]$ or $\pi(t_0) >^= \circ \underset{AC}{\sim} \pi(t_1) \underset{AC}{\sim} C[\pi(t_2)] \wedge C \not\equiv \square$ holds. In the former case, $\pi(t_0) > \circ \underset{AC}{\sim} C[\pi(t_2)]$ from the AC-compatibility and the transitivity of $>$. Thus, it follows that $t_0 >_\pi t_2$ by Lemma 4.2.12. In the latter case, $\pi(t_0) >^= \circ \underset{AC}{\sim} C[\pi(t_2)]$ and $C \not\equiv \square$. Thus, it follows that $t_0 >_\pi t_2$ by Lemma 4.2.12.

- (The AC-deletion property): Suppose that $f \in \Sigma^{\#}_{AC}$. If $\pi(f) = []$ then $\pi(f(f(x,y),z)) \equiv f \equiv \pi(f(x,y))$. Hence, it follows that $f(f(x,y),z) \gtrsim_\pi f(x,y)$. If $\pi(f) = [1,2]$ then $\pi(f(f(x,y),z)) \equiv f(f(x,y),z) \equiv C[\pi(f(x,y))]$ for $C \equiv f(\Box, z)$. Hence, it follows that $f(f(x,y),z) >_\pi f(x,y)$. $\Box$

Note that a weak AC-reduction pair $(\gtrsim_\pi, >_\pi)$ automatically has the AC-deletion property. Based on $\gtrsim_\pi \subseteq >_\pi$, we get the following theorem.

Theorem 4.2.15 *Let R be an AC-TRS. If there exists an AC-reduction order $>$ and argument filtering function π with the AC-condition such that*

- $f^{\#}$ *is identified to* f *or* $\pi(f^{\#}) = []$ *for any AC-symbol* f,
- $l \gtrsim r$ *for all* $l \to r \in R$, *and*
- $u^{\#} \gtrsim_\pi v^{\#}$ *for all* $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R)$,

then R is AC-terminating.

Proof. From Theorem 4.2.14, $(\gtrsim_\pi, >_\pi)$ is a weak AC-reduction pair with AC-deletion property. Since $f^{\#}$ is identified to f or $\pi(f^{\#}) = []$ for any AC-symbols f, $(\gtrsim_\pi, >_\pi)$ trivially satisfies the AC-marked condition. Since it is trivial that $\gtrsim_\pi \subseteq >_\pi$, $u^{\#} >_\pi v^{\#}$ for all $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R)$. Therefore R is AC-terminating by Theorem 4.2.6. $\Box$

In order to show the usefulness of the argument filtering method, we prove the AC-termination of AC-TRSs to which traditional techniques cannot be applied.

Example 4.2.16 *As an AC-reduction order $>$, we use the order $>^{flat}_{rpo}$ (see Proposition 2.4.14). Each AC-termination of R_2, R_3, R_4 is proved by Theorem 4.2.15.*

- *Consider the following AC-TRS R_1 with $\Sigma^{\#}_{AC} = \{g\}$.*

$$\begin{array}{rcl} R_1 & = & \{\ f(f(x)) \ \to \ f(g(f(x), f(x))) \\ DP^{\#}_{AC}(R_1) & = & \left\{ \begin{array}{l} \langle f^{\#}(f(x)), f^{\#}(x)\rangle \\ \langle f^{\#}(f(x)), f^{\#}(g(f(x), f(x)))\rangle \end{array} \right. \end{array}$$

  Let $\pi(g) = []$ and $f \rhd g$. Then $l \gtrsim_\pi r$ for all $l \to r \in R_1$, and $u^{\#} >_\pi v^{\#}$ for all $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R_1)$. Therefore R_1 is AC-terminating.

- *Consider the following AC-TRS R_2 with $\Sigma^{\#}_{AC} = \{h\}$.*

$$R_2 = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ g(x) & \to & h(f(x), f(x)) \end{array} \right. \qquad DP^{\#}_{AC}(R_2) = \left\{ \begin{array}{l} \langle f^{\#}(f(x)), f^{\#}(g(x))\rangle \\ \langle f^{\#}(f(x)), g^{\#}(x)\rangle \\ \langle g^{\#}(x), f^{\#}(x)\rangle \end{array} \right.$$

  Let $\pi(h) = []$, $f \rhd g \rhd h$ and $f \rhd g^{\#} \rhd f^{\#}$. Then $l \gtrsim_\pi r$ for all $l \to r \in R_2$, and $u^{\#} >_\pi v^{\#}$ for all $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R_2)$. Therefore R_2 is AC-terminating.

- *Consider the following AC-TRS R_3 with $\Sigma^{\#}_{AC} = \{g, h, h^{\#}\}$.*

$$R_3 = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ b & \to & g(h(a,a), a) \\ h(x,x) & \to & x \end{array} \right. \qquad DP^{\#}_{AC}(R_3) = \left\{ \begin{array}{l} \langle f^{\#}(a), f^{\#}(b)\rangle \\ \langle f^{\#}(a), b^{\#}\rangle \\ \langle b^{\#}, h^{\#}(a,a)\rangle \\ \langle h^{\#}(h^{\#}(x,x), z), h^{\#}(x,z)\rangle \end{array} \right.$$

  Let $\pi(g) = []$, $b^{\#} \rhd a \rhd b \rhd g$ and $f^{\#} \rhd b^{\#} \rhd h^{\#}$. Then $l \gtrsim_\pi r$ for all $l \to r \in R_3$, and $u^{\#} >_\pi v^{\#}$ for all $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R_3)$. Therefore R_3 is AC-terminating.

- *Consider the following AC-TRS R_4 with $\Sigma^{\#}_{AC} = \{f, f^{\#}, h\}$.*

$$R_4 = \left\{ \begin{array}{rcl} f(a,x) & \to & f(b,x) \\ b & \to & h(a,a) \end{array} \right. \quad DP^{\#}_{AC}(R_4) = \left\{ \begin{array}{l} \langle f^{\#}(a,x), f^{\#}(b,x)\rangle \\ \langle f^{\#}(a,x), b^{\#}\rangle \\ \langle f^{\#}(f^{\#}(a,x),z), f^{\#}(f^{\#}(b,x),z)\rangle \end{array} \right.$$

Let $\pi(h) = []$, $a \rhd b \rhd h$ and $a \rhd b^{\#}$. Then $l \gtrsim_{\pi} r$ for all $l \to r \in R_4$, and $u^{\#} >_{\pi} v^{\#}$ for all $\langle u^{\#}, v^{\#}\rangle \in DP^{\#}_{AC}(R_4)$. Therefore R_4 is AC-terminating.

4.2.4 Lexicographic Argument Filtering Method

By combining several argument filtering functions, we can strengthen the power of the argument filtering method. In this subsection, we propose the lexicographic argument filtering method, which lexicographically combines argument filtering functions to compare AC-dependency pairs. The method presented here offers useful means to prove AC-termination of complicated AC-TRSs on which a single argument filtering function does not work.

In this subsection, we suppose that $f^{\#}$ is identified to f or $\pi(f^{\#}) = []$ for any AC-symbol f. This restriction guarantees the AC-marked condition of $\gtrsim_{\pi}$ and $\pi((t\theta)^{\#}) \equiv \pi(t^{\#}\theta)$ if t is not a variable. The same restriction was supposed in Theorem 4.2.15, because Theorem 4.2.6 requests the AC-marked condition.

Theorem 4.2.17 *Let R be an AC-TRS, $>$ an AC-reduction order and π an argument filtering function with the AC-condition. Suppose that $\gtrsim_{\pi}$ is compatible with R and $\gtrsim_{\pi} \cup >_{\pi}$ is compatible with $DP^{\#}_{AC}(R)$. Then, R is not AC-terminating if and only if there exists an infinite AC-dependency chain $\langle u^{\#}_0, v^{\#}_0\rangle\langle u^{\#}_0, v^{\#}_0\rangle\langle u^{\#}_0, v^{\#}_0\rangle \cdots$ with substitution θ such that $\{\pi(u^{\#}_0), \pi(v^{\#}_0), \pi(u^{\#}_1), \pi(v^{\#}_1), \ldots\}$ is AC-unifiable by $\pi(\theta)$.*

Proof. ($\Leftarrow$) It is trivial from Theorem 4.1.15. ($\Rightarrow$) From Theorem 4.1.15, there exist $\langle u^{\#}_i, v^{\#}_i\rangle \in DP^{\#}_{AC}(R)$ $(i = 0, 1, 2, \ldots)$ and a substitution θ such that $(v_i\theta)^{\#} \xrightarrow{\#}{}^{*} \trianglerighteq_{hd} (u_{i+1}\theta)^{\#}$ for all i. From the assumption and the AC-marked condition, $(u_i\theta)^{\#} \gtrsim_{\pi} (v_i\theta)^{\#}$ or $(u_i\theta)^{\#} >_{\pi} (v_i\theta)^{\#}$ for all i. From the assumption, the transitivity, the AC-deletion property and the stability, $(v_i\theta)^{\#} \gtrsim_{\pi} (u_{i+1}\theta)^{\#}$ or $(v_i\theta)^{\#} >_{\pi} (u_{i+1}\theta)^{\#}$ for all i. From the well-foundedness and Lemma 4.2.12, there is some number k such that all $\pi((u_i\theta)^{\#})$ and $\pi((v_i\theta)^{\#})$ are AC-equivalent for all $i \geq k$. The assumption $f = f^{\#}$ or $\pi(f^{\#}) = []$ for any AC-symbol f yields AC-equivalence among $\pi(u^{\#}_i\theta)$ and $\pi(v^{\#}_i\theta)$ for all $i \geq k$. From Lemma 3.2.10, all $\pi(u^{\#}_i)\pi(\theta)$ and $\pi(v^{\#}_i)\pi(\theta)$ $(i \geq k)$ are AC-equivalent. Therefore, $\{\pi(u^{\#}_k), \pi(v^{\#}_k), \pi(u^{\#}_{k+1}), \pi(v^{\#}_{k+1}), \ldots\}$ is AC-unifiable by $\pi(\theta)$. $\square$

The following theorem gives a sufficient condition under which the lexicographic argument filtering method works well. In order to simplify the discussion, we treat only two argument filtering functions, though the following discussion can be easily extended to finitely many argument filtering functions.

Theorem 4.2.18 *Let R be an AC-TRS. If there exist AC-reduction orders $>^1$ and $>^2$ and argument filtering functions π_1 and π_2 with the AC-condition such that*

- $l \gtrsim^1_{\pi_1} r$ *and* $l \gtrsim^2_{\pi_2} r$ *for all* $l \to r \in R$,
- $u^\# \gtrsim^1_{\pi_1} v^\#$ *or* $u^\# >^1_{\pi_1} v^\#$ *for all* $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R)$, *and*
- $u^\# >^2_{\pi_2} v^\#$ *for all* $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R)$ *such that* $\pi_1(u^\#)$ *and* $\pi_1(v^\#)$ *are AC-unifiable,*

then R is AC-terminating.

Proof. We assume that R is not AC-terminating. From Theorem 4.2.17, there exist $\langle u^\#_i, v^\#_i \rangle \in DP^\#_{AC}(R)$ $(i = 0, 1, 2, \ldots)$ and a substitution θ, such that $\forall i.\ (v_i\theta)^\# \xrightarrow{\#}{}^* \trianglerighteq_{hd} (u_{i+1}\theta)^\#$ and $\{\pi_1(u^\#_0), \pi_1(v^\#_0), \pi_1(u^\#_1), \pi_1(v^\#_1), \ldots\}$ is AC-unifiable. From $l \gtrsim^2_{\pi_2} r$ for all $l \to r \in R$ and the AC-deletion property of $(\gtrsim^2_{\pi_2}, >^2_{\pi_2})$, it follows that $(v_i\theta)^\# \gtrsim^2_{\pi_2} (u_{i+1}\theta)^\#$ or $(v_i\theta)^\# >^2_{\pi_2} (u_{i+1}\theta)^\#$ for any i. From $u^\# >^2_{\pi_2} v^\#$ for all $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R)$ such that $\pi_1(u^\#)$ and $\pi_1(v^\#)$ are AC-unifiable, it follows that $(u_i\theta)^\# >^2_{\pi_2} (v_i\theta)^\#$ for any i. It is a contradiction to the well-foundedness of $>^2_{\pi_2}$. □

In order to show the usefulness of the lexicographic argument filtering method, we prove the AC-termination of an AC-TRS to which not only traditional techniques but also single argument filtering function cannot be applied.

Example 4.2.19 *As an AC-reduction order $>$, we use the order $>^{flat}_{rpo}$ (see Proposition 2.4.14). Consider the following AC-TRS R_5 with $\Sigma^\#_{AC} = \{g, g^\#\}$.*

$$R_5 = \left\{ \begin{array}{rcl} f(x,0) & \to & s(0) \\ f(s(x), s(y)) & \to & s(f(x,y)) \\ g(0,x) & \to & g(f(x,x),x) \end{array} \right.$$

$$DP^\#_{AC}(R_5) = \left\{ \begin{array}{l} \langle f^\#(s(x), s(y)), f^\#(x,y) \rangle \\ \langle g^\#(0,x), g^\#(f(x,x),x) \rangle \\ \langle g^\#(0,x), f^\#(x,x) \rangle \\ \langle g^\#(g^\#(0,x),z), g^\#(g^\#(f(x,x),x),z) \rangle \end{array} \right.$$

Let $\pi_1(s) = \pi_1(f) = \pi_1(f^\#) = []$ and $0 \rhd^1 f = f^\# \rhd^1 s$. Then, $l \gtrsim^1_{\pi_1} r$ for all $l \to r \in R_5$, and $u^\# \gtrsim^1_{\pi_1} v^\#$ or $u^\# >^1_{\pi_1} v^\#$ for all $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R_5)$. Let $\pi_2(g^\#) = \pi_2(g) = []$ and $f \rhd^2 s$. Then $l \gtrsim^2_{\pi_2} r$ for all $l \to r \in R_5$ and $f^\#(s(x), s(y)) >^2_{\pi_2} f^\#(x,y)$, which is an only AC-unifiable AC-dependency pair after argument filtering by π_1. From Theorem 4.2.18, R_5 is AC-terminating.

It should be mentioned that the lexicographic argument filtering method proposed here can be similarly applied to proving not only AC-termination but also termination of TRSs. Note that traditional proof techniques by simplification orders cannot be directly applied to TRS R_5.

Corollary 4.2.20 *Let R be an AC-TRS. If for any $i = 1, 2, \ldots, n$ there exist AC-reduction orders $>^i$ and argument filtering functions π_i with the AC-condition such that*

- $l \gtrsim^i_{\pi_i} r$ *for all i and $l \to r \in R$,*
- $u^\# \gtrsim^i_{\pi_i} v^\# \vee u^\# >^i_{\pi_i} v^\#$ *for all $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R)$ such that $\pi_j(u^\#)$ and $\pi_j(v^\#)$ are AC-unifiable for any $j < i$, and*
- $u^\# >^n_{\pi_n} v^\#$ *for all $\langle u^\#, v^\# \rangle \in DP^\#_{AC}(R)$ such that $\pi_j(u^\#)$ and $\pi_j(v^\#)$ are AC-unifiable for all $j < n$,*

then R is AC-terminating.

4.2.5 AC-Multisets Extension

An argument filtering function π cannot preserve the AC-equivalent without the AC-condition, i.e., it does not hold that $s \underset{AC}{\sim} t \Rightarrow \pi(s) \underset{AC}{\sim} \pi(t)$. Hence, we can not treat an argument filtering function π if $\pi(f) = 1$ or $\pi(f) = 2$ for some $f \in \Sigma_{AC}$, because $\pi(f(x, y)) = \pi(f(y, x))$ makes $x = y$ or $y = x$ for the axiom of commutative law $f(x, y) =_C f(y, x)$. This problem can be avoided by defining $\hat{\pi}(f(x, y)) = \{x, y\}$. In this subsection, in order to treat such π, we introduce the extension $\hat{\pi}$ of argument filtering function π over multisets modulo AC by permitting $\pi(f) = 0$ as an exception for any $f \in \Sigma^\#$.

For any AC-reduction order, the equivalence parts of both $\gtrsim_{AC}$ and $\gtrsim^{sub}_{AC}$ are trivially equal to $\underset{AC}{\sim}$. Hence it is enough to treat only $\underset{AC}{\sim}$ as equation in this subsection. To simplify the discussions, we call multiset extension for quasi-orders, whose equivalence part are equal to $\underset{AC}{\sim}$, by AC-multiset extension, and we omit the subscript $_{AC}$ in the relations ($\in_{AC}$, $=_{AC}$, $\subseteq_{AC}$ and $\subset_{AC}$) and in the operations ($\cup_{AC}$, $\cap_{AC}$ and $-_{AC}$) over multisets.

Definition 4.2.21 *We define the argument filtering function $\hat{\pi}$ from terms to multisets as follows:*

$$\begin{cases} \hat{\pi}(x) & = & \{x\} & \\ \hat{\pi}(f(\vec{t_i})) & = & \hat{\pi}(t_j) & \text{if } \pi(f) = j \ (\neq 0) \\ \hat{\pi}(f(\vec{t_i})) & = & \cup_i \hat{\pi}(t_i) & \text{if } \pi(f) = 0 \\ \hat{\pi}(f(\vec{t_i})) & = & \{f(\vec{t'_{i_j}}) \mid t'_{i_j} \in \hat{\pi}(t_{i_j}) \ \ (j = 1, \ldots, m)\} & \text{if } \pi(f) = [i_1, \ldots, i_m] \end{cases}$$

We also define the substitution $\hat{\pi}(\theta)$ from terms to multisets as follows:

$$\begin{aligned} \hat{\pi}(\theta)(x) &= \hat{\pi}(\theta(x)) \\ \hat{\pi}(\theta)(f(\vec{t_i})) &= \{f(\vec{t'_i}) \mid t'_i \in \hat{\pi}(\theta)(t_i)\} \end{aligned}$$

We extend $\hat{\pi}(\theta)$ over multisets as follows:

$$\hat{\pi}(\theta)(T) = \{t \mid t' \in T, \ t \in \hat{\pi}(\theta)(t')\}$$

For example, let $\pi(f) = 0$ and $\theta(x) = f(a, b)$. Then it follows that $\hat{\pi}(f(a, b)) = \{a, b\}$ and $\hat{\pi}(\theta)(g(x, x)) = \hat{\pi}(g(f(a, b), f(a, b))) = \{g(a, a), g(a, b), g(b, a), g(b, b)\}$.

Definition 4.2.22 *Let $>$ be an AC-reduction order. We define $\gtrsim_{AC}$ by the AC-multiset extension of $\gtrsim_{AC}$, $\gtrsim^{sub}_{AC}$ by the AC-multiset extension of $\gtrsim^{sub}_{AC}$, and $\gg^{sub}_{AC}$ by the strict part of $\gtrsim^{sub}_{AC}$.*

Definition 4.2.23 *We define* $s \gtrsim_{\pi}^{mul} t$ *by* $\hat{\pi}(s) \gg_{AC} \hat{\pi}(t)$*, and* $s >_{\pi}^{mul} t$ *by* $\hat{\pi}(s) \gg_{AC}^{sub} \hat{\pi}(t)$*.*

Lemma 4.2.24 $s \underset{AC}{\sim} t \Rightarrow \hat{\pi}(s) = \hat{\pi}(t)$

Proof. We prove the claim by induction on s. The case $s \equiv x \in \mathcal{V}$ is trivial, because $t \equiv x \equiv s$ by $s \underset{AC}{\sim} t$. Suppose that $s \equiv f(s_1, \ldots, s_n)$. Because of $s \underset{AC}{\sim} t$, the root symbol of t is f. Thus we denote $t \equiv f(t_1, \ldots, t_n)$. If $s_i \underset{AC}{\sim} t_i$ for all i then $\hat{\pi}(s_i) = \hat{\pi}(t_i)$ for all i by induction hypothesis. Hence it follows that $\hat{\pi}(s) = \hat{\pi}(t)$. On the other hand, since the equivalence relation $=$ over multisets modulo $\underset{AC}{\sim}$ is an equivalence relation, it suffices to show the cases $f \in \Sigma_{AC}$ and either $s \equiv f(s_1, s_2) \wedge t \equiv f(s_2, s_1)$ or $s \equiv f(f(s_{11}, s_{12}), s_2)$ $\wedge t \equiv f(s_{11}, f(s_{12}, s_2))$. We have the following three cases.

- $\pi(f) = 0$:

$$\hat{\pi}(f(s_1, s_2)) = \hat{\pi}(s_1) \cup \hat{\pi}(s_2) = \hat{\pi}(f(s_2, s_1))$$

$$\begin{aligned}
\hat{\pi}(f(f(s_{11}, s_{12}), s_2)) &= \hat{\pi}(f(s_{11}, s_{12})) \cup \hat{\pi}(s_2) \\
&= \hat{\pi}(s_{11}) \cup \hat{\pi}(s_{12}) \cup \hat{\pi}(s_2) \\
&= \hat{\pi}(s_{11}) \cup \hat{\pi}(f(s_{12}, s_2)) \\
&= \hat{\pi}(f(s_{11}, f(s_{12}, s_2)))
\end{aligned}$$

- $\pi(f) = []$:

$$\hat{\pi}(f(s_1, s_2)) = \{f\} = \hat{\pi}(f(s_2, s_1))$$
$$\hat{\pi}(f(f(s_{11}, s_{12}), s_2)) = \{f\} = \hat{\pi}(f(s_{11}, f(s_{12}, s_2)))$$

- $\pi(f) = [1, 2]$:

$$\hat{\pi}(f(s_1, s_2)) = \{f(\hat{s}_1, \hat{s}_2) \mid \hat{s}_i \in \hat{\pi}(s_i)\} = \{f(\hat{s}_2, \hat{s}_1) \mid \hat{s}_i \in \hat{\pi}(s_i)\} = \hat{\pi}(f(s_2, s_1))$$

$$\begin{aligned}
\hat{\pi}(f(f(s_{11}, s_{12}), s_2)) &= \{f(\hat{s}_3, \hat{s}_2) \mid \hat{s}_3 \in \hat{\pi}(f(s_{11}, s_{12})), \hat{s}_2 \in \hat{\pi}(s_2)\} \\
&= \{f(f(\hat{s}_{11}, \hat{s}_{12}), \hat{s}_2) \mid \hat{s}_{11} \in \hat{\pi}(s_{11}), \hat{s}_{12} \in \hat{\pi}(s_{12}), \hat{s}_2 \in \hat{\pi}(s_2)\} \\
&= \{f(\hat{s}_{11}, f(\hat{s}_{12}, \hat{s}_2)) \mid \hat{s}_{11} \in \hat{\pi}(s_{11}), \hat{s}_{12} \in \hat{\pi}(s_{12}), \hat{s}_2 \in \hat{\pi}(s_2)\} \\
&= \{f(\hat{s}_{11}, \hat{s}_4) \mid \hat{s}_{11} \in \hat{\pi}(s_{11}), \hat{s}_4 \in \hat{\pi}(f(s_{12}, s_2))\} \\
&= \hat{\pi}(f(s_{11}, f(s_{12}, s_2)))
\end{aligned}$$

□

Lemma 4.2.25 $\hat{\pi}(\theta)(\hat{\pi}(t)) = \hat{\pi}(t\theta)$

Proof. We prove the claim by induction on t. In the case $t \equiv x \in \mathcal{V}$, $\hat{\pi}(\theta)(\hat{\pi}(x))$ $= \hat{\pi}(\theta)(\{x\}) = \hat{\pi}(x\theta)$. Suppose that $t \equiv f(t_1, \ldots, t_n)$. We have the following three cases.

- $\pi(f) = j$ $(\neq 0)$:

$$\begin{aligned}
\hat{\pi}(\theta)(\hat{\pi}(f(t_1, \ldots, t_n))) &= \hat{\pi}(\theta)(\hat{\pi}(t_j)) \\
&= \hat{\pi}(t_j\theta) \\
&= \hat{\pi}(f(t_1\theta, \ldots, t_j\theta, \ldots, t_n\theta)) \\
&= \hat{\pi}(f(t_1, \ldots, t_j, \ldots, t_n)\theta)
\end{aligned}$$

- $\pi(f) = 0$:

$$\begin{aligned}
\hat{\pi}(\theta)(\hat{\pi}(f(t_1,\ldots,t_n))) &= \hat{\pi}(\theta)(\hat{\pi}(t_1)\cup\cdots\cup\hat{\pi}(t_n)) \\
&= \hat{\pi}(\theta)(\hat{\pi}(t_1))\cup\cdots\cup\hat{\pi}(\theta)(\hat{\pi}(t_n)) \\
&= \hat{\pi}(t_1\theta)\cup\cdots\cup\hat{\pi}(t_n\theta) \\
&= \hat{\pi}(f(t_1\theta,\ldots,t_n\theta)) \\
&= \hat{\pi}(f(t_1,\ldots,t_n)\theta)
\end{aligned}$$

- $\pi(f) = [i_1,\ldots,i_m]$:

$$\begin{aligned}
\hat{\pi}(\theta)(\hat{\pi}(f(\vec{t_i}))) &= \hat{\pi}(\theta)(\{f(\vec{t'_{i_j}}) \mid t'_{i_j}\in\hat{\pi}(t_{i_j})\}) \\
&= \{t'' \mid t'\in\{f(\vec{t'_{i_j}}) \mid t'_{i_j}\in\hat{\pi}(t_{i_j})\}, t''\in\hat{\pi}(\theta)(t')\} \\
&= \{t'' \mid t'_{i_j}\in\hat{\pi}(t_{i_j}), t''\in\{f(\vec{t''_{i_j}}) \mid t''_{i_j}\in\hat{\pi}(\theta)(t'_{i_j})\}\} \\
&= \{f(\vec{t''_{i_j}}) \mid t'_{i_j}\in\hat{\pi}(t_{i_j}), t''_{i_j}\in\hat{\pi}(\theta)(t'_{i_j})\} \\
&= \{f(\vec{t''_{i_j}}) \mid t''_{i_j}\in\{t'''_{i_j} \mid t'_{i_j}\in\hat{\pi}(t_{i_j}), t'''_{i_j}\in\hat{\pi}(\theta)(t'_{i_j})\}\} \\
&= \{f(\vec{t''_{i_j}}) \mid t''_{i_j}\in\hat{\pi}(\theta)(\hat{\pi}(t_{i_j}))\} \\
&= \{f(\vec{t''_{i_j}}) \mid t''_{i_j}\in\hat{\pi}(t_{i_j}\theta)\} \\
&= \hat{\pi}(f(t_1\theta,\ldots,t_n\theta)) \\
&= \hat{\pi}(f(t_1,\ldots,t_n)\theta)
\end{aligned}$$

□

Theorem 4.2.26 *If $>$ is an AC-reduction order and π is an argument filtering function with the AC-condition then $(\gtrsim_\pi^{mul}, >_\pi^{mul})$ satisfies the conditions of the weak AC-reduction pair except for the stability.*

Proof.

- (The AC-compatibility of $\gtrsim_\pi^{mul}$):
  Let $s \underset{AC}{\sim} t$. From Lemma 4.2.24, $\hat{\pi}(s) = \hat{\pi}(t)$. Hence it follows that $s \gtrsim_\pi^{mul} t$.

- (The monotonicity of $\gtrsim_\pi^{mul}$):
  Let $s \gtrsim_\pi^{mul} t$. We prove the claim by induction on C. It suffices to show the case $C \equiv f(\ldots, t_{i-1}, \Box, t_{i+1}, \ldots)$. In the case $\pi(f) = j$ $(\neq 0)$, if $j \neq i$ then $\hat{\pi}(C[s]) \gtrsim_{AC} \hat{\pi}(C[t])$ is trivial, otherwise $\hat{\pi}(C[s]) = \hat{\pi}(s) \gtrsim_{AC} \hat{\pi}(t) = \hat{\pi}(C[t])$. In the case $\pi(f) = 0$, $\hat{\pi}(C[s]) = \bigcup_{i\neq j}\hat{\pi}(t_i)\cup\hat{\pi}(s) \gtrsim_{AC} \bigcup_{i\neq j}\hat{\pi}(t_i)\cup\hat{\pi}(t) = \hat{\pi}(C[t])$. In the case $\pi(f) = [i_1,\ldots,i_m]$, if $i \notin \pi(f)$ then it is trivial. Suppose that $i \in [i_1,\ldots,i_m] = \pi(f)$. For any $f(\hat{t}_{i_1},\ldots,\hat{t},\ldots,\hat{t}_{i_m}) \in \hat{\pi}(C[t]) - \hat{\pi}(C[s])$, it follows that $\hat{t} \in \hat{\pi}(t) - \hat{\pi}(s)$. Thus, there is some $\hat{s} \in \hat{\pi}(s) - \hat{\pi}(t)$ such that $\hat{s} \gtrsim_{AC} \hat{t}$. From the monotonicity of $\gtrsim_{AC}$, $f(\hat{t}_{i_1},\ldots,\hat{s},\ldots,\hat{t}_{i_m}) \gtrsim_{AC} f(\hat{t}_{i_1},\ldots,\hat{t},\ldots,\hat{t}_{i_m})$. Moreover, it follows that $f(\hat{t}_{i_1},\ldots,\hat{s},\ldots,\hat{t}_{i_m}) \in \hat{\pi}(C[s]) - \hat{\pi}(C[t])$. Therefore, $\hat{\pi}(C[s]) \gtrsim_{AC} \hat{\pi}(C[t])$.

- (The well-foundedness of $>_\pi^{mul}$):
  From Lemma 4.2.10, $\gtrsim_{AC}^{sub}$ is well-founded. Hence, $\gg_{AC}^{sub}$ is well-founded by Proposition 2.1.20. Therefore, $>_\pi^{mul}$ is well-founded.

- $(\gtrsim_\pi^{mul} \circ >_\pi^{mul} \subseteq >_\pi^{mul})$:
  Let $\hat{\pi}(t_0) \gtrsim_{AC} \hat{\pi}(t_1) \gg_{AC}^{sub} \hat{\pi}(t_2)$. In the case $\hat{\pi}(t_0) = \hat{\pi}(t_1)$ it is trivial that $\hat{\pi}(t_0) \gg_{AC}^{sub} \hat{\pi}(t_2)$. Suppose that $\hat{\pi}(t_0) \neq \hat{\pi}(t_1)$. From $\gtrsim_{AC} \subseteq \gtrsim_{AC}^{sub}$, it follows that $\hat{\pi}(t_0) \gg_{AC}^{sub} \hat{\pi}(t_1)$. Since $\gg_{AC}^{sub}$ is transitive by Proposition 2.1.20, it follows that $\hat{\pi}(t_0) \gg_{AC}^{sub} \hat{\pi}(t_2)$. $\square$

Unfortunately, both $\gtrsim_\pi^{mul}$ and $>_\pi^{mul}$ are not stable. For example, let $s \equiv h(x)$, $t \equiv g(x,x)$, $\pi(f) = 0$ and $\theta = \{x := f(y,z)\}$. Using the order $>_{rpo}^{flat}$ with precedence $h \rhd g$ as an AC-reduction order, we trivially obtain $s >_\pi^{mul} t$. However, since $\hat{\pi}(s\theta) = \{h(y), h(z)\}$ and $\hat{\pi}(t\theta) = \{g(y,y), g(y,z), g(z,y), g(z,z)\}$, it follows that $s\theta \not>_\pi^{mul} t\theta$. Hence, we need a suitable restriction to assure the stability of $\gtrsim_\pi^{mul}$ and $>_\pi^{mul}$.

On the other hand, in general, for any t, $\hat{t} \in \hat{\pi}(t)$ and θ, we have

$$\hat{\pi}(\theta)(\hat{t}) \supseteq \{\hat{t}\theta_1, \ldots, \hat{t}\theta_n\}$$

for $\theta_1, \ldots, \theta_n$ such that $\forall x \in Var(t).x\theta_i \in \hat{\pi}(x\theta)$ and $\forall x \notin Var(t).x\theta_i \equiv x$. Moreover, if t is linear then the equivalence holds, i.e.,

$$\hat{\pi}(\theta)(\hat{t}) = \{\hat{t}\theta_1, \ldots, \hat{t}\theta_n\}.$$

In the previous example, letting $\hat{s} \equiv h(x) \in \hat{\pi}(s)$ and $\hat{t} \equiv g(x,x) \in \hat{\pi}(t)$, it follows that

$$\hat{\pi}(\theta)(\hat{s}) = \{\hat{s}\theta_1, \hat{s}\theta_2\} \text{ and } \hat{\pi}(\theta)(\hat{t}) \supseteq \{\hat{t}\theta_1, \hat{t}\theta_2\}$$

where $\theta_1 = \{x := y\}$ and $\theta_2 = \{x := z\}$. Using this fact we prove the following lemma.

Lemma 4.2.27 *Let s and t be terms. If any $\hat{t} \in \hat{\pi}(t) - \hat{\pi}(s)$ is linear then $s \gtrsim_\pi^{mul} t \Rightarrow s\theta \gtrsim_\pi^{mul} t\theta$ and $s >_\pi^{mul} t \Rightarrow s\theta >_\pi^{mul} t\theta$.*

Proof. $(\gtrsim_\pi^{mul})$: It suffices to show that $\hat{\pi}(\theta)(\hat{s}) \gtrsim_{AC} \cup_i \hat{\pi}(\theta)(\hat{t}_i)$ for any $\hat{s} \in \hat{\pi}(s) - \hat{\pi}(t)$ and $\hat{t}_i \in \hat{\pi}(t) - \hat{\pi}(s)$ $(1 \leq i \leq n)$ such that $\hat{s} \gtrsim_{AC} \hat{t}_i$. We suppose $\{\theta_1, \ldots, \theta_m\}$ constructed by each substitution θ_i satisfying $\forall x \in Var(s).x\theta_i \in \hat{\pi}(x\theta)$ and $\forall x \notin Var(s).x\theta_i \equiv x$. Then the following inclusion holds:

$$\hat{\pi}(\theta)(\hat{s}) \supseteq \{\hat{s}\theta_j \mid 1 \leq j \leq m\}.$$

We suppose $\{\theta_1^i, \ldots, \theta_{m_i}^i\}$ constructed by each substitution θ_j^i satisfying $\forall x \in Var(t_i).x\theta_j^i \in \hat{\pi}(x\theta)$ and $\forall x \notin Var(t_i).x\theta_i \equiv x$. From the linearity of $\hat{t}_i$ the following equation holds:

$$\hat{\pi}(\theta)(\hat{t}_i) = \{\hat{t}_i\theta_j^i \mid 1 \leq j \leq m_i\}.$$

Since $\hat{s} \gtrsim_{AC} \hat{t}_i$, it follows that $Var(\hat{s}) \supseteq Var(\hat{t}_i)$. Thus, $\{\theta_1^i, \ldots, \theta_{m_i}^i\} \subseteq \{\theta_1, \ldots, \theta_m\}$. Hence, the following inclusion holds:

$$\cup_i \hat{\pi}(\theta)(\hat{t}_i) \subseteq \{\hat{t}_i\theta_j \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}.$$

From the stability of $\gtrsim_{AC}$, it follows that $\hat{s}\theta_j \gtrsim_{AC} \hat{t}_i\theta_j$ for any j. Therefore, it follows that $\hat{\pi}(\theta)(\hat{s}) \gtrsim_{AC} \{\hat{s}\theta_j \mid 1 \leq j \leq m\} \gtrsim_{AC} \{\hat{t}_i\theta_j \mid 1 \leq i \leq n,\ 1 \leq j \leq m\} \gtrsim_{AC} \cup_i \hat{\pi}(\theta)(\hat{t}_i)$.

$(>_\pi^{mul})$: It suffices to show that $\hat{\pi}(\theta)(\hat{s}) \gg_{AC}^{sub} \cup_i \hat{\pi}(\theta)(\hat{t}_i)$ for any $\hat{s} \in \hat{\pi}(s) - \hat{\pi}(t)$, $\hat{t}_i \in \hat{\pi}(t) - \hat{\pi}(s)$ such that $\hat{s} \gtrsim_{AC}^{sub} \hat{t}_i$. Thanks to the stability of $\gtrsim_{AC}^{sub}$, as similar to the proof for $\gtrsim_\pi^{mul}$, it follows that $\hat{\pi}(\theta)(\hat{s}) \gg_{AC}^{sub} \cup_i \hat{\pi}(\theta)(\hat{t}_i)$. $\square$

Lemma 4.2.28 $\trianglerighteq_{hd} \subseteq \gtrsim_{\pi}^{mul} \cup >_{\pi}^{mul}$.

Proof. Let $s \trianglerighteq_{hd} t$. From the definition, $s \underset{AC}{\sim} C[t]_p$ and $p \in \mathcal{O}_{hd}(C[t])$ for some C. In the case $C \equiv \Box$, $\hat{\pi}(s) = \hat{\pi}(t)$ by Lemma 4.2.24. Thus $s \gtrsim_{\pi}^{mul} t$. Suppose that $C \not\equiv \Box$. Since $p \in \mathcal{O}_{hd}(C[t])$, there is some term t' and AC-symbol f such that $s \underset{AC}{\sim} C[t] \underset{AC}{\sim} f(t', t)$ and $f = (t)_\varepsilon = (C)_\varepsilon$. From Lemma 4.2.24, $\hat{\pi}(s) = \hat{\pi}(f(t', t))$. If $\pi(f) = []$ then $\hat{\pi}(f(t', t)) = \{f\} = \hat{\pi}(t)$. Hence, it follows that $f(t', t) \gtrsim_{\pi}^{mul} t$. If $\pi(f) = [1, 2]$ then $\hat{\pi}(s) = \hat{\pi}(f(t', t)) = \{f(v', v) \mid v' \in \hat{\pi}(t'),\ v \in \hat{\pi}(t)\} \gg_{AC}^{sub} \{v \mid v \in \hat{\pi}(t)\} = \hat{\pi}(t)$. Hence, it follows that $s >_{\pi}^{mul} t$. If $\pi(f) = 0$ then $\hat{\pi}(s) = \hat{\pi}(f(t', t)) \supseteq \hat{\pi}(t') \cup \hat{\pi}(t) \ \gg_{AC} \hat{\pi}(t)$. Hence, it follows that $s \gtrsim_{\pi}^{mul} t$. $\Box$

Theorem 4.2.29 *Let R be an AC-TRS. If there exists an AC-reduction order $>$ and an argument filtering function π with the AC-condition such that*

- $f^\#$ *is identified to* f *or* $\pi(f^\#) = []$ *for all AC-symbols* f,
- $\hat{r} \in \hat{\pi}(r) - \hat{\pi}(l)$ *is linear* *for all* $l \to r \in R$,
- $l \gtrsim_{\pi}^{mul} r$ *for all* $l \to r \in R$, *and*
- $u^\# >_{\pi}^{mul} v^\#$ *for all* $\langle u^\#, v^\# \rangle \in DP_{AC}^\#(R)$,

then R is AC-terminating.

Proof. As similar to the proof of Theorem 4.2.15, using Theorem 4.2.26, Lemmas 4.2.27 and 4.2.28. $\Box$

Note that as similar to the proof of Theorem 4.2.26, it can be proved that for any given AC-reduction order $>$, $\gtrsim_{\pi}^{mul}$ is a weak AC-reduction order except for the stability. Under the condition of Lemma 4.2.27, the strict part $\gtrsim_{\pi}^{mul}$ of $\gtrsim_{\pi}^{mul}$ is stable.

In order to show the usefulness of AC-multiset extension, we prove the AC-termination of an AC-TRS to which not only traditional techniques but also single argument filtering function and lexicographic argument filtering method cannot be applied.

Example 4.2.30 *As an AC-reduction order $>$, we use the order $>_{rpo}^{flat}$ (see Proposition 2.4.14). Consider the following AC-TRS R_6 with $\Sigma_{AC}^\# = \{f\}$.*

$$R_6 = \left\{ \begin{array}{rcl} g(0, f(x, x)) & \to & x \\ g(x, s(y)) & \to & g(f(x, y), 0) \\ g(s(x), y) & \to & g(f(x, y), 0) \\ g(f(x, y), 0) & \to & f(g(x, 0), g(y, 0)) \end{array} \right.$$

$$DP_{AC}^\#(R_6) = \left\{ \begin{array}{l} \langle g^\#(x, s(y)), g^\#(f(x, y), 0) \rangle \\ \langle g^\#(s(x), y), g^\#(f(x, y), 0) \rangle \\ \langle g^\#(f(x, y), 0), g^\#(x, 0) \rangle \\ \langle g^\#(f(x, y), 0), g^\#(y, 0) \rangle \end{array} \right.$$

Let $\pi(f) = 0$ and $s \rhd 0$. Then $l \gtrsim_{\pi}^{mul} r$ for all $l \to r \in R_6$, and $u^\# >_{\pi}^{mul} v^\#$ for all $\langle u^\#, v^\# \rangle \in DP_{AC}^\#(R_6)$. From Theorem 4.2.29, R_6 is AC-terminating.

4.2.6 Polynomial Interpretation

Theorem 4.2.31 *[43] Let $A \subseteq \mathcal{N}\backslash\{0\}$. We define the polynomial quasi-order $\gtrsim_A$ as follows:*

$$s \gtrsim_A t \iff \forall\sigma([\![s]\!]_\sigma > [\![t]\!]_\sigma) \text{ or } \forall\sigma([\![s]\!]_\sigma = [\![t]\!]_\sigma)$$

If all AC-function symbols satisfy the condition of Proposition 2.4.5 then the polynomial quasi-order $\gtrsim_A$ is a weak AC-reduction order.

Proof. As similar to Proposition 2.4.5, we can prove that $\gtrsim_A$ is AC-compatible. From Theorem 3.2.14, $\gtrsim_A$ is a weak AC-reduction order. □

Proposition 4.2.32 *[52] Let $A \subseteq \mathcal{N}$. We define the pair $(\gtrsim_A, >_A)$ as follows:*

$$s \gtrsim_A t \overset{\text{def}}{\iff} \forall\sigma([\![s]\!]_\sigma \geq [\![t]\!]_\sigma), \qquad s >_A t \overset{\text{def}}{\iff} \forall\sigma([\![s]\!]_\sigma > [\![t]\!]_\sigma)$$

If all AC-function symbols satisfy the condition of Proposition 2.4.5 then the pair $(\gtrsim_A, >_A)$ is a weak AC-reduction pair.

4.3 AC-Dependency Graph

The notion of dependency graphs introduced by Arts and Giesl is very useful to prove termination of TRSs [1, 2]. In this section, we extend the notion to AC-TRSs.

Definition 4.3.1 *An AC-dependency graph of R is a directed graph of which the nodes are AC-dependency pairs, and there is an arc from $\langle u^\#, v^\#\rangle$ to $\langle u'^\#, v'^\#\rangle$ if $\langle u^\#, v^\#\rangle\langle u'^\#, v'^\#\rangle$ is an AC-dependency chain.*

Theorem 4.3.2 *[43] Let R be an AC-TRS. If there exists a weak AC-reduction order $\gtrsim$ such that*

- $l \gtrsim r$ *for all $l \to r \in R$,*
- $u^\# \gtrsim v^\#$ *for all $\langle u^\#, v^\#\rangle$ on a cycle in the AC-dependency graph of R, and*
- $u^\# \gtrsim v^\#$ *for at least one $\langle u^\#, v^\#\rangle$ on each cycle in the AC-dependency graph of R,*

then R is AC-terminating.

Proof. As similar to the proof of Theorem 3.4.3. □

In general, AC-dependency graphs are not computable, because it is undecidable whether there is some substitution θ such that $(v\theta)^\# \xrightarrow{\#}{}^* \trianglerighteq_{hd} (u'\theta)^\#$ for two AC-dependency pairs $\langle u^\#, v^\#\rangle$ and $\langle u'^\#, v'^\#\rangle$. Since dependency graphs are also not computable in TRSs, algorithms for generating approximated dependency graphs was introduced. We also propose another algorithm for generating an approximated AC-dependency graph, using the techniques of Ω-reduction and Ω_V-reduction, which are introduced to analyze decidable call-by-need computations in TRSs [28, 55, 59].

Definition 4.3.3 *Let Ω be a special constant symbol. A term t is an Ω-term if $t \in \mathcal{T}(\Sigma \cup \{\Omega\}, \emptyset)$. The prefix order $\underset{\Omega}{\geq}$ over Ω-terms is defined as follows:*

- $t \underset{\Omega}{\geq} \Omega$ *for all* t,
- $f(s_1, \ldots, s_n) \underset{\Omega}{\geq} f(t_1, \ldots, t_n)$ *if* $s_i \underset{\Omega}{\geq} t_i$ $(1 \leq i \leq n)$.

We denote by t_Ω the Ω-term obtained from t by replacing all variables in t by Ω. Notice that $t_\Omega \underset{\Omega}{\leq} t\theta_\Omega$ for any term t and for any substitution θ.

In this section, $\mathcal{T}(\Sigma, \mathcal{V})$ and $\mathcal{T}(\Sigma \cup \{\Omega\}, \emptyset)$ are abbreviated to $\mathcal{T}$ and $\mathcal{T}_\Omega$, respectively.

Definition 4.3.4 *Two Ω-terms t_1 and t_2 are compatible, written by $t_1 \uparrow t_2$, if there is some Ω-term s such that $s \underset{\Omega}{\geq} t_1$ and $s \underset{\Omega}{\geq} t_2$. A body AC-$\Omega$-reduction relation over Ω-terms, written by $\underset{\Omega}{\overset{bd}{\rightarrow}}$, is defined as follows:*

$$s \underset{\Omega}{\overset{bd}{\rightarrow}} t \quad \overset{\text{def}}{\Longleftrightarrow} \quad s \underset{AC}{\sim} C[s']_p \wedge s' \uparrow l_\Omega \wedge s' \not\equiv \Omega \wedge t \equiv C[\Omega]_p \wedge p \notin \mathcal{O}_{hd}(C[s']_p)$$
$$\textit{for some } \; l \rightarrow r \in R, \; s', \; C[\;]_p.$$

Lemma 4.3.5 *The body AC-Ω-reduction $\underset{\Omega}{\overset{bd}{\rightarrow}}$ is terminating.*

Proof. It is trivial. □

$NF_\Omega(t)$ denotes the set of all normal forms of t with respect to $\underset{\Omega}{\overset{bd}{\rightarrow}}$. Note that the previous lemma guarantees the computability of $NF_\Omega(t)$.

Definition 4.3.6 *A body AC-Ω_V-reduction from an Ω-term s to an Ω-term t, denoted by $s \underset{\Omega_V}{\overset{bd}{\rightarrow}} t$, is defined as follows:*

$$s \underset{\Omega_V}{\overset{bd}{\rightarrow}} t \quad \overset{\text{def}}{\Longleftrightarrow} \quad s \underset{AC}{\sim} C[s']_p \wedge s' \uparrow l_\Omega \wedge s' \not\equiv \Omega \wedge t \equiv C[r_\Omega]_p \wedge p \notin \mathcal{O}_{hd}(C[s']_p)$$
$$\textit{for some } \; l \rightarrow r \in R, \; C[\;]_p, \; s'.$$

$\underset{\Omega_V}{\overset{bd}{\rightarrow}}{}^n$ *denotes a* $\underset{\Omega_V}{\overset{bd}{\rightarrow}}$ *reduction of n steps.*

Lemma 4.3.7 *For any $s, t \in \mathcal{T}_\Omega$, if $s \underset{\Omega_V}{\overset{bd}{\rightarrow}}{}^* t$ then $t' \underset{\Omega}{\leq} t$ for some $t' \in NF_\Omega(s)$.*

Proof. It is trivial. □

Lemma 4.3.8 *The following properties hold for any $s, t \in \mathcal{T}$ and $s' \in \mathcal{T}_\Omega$ such that $s' \underset{\Omega}{\leq} s_\Omega$.*

(a) $s \underset{AC}{\sim} t \Rightarrow \exists t' \in \mathcal{T}_\Omega.\; s' \underset{AC}{\sim} t' \underset{\Omega}{\leq} t_\Omega$

(b) $s \overset{bd}{\rightarrow}{}^* t \Rightarrow \exists t' \in \mathcal{T}_\Omega.\; s' \underset{\Omega_V}{\overset{bd}{\rightarrow}}{}^* t' \underset{\Omega}{\leq} t_\Omega$

Proof. It is routine. $\square$

The predicate $isConnect(v, u')$ is defined as $isConnect(v, u') \iff \exists\theta.\ v\theta \xrightarrow{bd}{}^{*} \trianglerighteq_{hd} u'\theta$. Note that two AC-dependency pairs $\langle u, v\rangle$ and $\langle u', v'\rangle$ is connectable if $isConnect(v, u')$ holds. For a given approximation level n ($n \geq 0$), the above two lemmas offer a decidable approximation $isConnect_n(v, u')$ of $isConnect(v, u')$ as follows:

- $(v)_\varepsilon \notin \Sigma_{AC}$.

$$\begin{array}{rcl} isConnect_n(v, u') & = & \begin{cases} True & \text{if } Connect_n(v, u') \neq \emptyset \\ False & \text{if } Connect_n(v, u') = \emptyset \end{cases} \\ Connect_n(v, u') & = & \{t \mid t \underset{AC}{\sim} t' \in Reach_n(v_\Omega), t \uparrow u'_\Omega\} \\ Reach_n(t) & = & \{t' \mid t \underset{\Omega_V}{\xrightarrow{bd}}{}^{m} t',\ m < n\} \cup \{t' \mid t \underset{\Omega_V}{\xrightarrow{bd}}{}^{n} t'', t' \in NF_\Omega(t'')\} \end{array}$$

- $(v)_\varepsilon \in \Sigma_{AC}$.

$$isConnect_n(v, u') = \begin{cases} True & \text{if } (v)_\varepsilon = (u')_\varepsilon \\ False & \text{if } (v)_\varepsilon \neq (u')_\varepsilon \end{cases}$$

Note that the predicate $isConnect_n$ is decidable for every approximation level n.

Definition 4.3.9 *The n-approximated AC-dependency graph of R is a directed graph of which the nodes are AC-dependency pairs, and there is an arc from $\langle u^\#, v^\#\rangle$ to $\langle u'^\#, v'^\#\rangle$ if $isConnect_n(v, u')$ holds.*

Lemma 4.3.10 *Let R be an arbitrary AC-TRS and n an arbitrary approximation level. An AC-dependency graph of R is a subgraph of the n-approximated AC-dependency graph of R.*

Proof. For each arc $(\langle u^\#, v^\#\rangle, \langle u'^\#, v'^\#\rangle)$ of an AC-dependency graph, it is enough to show $isConnect_n(v, u') = True$. If $(v)_\varepsilon \in \Sigma_{AC}$ then it is trivial. Let $(v)_\varepsilon \notin \Sigma_{AC}$ and $v\theta \xrightarrow{bd}{}^{*} t \underset{AC}{\sim} u'\theta$ for some t and θ. From Lemma 4.3.8(b), $v_\Omega \underset{\Omega_V}{\xrightarrow{bd}}{}^{m} t'$ for some m and $t' \in \mathcal{T}_\Omega$ such that $t' \underset{\Omega}{\leq} t_\Omega$. We distinguish the following two cases:

- $m < n$:
  By the assumption, $t' \in Reach_n(v_\Omega)$. From Lemma 4.3.8(a), $t' \underset{AC}{\sim} u''$ for some $u'' \in \mathcal{T}_\Omega$ such that $u'' \underset{\Omega}{\leq} u'\theta_\Omega$. Thus, $u'' \uparrow u'_\Omega$. Therefore, $isConnect_n(v, u') = True$.

- $n \leq m$:
  Let $v_\Omega \underset{\Omega_V}{\xrightarrow{bd}}{}^{n} t'' \underset{\Omega_V}{\xrightarrow{bd}}{}^{*} t'$. From Lemma 4.3.7, $t''' \underset{\Omega}{\leq} t'$ for some $t''' \in NF_\Omega(t'')$. Thus, $t''' \in Reach_n(v_\Omega)$. From Lemma 4.3.8(a), $t''' \underset{AC}{\sim} u''$ for some $u'' \in \mathcal{T}_\Omega$ such that $u'' \underset{\Omega}{\leq} u'\theta_\Omega$. Thus, $u'' \uparrow u'_\Omega$. Therefore, $isConnect_n(v, u') = True$. $\square$

Theorem 4.3.11 *[43] Let R be an AC-TRS and n an approximation level. If there exists a weak AC-reduction order $\gtrsim$ such that*

- $l \gtrsim r$ *for all* $l \to r \in R$,
- $u \gtrsim v$ *for all* $\langle u^\#, v^\# \rangle$ *on a cycle in the n-approximated AC-dependency graph, and*
- $u^\# \gtrsim v^\#$ *for at least one* $\langle u^\#, v^\# \rangle$ *on each cycle in the n-approximated AC-dependency graph,*

then R is AC-terminating.

Proof. It is a direct consequence of Theorem 4.3.2 and Lemma 4.3.10. □

Finally, in order to show the usefulness of the approximated AC-dependency graph, we prove the AC-termination of AC-TRS to which traditional techniques cannot be applied.

Example 4.3.12 *Let $\Sigma_{AC} = \{N\}$ and R be the following AC-TRS:*

$$\begin{cases} max(L(x)) & \to & x \\ max(N(L(0), L(y))) & \to & y \\ max(N(L(s(x)), L(s(y)))) & \to & s(max(N(L(x), L(y)))) \\ max(N(L(x), N(y, z))) & \to & max(N(L(x), L(max(N(y, z))))) \end{cases}$$

The data structure $N(L(0), N(L(0), L(s(0))))$ for binary tree naturally represents the multiset $\{0, 0, s(0)\}$ by interpreting N as an AC-function symbol. The normal form of term $max(t)$ corresponds with maximal number in non-empty multiset t. For this AC-TRS R, the 1-approximated AC-dependency graph is displayed as follows (Figure 4.1):

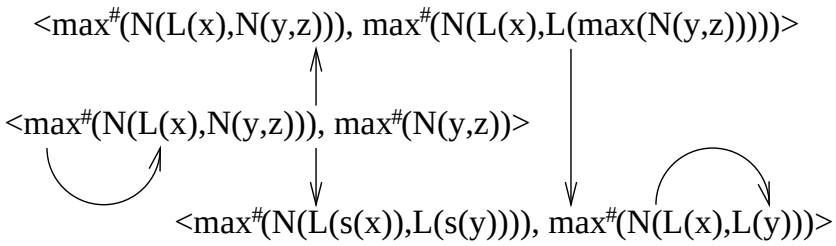

Figure 4.1: AC-dependency graph

Let $A = \mathcal{N} \backslash \{0\}$. We associate the polynomial $0_A = 1$, $s_A = X + 1$, $L_A = X$, $N_A = X + Y$ and $max_A = max^\#_A = X$. From Theorem 4.2.31, this interpretation offers a weak AC-reduction order $\gtrsim$. Moreover, it satisfies the AC-marked condition. It is trivial that $l \gtrsim r$ for all $l \to r \in R$, and for all $\langle u^\#, v^\# \rangle$ on each cycle in the 1-approximated AC-dependency graph (Figure 4.1), we have $u^\# > v^\#$ as follows:

$$\begin{aligned} max^\#(N(L(s(x)), L(s(y)))) &> max^\#(N(L(x), L(y))) \\ max^\#(N(L(x), N(y, z))) &> max^\#(N(y, z)) \end{aligned}$$

Therefore, R is AC-terminating by theorem 4.3.11.

This example displays the usefulness of approximated AC-dependency graphs, because it is difficult to give a weak AC-reduction order directly whose strict part is compatible to the AC-dependency pair $\langle max^{\#}(N(L(x), N(y, z))), max^{\#}(N(L(x), L(max(N(y, z)))))\rangle$. Even for TRS R, it is difficult to give a weak reduction order directly whose strict part is compatible to the pair.

Chapter 5

Argument Filtering Transformation

Elimination transformations have a lively studied in the 1990's. Elimination transformations try to transform a given TRS into a TRS whose termination is easier to prove than the original one. The dummy elimination [20], the distribution elimination [53, 67], the general dummy elimination [21] and the improved general dummy elimination [57] are examples of elimination transformations. Moreover, the dummy elimination and the distribution elimination extend to AC-TRSs in [22] and [58], respectively.

In this chapter, we first study the relation between various elimination transformations and the argument filtering method based on AC-dependency pairs. The key of our result is the observation that the argument filtering method combining with the AC-dependency pair technique is essential in all elimination transformations. Indeed, we present remarkable simple proofs for the soundness of all elimination transformations based on this observation, though the original proofs treated as rather different methods respectively. This observation also leads us to a new powerful elimination transformations, called the argument filtering transformation, which is not only more powerful than all the other elimination transformations but also especially useful to make clear the essential relation hidden behind these methods.

5.1 Soundness Condition for Transformation

In this section, using AC-dependency pairs and the argument filtering method, we show a theorem, which makes a general and essential property clear for transformations of AC-TRSs to be sound with respect to AC-termination.

Definition 5.1.1 *We define the including relation $\sqsubseteq$ as follows:*

$$R_1 \sqsubseteq R_2 \overset{\text{def}}{\iff} \forall l \to r \in R_1. \exists C.\ l \to C[r] \in R_2$$

Theorem 5.1.2 *Let R be an AC-TRS, R' an AC-terminating AC-TRS and π an argument filtering function with the AC-condition. If $\pi(R) \subseteq R'$ and $\pi(DP(R)) \sqsubseteq R'$ then R is AC-terminating.*

Proof. Assume that R is not AC-terminating. We define $>$ as $\underset{R'/AC}{\overset{+}{\to}}$. The AC-termination of R' ensure that $>$ is an AC-reduction order. From the assumption, $\gtrsim_\pi$ is compatible with R, $>_\pi$ is compatible with $DP(R)$, and $\gtrsim_\pi \cup >_\pi$ is compatible with $DP_{AC}(R) \backslash DP(R)$.

Thanks to Theorem 4.2.17, there exists an AC-symbol f with $\pi(f) = []$ and an infinite unmarked AC-dependency chain

$$\langle f(l_0, z_0), f(r_0, z_0)\rangle\langle f(l_1, z_1), f(r_1, z_1)\rangle\langle f(l_2, z_2), f(r_2, z_2)\rangle \cdots$$

such that $f(r_i\theta, z_i\theta) \underset{bd}{\overset{*}{\to}} \trianglerighteq_{hd} f(l_{i+1}\theta, z_{i+1}\theta)$ $(i = 0, 1, 2, \ldots)$ for some AC-terminating substitution θ. If $r_i\theta \underset{R/AC}{\overset{*}{\to}} t$ with $(t)_\varepsilon = f$, then it is a contradiction with $f \equiv \pi(l_i) \underset{R'/AC}{\to} \pi(r_i) \underset{R'/AC}{\overset{*}{\to}} \pi(t) \equiv f$ and AC-termination of R'. Thus, $r_i\theta$ can not be reduced to a term with the root symbol f.

For any t, we define $B_f(t)$ as follows:

$$B_f(t) = \begin{cases} \{t\} & \text{if } (t)_\varepsilon \neq f \\ \{t_1, \ldots, t_n\} & \text{if } \bar{t} \equiv f(\overline{t_1}, \ldots, \overline{t_n}) \end{cases}$$

where $\bar{t}$ is the flattening term of t, and $\overline{t_i}$ are flattening terms of t_i.

Suppose that $f(r_i, z_i)\theta \overset{*}{\to} f(t_i, t'_i) \trianglerighteq_{hd} f(l_{i+1}, z_{i+1})\theta \to f(r_{i+1}, z_{i+1})\theta$ such that $r_i\theta \overset{*}{\to} t_i$ and $z_i\theta \overset{*}{\to} t'_i$. Then the following properties hold:

(1) $|B_f(r_i\theta)| = |B_f(t_i)| = 1$

(2) $B_f(f(t_i, t'_i)) \supseteq B_f(f(l_{i+1}, z_{i+1})\theta)$

(3) $|B_f(f(l_i, z_i)\theta)| > |B_f(f(r_i, z_i)\theta)|$

Properties (1) and (2) are trivial. Property (3) follows from $|B_f(l_i\theta)| \geq 2$ and (1).

Let $n_i = |B_f(t'_i)| - |B_f(z_i\theta)|$. Then, it is obvious that $n_i \geq 0$ and $|B_f(f(r_i, z_i)\theta)| + n_i > |B_f(f(r_{i+1}, z_{i+1})\theta)|$. Since $z_0\theta$ is AC-terminating, $\Sigma\{|B_f(t)| \mid z_0\theta \overset{*}{\to} t\}$ is finite. Because $\Sigma_{i=0}^{\infty} n_i < \Sigma\{|B_f(t)| \mid z_0\theta \overset{*}{\to} t\}$, $\Sigma_{i=0}^{\infty} n_i$ is also finite. Hence, there exists a integer k such that $n_i = 0$ for all $i \geq k$. Therefore it follows that $|B_f(f(r_k, z_k)\theta)| > |B_f(f(r_{k+1}, z_{k+1})\theta)| > |B_f(f(r_{k+2}, z_{k+2})\theta)| > \cdots$. It is a contradiction. □

Taking R as a given AC-TRS and R' as a transformed AC-TRS in an elimination transformation, the above simple theorem can uniformly explain why elimination transformations work well. This fact is very interesting because in the original literatures the soundness of these elimination transformations were proved by rather different methods. In the following sections, we will explain how Theorem 5.1.2 simplifies the requirement conditions in elimination transformations into acceptable one.

Note that in the above theorem we use dependency pairs $DP(R)$ instead of AC-dependency pairs $DP_{AC}(R)$. Though the fact rises the difficulty of the proof, it is very effective when we analyze elimination transformations.

Theorem 5.1.3 *Let R be an AC-TRS, R' a simply AC-terminating AC-TRS and π an argument filtering function with the AC-condition. If $\pi(R \cup DP(R)) \sqsubseteq R'$ then R is AC-terminating.*

Proof. As similar to Theorem 5.1.2 by defining $>$ as $\underset{R' \cup Emb/AC}{\overset{+}{\longrightarrow}}$. □

5.2 Argument Filtering Transformation

In this section, we design a new elimination transformation, called the argument filtering transformation. This transformation is designed based on Theorem 5.1.2, which is the essence for elimination transformations.

Definition 5.2.1 *[46](Argument Filtering Transformation) Let π be an argument filtering function. The argument filtering transformation (AFT_π) is defined as follows:*

- $$\left\{\begin{array}{rcll} dec_\pi(x) & = & \emptyset & \\ dec_\pi(f(t_1,\ldots,t_n)) & = & \bigcup_{i\neq\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & \textit{if } \ \pi(f)=i \\ dec_\pi(f(t_1,\ldots,t_n)) & = & \bigcup_{i\notin\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & \textit{otherwise} \end{array}\right.$$

- $pick_\pi(T) = \{t \in T \mid \overline{\pi}(t) \textit{ includes some defined symbols of } R\}$
  $$\textit{where} \quad \overline{\pi}(f) = \left\{\begin{array}{ll} [i] & \textit{if } \ \pi(f)=i \\ \pi(f) & \textit{otherwise} \end{array}\right.$$

- $AFT_\pi(R) = \pi(R) \cup \{\pi(l) \to \pi(r') \mid l \to r \in R,\ r' \in pick_\pi(dec_\pi(r))\}$

Example 5.2.2 *Let*

$$R = \{f(x, f(x,x)) \to f(e(e'(0,1,2),3), e''(f(4,5),6)),\ 4 \to 1,\ 5 \to 1\}.$$

Here, $DF(R) = \{f, 4, 5\}$. Let $r \equiv f(e(e'(0,1,2),3), e''(f(4,5),6))$, $\pi(e) = []$, $\pi(e') = [1,3]$ and $\pi(e'') = 2$. Then, we obtain $AFT_\pi(R)$ as follows (Figure 5.1):

$$\begin{array}{rcl} \pi(r) & = & f(e,6) \\ dec_\pi(r) & = & \{e'(0,1,2),\ 1,\ 3,\ f(4,5)\} \\ pick_\pi(dec_\pi(r)) & = & \{f(4,5)\} \\ \pi(R) & = & \{f(x,f(x,x)) \to f(e,6),\ 4 \to 1,\ 5 \to 1\} \\ AFT_\pi(R) & = & \pi(R) \cup \{f(x,f(x,x)) \to f(4,5)\} \end{array}$$

The termination of $AFT_\pi(R)$ is easily proved by the recursive path order. Thus, R is terminating, if the argument filtering transformation is sound. The soundness is showed in this section.

Since the argument filtering transformation is designed based on Theorem 5.1.2, we must keep information of dependency pairs, i.e., $\pi(DP(R)) \sqsubseteq AFT_\pi(R)$. For this motivation, $\overline{\pi}$ is useful. In fact, using $\overline{\pi}$, we can easily check whether a given term is necessary to hold $\pi(DP(R)) \sqsubseteq AFT_\pi(R)$.

Lemma 5.2.3 *Let C be a context and t a term. Then, there exists a context D such that $D[\pi(t)] \in \pi(dec_\pi(C[t]))$ or $D[\pi(t)] \equiv \pi(C[t])$.*

Proof. We prove the claim by induction on the structure of C. In the case $C \equiv \Box$, it is trivial. Suppose that $C \equiv f(t_1,\ldots,t_{i-1},C',t_{i+1},\ldots,t_n)$. From induction hypothesis, there exists a context D' such that $D'[\pi(t)] \in \pi(dec_\pi(C'[t]))$ or $D'[\pi(t)] \equiv \pi(C'[t])$. In the former case, it follows that $D'[\pi(t)] \in \pi(dec_\pi(C'[t])) \subseteq \pi(dec_\pi(C[t]))$. In the latter case, if $i = \pi(f)$ or $i \in \pi(f)$ then trivial. Otherwise, $D'[\pi(t)] \equiv \pi(C'[t]) \in \pi(dec_\pi(C[t]))$ from the definition of dec_π. $\Box$

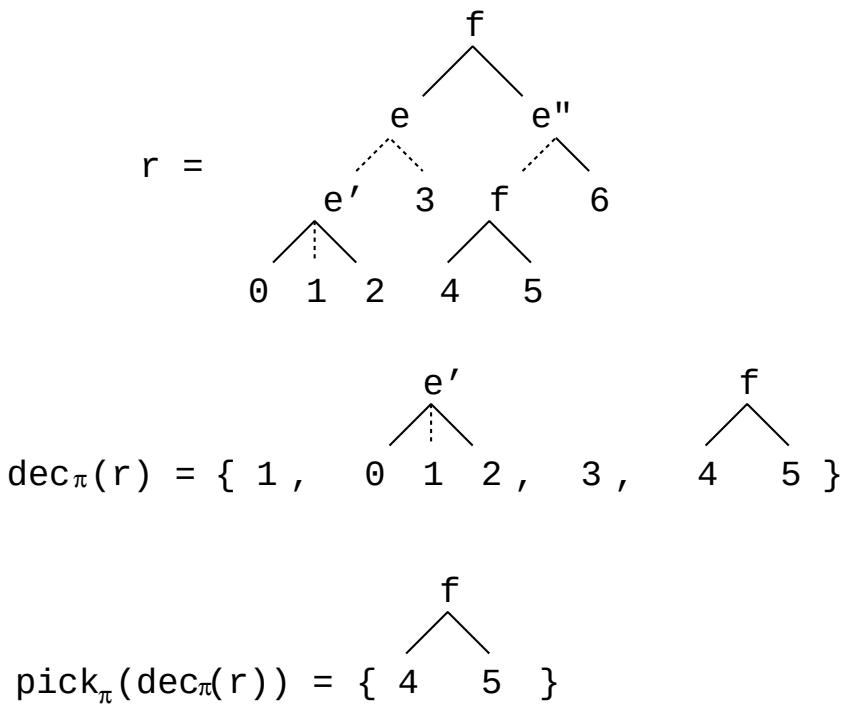

Figure 5.1: Argument Filtering Transformation

Theorem 5.2.4 *If $AFT_\pi(R)$ is AC-terminating and π satisfies the AC-condition then R is AC-terminating.*

Proof. From the definition, $\pi(R) \subseteq AFT_\pi(R)$. Let $\langle u, v\rangle \in DP(R)$. From the definition of DP, there exists a rule $u \to C[v] \in R$. From Lemma 5.2.3, there exists a context D such that $D[\pi(v)] \in \pi(dec_\pi(C[v]))$ or $D[\pi(v)] \equiv \pi(C[v])$. In the former case, from the definition of DP and $\overline{\pi}$, $(\overline{\pi}(v))_\varepsilon$ is a defined symbol. Thus, $D[\pi(v)] \in \pi(pick_\pi(dec_\pi(C[v])))$. Therefore, it follows that $\pi(u) \to D[\pi(v)] \in AFT_\pi(R)$. In the latter case, it follows that $\pi(u) \to D[\pi(v)] \in \pi(R) \subseteq AFT_\pi(R)$. From Theorem 5.1.2, R is AC-terminating. $\square$

The two corollaries follow from the above theorem.

Corollary 5.2.5 *If $AFT_\pi(R)$ is terminating and $\pi(f) = []$ for any AC-symbols f then R is AC-terminating.*

Corollary 5.2.6 *[46] If $AFT_\pi(R)$ is terminating then R is terminating.*

From the proof of the above theorem, it is obvious that the second argument $\{\pi(l) \to \pi(r') \mid l \to r \in R,\ r' \in pick_\pi(dec_\pi(r))\}$ of the definition of the argument filtering transformation AFT_π is used only to keep information of dependency pairs. Thus, introducing redundancy context does not destroy the soundness of argument filtering transformation. Therefore, we can define another argument filtering transformation $AFT_\pi^{\vec{C}_i}(R)$ as

$$AFT_\pi^{\vec{C}_i}(R) = \pi(R) \cup \{l_1 \to C_1[r_1], \ldots, l_n \to C_n[r_n]\}$$

where $\{l_1 \to r_1, \ldots, l_n \to r_n\} = \{\pi(l) \to \pi(r') \mid l \to r \in R,\ r' \in pick_\pi(dec_\pi(r))\}$ and $\vec{C}_i$ denotes the list of contexts $C_1, C_2, \ldots, C_n$.

Corollary 5.2.7

1. *If $AFT_{\pi}^{\vec{C}_i}(R)$ is AC-terminating and π satisfies the AC-condition then R is AC-terminating.*

2. *If $AFT_{\pi}^{\vec{C}_i}(R)$ is terminating and $\pi(f) = []$ for any AC-symbols f then R is AC-terminating.*

3. *[46] If $AFT_{\pi}^{\vec{C}_i}(R)$ is terminating then R is terminating.*

5.3 Comparison with Other Eliminations

In this section, we compare other elimination transformations with the argument filtering transformation. As a result, we conclude that the argument filtering transformation is a generalization of these elimination transformations.

5.3.1 Dummy Elimination

Definition 5.3.1 *[20](Dummy Elimination) Let e be a function symbol, called an eliminated symbol. The dummy elimination (DE_e) is defined as follows:*

- $$\begin{cases} cap_e(x) & = & x \\ cap_e(e(t_1,\ldots,t_n)) & = & \diamond \\ cap_e(f(t_1,\ldots,t_n)) & = & f(cap_e(t_1),\ldots,cap_e(t_n)) \quad \textit{if} \;\; f \neq e \end{cases}$$

- $$\begin{cases} dec_e(x) & = & \emptyset \\ dec_e(e(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n}(\{cap_e(t_i)\} \cup dec_e(t_i)) \\ dec_e(f(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n} dec_e(t_i) \quad \textit{if} \;\; f \neq e \end{cases}$$

- $DE_e(R) = \{cap_e(l) \to r' \mid l \to r \in R,\ r' \in \{cap_e(r)\} \cup dec_e(r)\}$

Example 5.3.2 *Let $t \equiv f(e(0, g(1, e(2,3))), 4)$. Then, $cap_e(t) = f(\diamond, 4)$ and $dec_e(t) = \{0,\ 2,\ 3,\ g(1,\diamond)\}$ (Figure 5.2).*

Proposition 5.3.3 *[20] If $DE_e(R)$ is terminating then R is terminating.*

Proposition 5.3.4 *[22] If $DE_e(R)$ is terminating and e is only AC-symbol (i.e., $\Sigma_{AC} = \{e\}$) then R is AC-terminating.*

For $\pi(e) = []$, we can treat the constant $\pi(e(\cdots))$ as $\diamond$.

Lemma 5.3.5 *For $\pi(e) = []$, $AFT_{\pi}(R) \subseteq DE_e(R)$.*

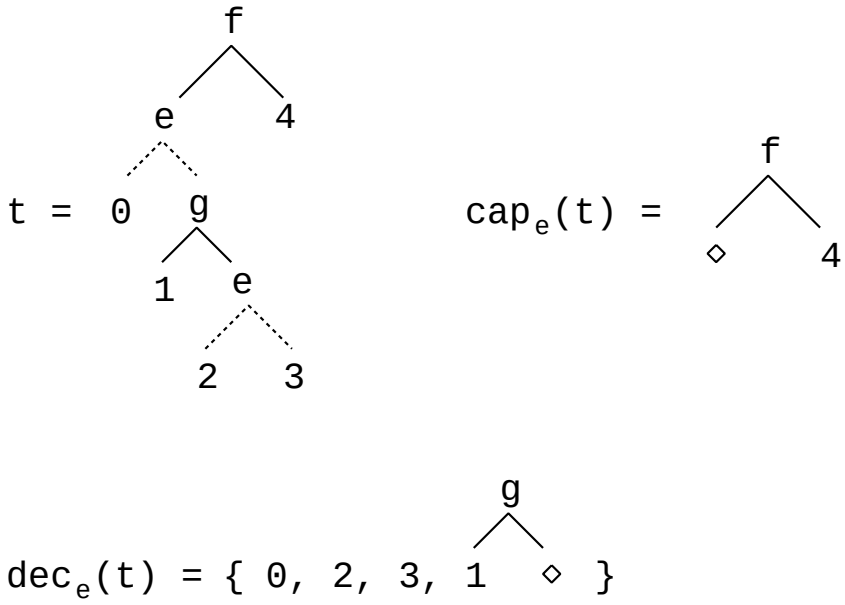

Figure 5.2: Dummy Elimination

Proof. It suffices to show that $\pi(dec_\pi(t)) = dec_e(t)$ by induction on t. In the case $t \in \mathcal{V}$, it is trivial. Suppose that $t \equiv f(t_1, \ldots, t_n)$. In the case $f \neq e$, it follows that $\pi(dec_\pi(f(t_1, \ldots, t_n))) = \bigcup_{i=1}^n \pi(dec_\pi(t_i)) = \bigcup_{i=1}^n dec_e(t_i) = dec_e(f(t_1, \ldots, t_n))$. In the case $f = e$, it follows that $\pi(dec_\pi(e(t_1, \ldots, t_n))) = \bigcup_{i=1}^n (\{\pi(t_i)\} \cup \pi(dec_\pi(t_i))) = \bigcup_{i=1}^n (\{cap_e(t_i)\} \cup dec_e(t_i)) = dec_e(e(t_1, \ldots, t_n))$. □

This lemma means that the argument filtering transformation is a proper extension of the dummy elimination. The corollary follows from the above lemma.

Corollary 5.3.6 *If $DE_e(R)$ is AC-terminating, simply AC-terminating, terminating or simply terminating then so is $AFT_\pi(R)$ with $\pi(e) = []$, respectively.*

The following corollary is a directly consequence of the above corollary.

Corollary 5.3.7 *If $DE_e(R)$ is AC-terminating then R is AC-terminating.*

Note that this corollary includes Propositions 5.3.3 and 5.3.4 as special cases, i.e., $\Sigma_{AC} = \emptyset$ and $\Sigma_{AC} = \{e\}$, respectively.

5.3.2 Distribution Elimination

Definition 5.3.8 *[67](Distribution Elimination) A rule $l \to r$ is a distribution rule for e if $l \equiv C[e(x_1, \ldots, x_n)]$ and $r \equiv e(C[x_1], \ldots, C[x_n])$ for some non-empty context C in which e does not occur and pairwise different variables $x_1, \ldots, x_n$. Let e be an eliminated symbol. The distribution elimination (DIS_e) is defined as follows:*

- $E_e(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ \bigcup_{i=1}^n E_e(t_i) & \text{if } t \equiv e(t_1, \ldots, t_n) \\ \{f(s_1, \ldots, s_n) \mid s_i \in E_e(t_i)\} & \text{if } t \equiv f(t_1, \ldots, t_n) \text{ with } f \neq e \end{cases}$

- $DIS_e(R) = \{l \to r' \mid l \to r \in R$ *is not a distribution rule for* e, $r' \in E_e(r)\}$

Example 5.3.9 *Let* $t \equiv f(e(0, g(1, e(2, 3))), 4)$.
Then, $E_e(t) = \{f(0, 4),\ f(g(1, 2), 4),\ f(g(1, 3), 4)\}$ *(Figure 5.3).*

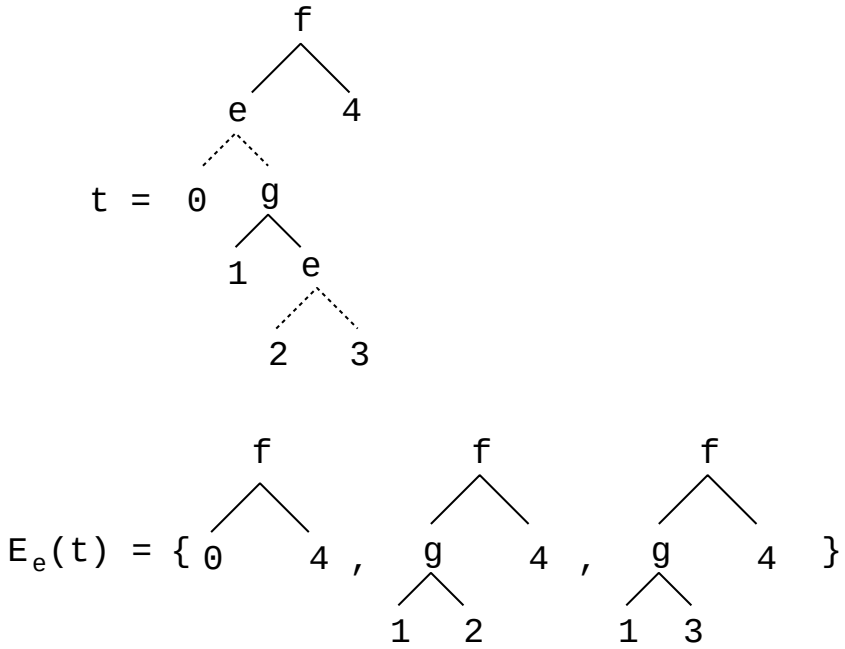

Figure 5.3: Distribution Elimination

Unfortunately, the distribution elimination is not sound with respect to AC-termination, i.e., AC-termination of $DIS_e(R)$ does not ensure AC-termination of R. With respect to termination it is also not sound. Thus, the distribution elimination requires suitable restrictions to ensure the soundness.

Proposition 5.3.10 *Suppose that each rule* $l \to r \in R$ *is a distribution rule or a rule in which the eliminated symbol* e *does not occur in* l.

(a) *[67] If* $DIS_e(R)$ *is terminating and right-linear then* R *is terminating.*

(b) *[53] If* $DIS_e(R)$ *is terminating and there exist no distribution rule in* R *then* R *is terminating.*

(c) *[58] If* $DIS_e(R)$ *is terminating, right-linear and* e *is only AC-symbol (i.e.* $\Sigma_{AC} = \{e\}$*) then* R *is AC-terminating.*

Lemma 5.3.11 *Let* $\pi(e) = i$ *such that* $1 \leq i \leq arity(e)$. *Under the condition of Proposition 5.3.10 (b),* $\pi(R) \subseteq DIS_e(R)$ *and* $AFT_\pi(R) \sqsubseteq DIS_e(R)$.

Proof. $\pi(R) \subseteq DIS_e(R)$ is trivial. From the definition of AFT_π, for any $l \to r \in AFT_\pi(R)$ there exists a rule $l' \to C'[r'] \in R$ with $l \equiv \pi(l')$ and $r \equiv \pi(r')$. Thus, it suffices to show that for any t and C', there exists a context C such that $C[\pi(t)] \in E_e(C'[t])$. It is easily proved by induction on C'. $\square$

Theorem 5.3.12 *Suppose that $\pi(e) = 1$, $e \notin \Sigma_{AC}$ and the condition of Proposition 5.3.10 (b). If $DIS_e(R)$ is AC-terminating or terminating then so is $AFT_\pi^{\vec{C}_i}(R)$ for some $\vec{C}_i$, respectively. If $DIS_e(R)$ is simply AC-terminating or simply terminating then so is $AFT_\pi(R)$, respectively.*

Proof. From Lemma 5.3.11, it is trivial. □

The following corollary is a directly consequence of the above theorem.

Corollary 5.3.13 *If $DIS_e(R)$ is AC-terminating, $e \notin \Sigma_{AC}$ and there exist no distribution rule in R then R is AC-terminating.*

Note that this corollary includes Proposition 5.3.10 as a special case, i.e., $\Sigma_{AC} = \emptyset$.

It is not so easy to treat the cases of conditions (a) and (c) in Proposition 5.3.10, because the distribution elimination eliminate distribution rules themselves. In order to treat such cases, we use the AC-multiset extension of the argument filtering method.

Theorem 5.3.14 *Suppose that each rule $l \to r \in R$ is a distribution rule or a rule in which the eliminated symbol e does not occur in l. If $DIS_e(R)$ is AC-terminating and right-linear then R is AC-terminating.*

Proof. Let R_D be the AC-TRS constructed by all distribution rule in R, and $R_0 = R \backslash R_D$. Let $>$ be $\xrightarrow[DIS_e(R)/AC]{+}$. Since $DIS_e(R)$ is AC-terminating, $>$ is an AC-reduction order. We denote DP_0 dependency pairs constructed from R_0, DP_0^{ex} extended dependency pairs constructed from R_0, DP_D dependency pairs constructed from R_D, and DP_D^{ex} extended dependency pairs constructed from R_D.

- $arity(e) = 1$: We choose $\pi(e) = 1$.

  It is trivial that $\pi(R_0) = DIS_e(R)$ and $\pi(l) \equiv \pi(r)$ for all $l \to r \in R_D$. Thus, if R is not AC-terminating then there exists an infinite reduction $t_1 \underset{R_D}{\to} t_2 \underset{R_D}{\to} t_3 \underset{R_D}{\to} \cdots$. However, R_D is trivially AC-terminating. It is a contradiction.

- $arity(e) > 1$: We choose $\pi(e) = 0$.

  First, we investigate non distribution rules. It is obvious that $\hat{\pi}(l) = \{l\}$ and $l \to r' \in DIS_e(R)$ for any $l \to r \in R_0$ and $r' \in \hat{\pi}(r)$. Thus, $l \gtrsim_\pi^{mul} r$ and $l >_\pi^{mul} r$ for any rule $l \to r \in R_0$. It follows that $s \underset{R_0}{\to} t$ implies $s \gtrsim_\pi^{mul} t$, because $\gtrsim_\pi^{mul}$ is monotonic and stable by the right-linearity of $DIS_e(R)$. Moreover, it follows that $u\theta >_\pi^{mul} v\theta$ for all θ and $\langle u, v\rangle \in DP_0^{ex}$.

  On the other hand, it is obvious that for any $\langle u, v\rangle \in DP_0$ and $v' \in \hat{\pi}(v)$, $\hat{\pi}(u) = \{u\}$ and $u \to C[v'] \in DIS_e(R)$ for some C. Thus, $u >_\pi^{mul} v$ for any $\langle u, v\rangle \in DP_0$. Since $DIS_e(R)$ is right-linear, it follows that $u\theta >_\pi^{mul} v\theta$ for all θ and $\langle u, v\rangle \in DP_0$.

  Next, we focus on the distribution rules. For any distribution rule $C[e(x_1, \ldots, x_n)] \to e(C[x_1], \ldots, C[x_n]) \in R_D$, it follows that $\hat{\pi}(C[e(x_1, \ldots, x_n)]) = \{C[x_1], \ldots, C[x_n]\}$ $=\hat{\pi}(e(C[x_1], \ldots, C[x_n]))$. Thus, $s \underset{R_D}{\overset{*}{\to}} t$ implies $s \gtrsim_\pi^{mul} t$. Moreover, $f(l, z)\theta \gtrsim_\pi^{mul} f(r, z)\theta$ for any $\langle f(l, z), f(r, z)\rangle \in DP_D^{ex}$ and θ. The dependency pair in DP_D can

be denoted by $\langle C[e(x_1,\ldots,x_n)], v\rangle$ such that $C'[v] \equiv C[x_i]$ for some i and C'. Since $\hat{\pi}(C[e(x_1,\ldots,x_n)]) = \{C[x_1],\ldots,C[x_n]\} \gg_{AC} \{C[x_i]\} = \{C'[v]\}$, it follows that $C[e(x_1,\ldots,x_n)]\theta >_\pi^{mul} v\theta$ for any θ.

Finally, we prove this theorem based on the above properties. Assume that R is not AC-terminating. From Theorem 4.1.10, there exists an infinite unmarked AC-dependency chain $\langle u_0, v_0\rangle\langle u_1, v_1\rangle\cdots$ and θ such that $v_i\theta \underset{R/AC}{\overset{*}{\to}} t_i \trianglerighteq_{hd} u_{i+1}\theta$ for some t_i $(i = 1, 2, \ldots)$. We have already proved that $v_i\theta \gtrsim_\pi^{mul} t_i$, $u_i\theta \gtrsim_\pi^{mul} v_i\theta$ for all $\langle u_i, v_i\rangle \in DP_D^{ex}$, and $u_i\theta >_\pi^{mul} v_i\theta$ for all $\langle u_i, v_i\rangle \in DP_{AC}(R)\backslash DP_D^{ex}$. From Lemma 4.2.28 and $\gtrsim_\pi^{mul} \subseteq >_\pi^{mul}$, it follows that $t_i \underset{AC}{\sim} u_{i+1}\theta$ or $t_i >_\pi^{mul} u_{i+1}\theta$. Hence there exists a number m such that $\langle u_i, v_i\rangle \in DP_D^{ex}$ for all $i \geq m$. Moreover, there exists an infinite AC-reduction sequence $f(l_0, z_0)\theta \underset{R_D}{\to} f(r_0, z_0)\theta \underset{R_D}{\overset{*}{\to}} f(l_1, z_1) \underset{R_D}{\to} f(r_1, z_1)\theta\cdots$ for some $l_i \to r_i \in R_D$ $(i = 0, 1, \ldots)$. However, R_D is trivially AC-terminating. It is a contradiction. □

Note that this theorem includes Proposition 5.3.10 (a) and (c) as special cases, i.e., $\Sigma_{AC} = \emptyset$ and $\Sigma_{AC} = \{e\}$, respectively.

5.3.3 General Dummy Elimination

For any $e \in \Sigma$, an e-status τ satisfy $\tau(e) = (\emptyset, 0)$ or (I, i) with $i \in I$.

Definition 5.3.15 *[21](General Dummy Elimination) Let e be an eliminated symbol and $\tau(e) = (I, i)$. The general dummy elimination (GDE_e) is defined as follows:*

- $$cap_i(t) = \begin{cases} t & \text{if } t \in \mathcal{V} \\ f(cap_i(t_1),\ldots,cap_i(t_n)) & \text{if } t \equiv f(t_1,\ldots,t_n) \wedge f \neq e \\ cap_i(t_i) & \text{if } t \equiv e(t_1,\ldots,t_n) \wedge i \neq 0 \\ \diamond & \text{if } t \equiv e(t_1,\ldots,t_n) \wedge i = 0 \end{cases}$$

- $$E_i(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ \{f(s_1,\ldots,s_n) \mid s_j \in E_i(t_j)\} & \text{if } t \equiv f(t_1,\ldots,t_n) \wedge f \neq e \\ E(t_i) & \text{if } t \equiv e(t_1,\ldots,t_n) \end{cases}$$

- $$E(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V} \\ \{cap_0(t)\} & \text{if } I = \emptyset \\ \bigcup_{j\in I} E_j(t) & \text{if } I \neq \emptyset \end{cases}$$

- $$dec(t) = \begin{cases} \emptyset & \text{if } t \in \mathcal{V} \\ \bigcup_{j=1}^n dec(t_j) & \text{if } t \equiv f(t_1,\ldots,t_n) \wedge f \neq e \\ \bigcup_{j=1}^n dec(t_j) \cup \bigcup_{j\notin I} E(t_j) & \text{if } t \equiv e(t_1,\ldots,t_n) \end{cases}$$

- $GDE_e(R) = \{cap_i(l) \to r' \mid l \to r \in R,\ r' \in E(r) \cup dec(r)\}$

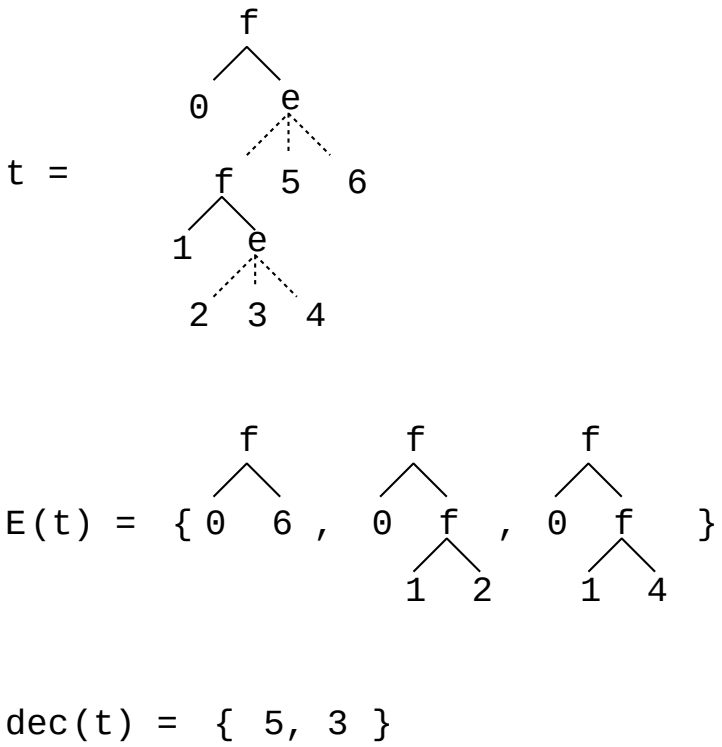

Figure 5.4: General Dummy Elimination

Example 5.3.16 *Let* $t \equiv f(0, e(f(1, e(2, 3, 4)), 5, 6))$ *and* $\tau(e) = (\{1, 3\}, 1)$. *Then,* $E(t) = \{f(0, 6),\ f(0, f(1, 2)),\ f(0, f(1, 4))\}$ *and* $dec(t) = \{5,\ 3\}$ *(Figure 5.4).*

Proposition 5.3.17 *[21] If* $GDE_e(R)$ *is terminating then* R *is terminating.*

Lemma 5.3.18 *Let* $\tau(e) = (I, i)$. *In the case* $\tau(e) = (\emptyset, 0)$, *we define* $\pi(e) = []$. *In the case* $e \notin \Sigma_{AC}$ *and* $\tau(e) = (I, i)$ *with* $i \in I$, *we define* $\pi(e) = i$. *Then* $\pi(R) \subseteq GDE_e(R)$ *and* $AFT_\pi(R) \sqsubseteq GDE_e(R)$.

Proof. $\pi(R) \subseteq GDE_e(R)$ is trivial. In the case $\tau(e) = (\emptyset, 0)$, it is trivial that $DE_e(R) = GDE_e(R)$. Thus, $AFT_\pi(R) \subseteq GDE_e(R)$. In the case $e \notin \Sigma_{AC}$ and $\tau(e) = (I, i)$ with $i \in I$, as similar to Lemma 5.3.11 by replacing $E_e(r)$ with $dec(r) \cup E(r)$. □

Theorem 5.3.19 *Suppose that* $e \notin \Sigma_{AC}$ *or* $\tau(e) = (\emptyset, 0)$. *If* $GDE_e(R)$ *is AC-terminating or terminating then so is* $AFT_\pi^{\vec{C}_i}(R)$, *respectively. If* $GDE_e(R)$ *is simply AC-terminating or simply terminating then so is* $AFT_\pi(R)$, *respectively.*

Proof. From Lemma 5.3.18, it is trivial. □

Theorem 5.3.20 *Suppose that* $GDE_e(R)$ *is AC-terminating.*

1. *If* $e \notin \Sigma_{AC}$ *or* $\tau(e) = (\emptyset, 0)$ *then* R *is AC-terminating.*
2. *If* $GDE_e(R)$ *is right-linear then* R *is AC-terminating.*

Proof. (1) A directly consequence of Theorem 5.3.19. (2) Suppose that $e \in \Sigma_{AC}$ and $\tau(e) = (I, i)$ with $i \in I$. We choose $\pi(e) = 0$. Thanks to $cap_i(l) \in \hat{\pi}(l)$, as similar to Theorem 5.3.14. $\square$

Note that this theorem (1) includes Proposition 5.3.17 as a special case, i.e., $\Sigma_{AC} = \emptyset$.

We give the following example that the argument filtering transformation can be applied to, but the general dummy elimination can not be.

Example 5.3.21 *Consider the AC-TRS*

$$R = \left\{ \begin{array}{rcl} g(a) & \to & g(b) \\ b & \to & f(a,a) \\ f(a,a) & \to & g(d) \end{array} \right.$$

with $\Sigma_{AC} = \{f\}$. *Let* $\pi(f) = []$. *Then,*

$$AFT_\pi(R) = \left\{ \begin{array}{rcl} g(a) & \to & g(b) \\ b & \to & f \\ f & \to & g(d) \end{array} \right.$$

The termination of $AFT_\pi(R)$ *is easily proved by the recursive path order with the precedence* $a \rhd b \rhd f \rhd g \rhd d$. *From Corollary 5.2.5,* R *is AC-terminating. We easily observe that the dummy elimination, the distribution elimination and the general dummy elimination can not be applied. Indeed, the following systems are clearly not terminating.*

$\tau(f)$	$GDE_f(R)$
$(\emptyset, 0)$	$g(a) \to g(b)$ $b \to \diamond$ $b \to a$ $\diamond \to g(d)$
$(\{1\}, 1)$	$g(a) \to g(b)$ $b \to a$ $a \to g(d)$

Note that the AC-termination of R *is not easily proved since* R *is not simply AC-terminating.*

5.3.4 Improved General Dummy Elimination

Definition 5.3.22 *[57](Improved General Dummy Elimination) The functions* cap_i, E *and dec are the same as that of the general dummy elimination. In the case* $e \in DF(R)$, *we take* $IGDE_e(R) = GDE_e(R)$. *Otherwise,*

- $E'(t) = \{s \in E(t) \mid \ s$ *includes some defined symbols of* $R\}$
- $dec'(t) = \{s \in dec(t) \mid \ s$ *includes some defined symbols of* $R\}$
- $IGDE_e(R) = \{cap_i(l) \to r' \mid l \to r \in R,\ r' \in \{cap_i(r)\} \cup E'(r) \cup dec'(r)\}$

Proposition 5.3.23 *[57] If $IGDE_e(R)$ is terminating then R is terminating.*

Lemma 5.3.24 *Let $\tau(e) = (I, i)$. In the case $\tau(e) = (\emptyset, 0)$, we define $\pi(e) = []$. In the case $e \notin \Sigma_{AC}$ and $\tau(e) = (I, i)$ with $i \in I$, we define $\pi(e) = i$. Then $\pi(R) \subseteq IGDE_e(R)$ and $AFT_\pi(R) \sqsubseteq IGDE_e(R)$.*

Proof. As similar to Lemma 5.3.18. □

Theorem 5.3.25 *Suppose that $e \notin \Sigma_{AC}$ or $\tau(e) = (\emptyset, 0)$. If $IGDE_e(R)$ is AC-terminating or terminating then so is $AFT_\pi^{\vec{C}_i}(R)$, respectively. If $IGDE_e(R)$ is simply AC-terminating or simply terminating then so is $AFT_\pi(R)$, respectively.*

Proof. From Lemma 5.3.24, it is trivial. □

Theorem 5.3.26 *Suppose that $IGDE_e(R)$ is AC-terminating.*

1. *If $e \notin \Sigma_{AC}$ or $\tau(e) = (\emptyset, 0)$ then R is AC-terminating.*
2. *If $IGDE_e(R)$ is right-linear then R is AC-terminating.*

Proof. As similar to Theorem 5.3.20. □

Note that this theorem (1) includes Proposition 5.3.23 as a special case, i.e., $\Sigma_{AC} = \emptyset$.

At the end, we give an example that the argument filtering transformation can be applied to, but other elimination transformations discussed here can not be.

Example 5.3.27 *Consider the AC-TRS*

$$R = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(f(x), x)) \\ f(f(x)) & \to & f(h(f(x), f(x))) \\ g(x, y) & \to & y \\ h(x, x) & \to & g(x, 0) \end{array} \right.$$

with $\Sigma_{AC} = \{h\}$. Let $\pi(g) = [2]$ and $\pi(h) = []$. Then,

$$AFT_\pi(R) = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ f(f(x)) & \to & f(x) \\ f(f(x)) & \to & f(h) \\ g(y) & \to & y \\ h & \to & g(0) \end{array} \right.$$

The termination of $AFT_\pi(R)$ is easily proved by the recursive path order with the precedence $f \rhd h \rhd g \rhd 0$. From Corollary 5.2.5, R is AC-terminating.

We observe that the improved general dummy elimination can not be applied. In fact, we can see that $IGDE_f(R)$, $IGDE_g(R)$ and $IGDE_h(R)$ are not terminating for all status τ. The dummy elimination, the distribution elimination and the general dummy elimination cannot be applied, too. Note that the AC-termination of R is not easily proved since R is not simply AC-terminating.

5.4 Comparison with Argument Filtering Method

In the previous section, we unified all elimination transformations by the argument filtering transformation, which is designed based on AC-dependency pairs and the argument filtering method. The reader might have the question which is more useful, the argument filtering method or the argument filtering transformation. Here, we say that the argument filtering method succeeds to prove AC-termination when Theorem 4.2.15 is applicable with a suitable AC-compatible simplification order, and we say that the argument filtering transformation succeeds to prove AC-termination when we prove the AC-termination of $AFT_\pi(R)$ by a suitable AC-compatible simplification order and Theorem 5.2.4 is applicable.

Suppose that $\pi(f) = []$ for some defined AC-symbol f. Then the argument filtering method always fails, because an extended dependency pair $\langle f(l,z)^\#, f(r,z)^\#\rangle$ for f produces $\langle f^\#, f^\#\rangle$ by π, which can not be ordered by the strict part of any weak AC-reduction order. Notice that $\pi(f) = []$ requires $\pi(f^\#) = []$ in Theorem 4.2.15. Hence the argument filtering transformation is more useful than the argument filtering method in such cases. Indeed, Examples 5.3.21 and 5.3.27 are such examples.

On the other hand, in the case $\pi(f) = [1,2]$ for all defined AC-symbols, if the argument filtering transformation succeed to prove the AC-termination of AC-TRS R by a simplification order then the argument filtering method also succeed to prove the AC-termination by the same simplification order, because $\pi(R) \cup \pi(DP(R)) \sqsubseteq AFT_\pi(R)$. The reader might have the question what kind of AC-TRSs the argument filtering method has an extra power. The following theorem is an answer.

Theorem 5.4.1 *Suppose that $\pi(R) \cup \pi(DP(R)) \sqsubseteq R'$ and $\pi(f) = [1,2]$ for all $f \in \Sigma_{AC} \cap DF(R)$. If $R \cup DP(R)$ is not AC-terminating then R' is not simply AC-terminating.*

Proof. Assume that R' is simply AC-terminating. We define $>$ as $\xrightarrow[R' \cup Emb/AC]{+}$. The AC-termination of R' ensures that $>$ is an AC-compatible simplification order. It is trivial that $\pi(R_D) \cup \pi(DP(R_D)) \sqsubseteq R'$ where $R_D = R \cup DP(R)$. Thus, $l \gtrsim_\pi r$ for all $l \to r \in R_D$ and $u \gtrsim_\pi v$ for all $\langle u, v\rangle \in DP(R_D)$. For any extended unmarked dependency pair $\langle f(l,z), f(r,z)\rangle$, it follows that $\pi(f(l,z)) \equiv f(\pi(l),z) > f(\pi(r),z) \equiv \pi(f(r,z))$. By regarding that $f^\#$ is identified to f, R_D is AC-terminating by Theorem 4.2.15. It is a contradiction. $\square$

In the case $\pi(f) = [1,2]$ for all defined AC-symbols, this theorem means that if $R \cup DP(R)$ is not AC-terminating then $AFT_\pi(R)$ is not simply AC-terminating, because $\pi(R) \cup \pi(DP(R)) \sqsubseteq AFT_\pi(R)$. Hence the argument filtering transformation with simplification orders always fails to prove the AC-termination of R.

Finally, we give two examples such that R is AC-terminating but $R \cup DP(R)$ is not AC-terminating. The AC-termination of these examples was proved by the argument filtering method in Example 4.2.16. Hence the argument filtering method is more useful than the argument filtering transformation for such AC-TRSs.

Example 5.4.2 *The following AC-TRSs R_2 and R_3 are displayed in Example 4.2.16. In both systems, $\Sigma_{AC} = \{h\}$.*

- *R_2 is AC-terminating, but $DP(R_2)$ and $R_2 \cup DP(R_2)$ are not AC-terminating.*

$$R_2 = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ g(x) & \to & h(f(x), f(x)) \end{array} \right. \qquad DP(R_2) = \left\{ \begin{array}{rcl} f(f(x)) & \to & f(g(x)) \\ f(f(x)) & \to & g(x) \\ g(x) & \to & f(x) \end{array} \right.$$

- *R_3, $DP(R_3)$ and $DP_{AC}(R_3)$ are AC-terminating, but $R_3 \cup DP(R_3)$ is not AC-terminating.*

$$R_3 = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ b & \to & g(h(a,a),a) \\ h(x,x) & \to & x \end{array} \right. \qquad DP(R_3) = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ f(a) & \to & b \\ b & \to & h(a,a) \end{array} \right.$$

Chapter 6

Conclusion

In this thesis, we have discussed the termination and the AC-termination property of term rewriting systems. We summarize the main results in this thesis.

First, we extend the notion of dependency pairs to AC-TRSs. It is impossible to directly apply the notion of dependency pairs to AC-TRSs. To avoid this difficulty, we introduce the head parts in terms and show an analogy between the root positions in infinite reduction sequences by TRSs and the head positions in those by AC-TRSs. Indeed, this analogy is essential for extensions of dependency pairs to AC-TRSs. Based on this analogy, we define AC-dependency pairs and AC-dependency chains.

Second, we extend argument filtering methods to AC-TRSs. Our extension gives a design of a weak AC-reduction order and a weak AC-reduction pair from an arbitrary AC-reduction order. Moreover, in order to strengthen the power of the argument filtering method, we improve the method in two directions. One is the lexicographic argument filtering method, in which argument filtering functions are lexicographically combined to compare AC-dependency pairs. Another one is an extension over multisets. In the argument filtering method on AC-TRSs, any argument filtering function must be compatible to AC-equations. We relax this restriction using the AC-multiset extension. These methods are effective for proving not only AC-termination but also termination of TRSs.

Next, we propose a powerful algorithm for generating an approximated AC-dependency graph, using the techniques of Ω-reduction and Ω_V-reduction, which are introduced to analyze decidable call-by-need computations in TRSs. Of course, our algorithm can also apply to TRSs, because TRSs are AC-TRSs without AC-symbols.

On the other hand, the AC-dependency pair method is useful for not only proving AC-termination but also analyzing other proving methods for AC-termination. We show that the argument filtering method combined with the dependency pair technique can clearly explain in a uniform framework why various elimination transformations work well.

Based on this observation, a new powerful elimination method, called the argument filtering transformation, is proposed. Since the transformation is carefully designed by removing all unnecessary rewrite rules generated by other elimination methods, it is the most powerful among all elimination methods.

Finally, we make the relation clear among various elimination methods through comparing them with corresponding restricted argument filtering transformation. For example, the dummy elimination method can be seen as a restricted argument filtering transformation in which each argument filtering always removes all arguments, and the distribution elimination method restricts each argument filtering into collapsing one.

Bibliography

[1] T.Arts, Automatically Proving Termination and Innermost Normalization of Term Rewriting Systems, Ph.D. thesis, Utrecht University, 1997.

[2] T.Arts, J.Giesl, Automatically Proving Termination where Simplification Orderings Fail, In *Proc. of 7th Int. Joint Conf. on Theory and Practice of Software Development*, LNCS 1214 (TAPSOFT'97), pp.261–272, 1997.

[3] T.Arts, J.Giesl, Proving Innermost Normalization Automatically, In *Proc. of 8th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1232 (RTA'97), pp.157–171, 1997.

[4] T.Arts,J.Giesl, Applying Rewriting Techniques to the Verification of Erlang Processes, In *Proc. of 8th Int. Conf. on* LNCS 1683 (CSL'99), pp.96–110, 1999.

[5] T.Arts, J.Giesl, Termination of Term Rewriting Using Dependency Pairs, to appear in *Theoretical Computer Science*.

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

[7] L.Bachmair, D.A.Plaisted, Termination Orderings for Associative-Commutative Rewriting Systems, *J. Symbolic Computation*, vol. 1, pp.329–349 ,1985.

[8] L.Bachmair, Associative-Commutative Reduction Orderings, *Information Processing Letters*, vol.43, pp.21–27, 1992.

[9] A.B.Cherifa, P.Lescanne, An Actual Implementation of a Procedure that Mechanically Proves Termination of Rewriting Systems based on Inequalities between Polynomial Interpretations, In *Proc. of 8th Int. Conf. on Automated Deduction*, LNCS 230, pp.42–51, 1986.

[10] A.Church, An Unsolvable Problem of Elementary Number Theory, *American Journal of Mathematics*, vol.58, pp.345–363, 1936.

[11] B.Courcelle, Recursive Applicative Program Schemes, in *Handbook of Theoretical Computer Science*, vol.B, ed. J. van Leeuwen, North-Holland, pp.459–492, 1990.

[12] N.J.Cutland, Computability, Cambridge University Press, 1980.

[13] M.Dauchet, Simulation of Turing Machines by a Regular Rewrite Rule, *Theoretical Computer Science*, vol.103(2), pp.409–420, 1992.

[14] C.Delor, L.Puel, Extension of the Associative Path Ordering to a Chain of Associative Commutative Symbols, In *Proc. of 5th Int. Conf. on Rewriting Techniques and Applications*, LNCS 690 (RTA'93), pp.389–404, 1993.

[15] N.Dershowitz, A Note on Simplification Orderings, *Information Processing Letters*, 9(5), pp.212–215, 1979.

[16] N.Dershowitz, Orderings for Term-Rewriting Systems, *Theoretical Computer Science*, vol.17(3), pp.279–301, 1982.

[17] N.Dershowitz, Termination of Rewriting, *J. Symbolic Computation* 3, pp.69–115, 1987.

[18] N.Dershowitz, J.-P.Jouannaud, Rewrite Systems, in *Handbook of Theoretical Computer Science*, vol.B, ed. J. van Leeuwen, North-Holland, pp.243–320, 1990.

[19] N.Dershowitz, Z.Manna, Proving Termination with Multiset Orderings, *Communications of the ACM* 22(8), pp.465–476, 1979.

[20] M.Ferreira, H.Zantema, Dummy Elimination: Making Termination Easier, In *Proc. of 10th Int. Conf. on Fundamentals of Computation Theory*, LNCS 965 (FCT'95), pp.243–252, 1995.

[21] M.Ferreira, Termination of Term Rewriting, Well-foundedness, Totality and Transformations, Ph.D. thesis, Utrecht University, 1995.

[22] M.Ferreira, D.Kesner, L.Puel, Reducing AC-Termination to Termination, In *Proc. of 23rd Int. Symp. on Mathematical Foundations of Computer Science*, LNCS 1450 (MFCS'98), pp.239–247, 1998.

[23] J.Giesl, E.Ohlebusch, Pushing the Frontiers of Combining Rewrite Systems Farther Outwards, In *Proc. of 2nd Int. Workshop on Frontiers of Combining Systems* (FroCos '98), Amsterdam, The Netherlands, October 1998.

[24] K.Gödel, Über Formal Unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme I, *Monatschefte für Mathematik und Physik*, vol.38, pp.173–198, 1931.

[25] K.Gödel, On Undecidable Propositions of Formal Mathematical Systems, *mimeographed lecture notes*, Institute for Advanced Study Princeton, N.J., 30pp., 1934.

[26] B.Gramlich, Termination and Confluence Properties of Structured Rewrite Systems, Ph.D. thesis, Universität Kaiserslautern, Germany, 1996.

[27] G.Huet, D.Lankford, On the Uniform Halting Problem for Term Rewriting Systems, Technical Report 283, INRIA, 1978.

[28] G.Huet, J.-J.Levy, Computations in Orthogonal Rewriting Systems, I and II, in *Computational Logic, Essays in Honor of Alan Robinson*, eds. J.-L.Lassez and G.Plotkin, MIT Press, pp.396-443, 1991.

[29] G.Huet, D.C.Oppen, Equations and Rewrite Rules: a Survey, In *Formal Languages Theory: Perspectives and Open Problems*, R. Book, Ed. Academic Press, New York, pp.349–405, 1980.

[30] J.-P.Jouannaud, P.Lescanne, On Multiset Orderings, *Information Processing Letters* 15(2), pp.57–63, 1982.

[31] J.-P.Jouannaud, P.Lescanne, F.Reinig, Recursive Decomposition Ordering, In *Working Conference on Formal Description of Programming Concepts* II (IFIP), D.Bjϕrner, Ed., North-Holland Publishing Company, pp.331–353, 1982.

[32] S.Kamin, J.Lévy, Two Generalizations of the Recursive Path Ordering, University of Illinois at Urbana-Champaign, Unpublished manuscript, 1980.

[33] D.Kapur, P.Narendran, G.Sivakumar, A Path Ordering for Proving Termination of Term Rewriting Systems, In *Proc. of 10th Colloquium on Trees in Algebra and Programming*, LNCS 185 (CAAP–10), pp.173–187, 1985.

[34] D.Kapur, G.Sivakumar, H.Zhang, A New Method for Proving Termination of AC-Rewrite Systems, In *Proc. of 10th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science*, LNCS 472, pp.134–148, 1990.

[35] D.Kapur, G.Sivakumar, A Total Ground Path Ordering for Proving Termination of AC-Rewrite Systems, In *Proc. of 8th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1232 (RTA'97), pp.142–156.

[36] S.C.Kleene, General Recursive Functions of Natural Numbers, *Mathematische Annalen*, vol. 112, pp.727–742, 1936.

[37] S.C.Kleene, A Note on Recursive Functions, *Bulletin of the American Mathematical Society*, vol.42, pp.544–546, 1936.

[38] S.C.Kleene, λ-Definability and Recursiveness, *Duke Mathematical Journal*, vol.2, pp.340–353, 1936.

[39] J.W.Klop, Term Rewriting Systems, *Handbook of Logic in Computer Science* II, pp.1–112, Oxford University Press, 1992.

[40] D.E.Knuth, P.B.Bendix, Simple Word Problems in Universal Algebra, In J.Leech Ed., *Computational Problems in Abstract Algebra*, pp.263–297, Pergamon Press, Oxford, U.K., 1970.

[41] M.Kurihara, A.Ohuchi, Modularity of Simple Termination of Term Rewriting Systems with Shared Constructors, *Theoretical Computer Science*, vol.103, pp.273–282, 1992.

[42] J.B.Kruskal, Well-quasi-ordering, the Tree Theorem, and Vazsonyi's Conjecture, *Trans. Amer. Math. Soc.*, vol.95, pp.210-225, 1960.

[43] K.Kusakari, Y.Toyama, On Proving AC-Termination by AC-Dependency Pairs, Research Report IS-RR-98-0026F, School of Information Science, JAIST, 1998.

[44] K.Kusakari, Y.Toyama, On Proving AC-Termination by Argument Filtering Method, Research Report IS-RR-99-0006F, School of Information Science, JAIST, 1999.

[45] K.Kusakari, Y.Toyama, The Hierarchy of Dependency Pairs, Research Report. IS-RR-99-0007F, School of Information Science, JAIST, 1999.

[46] K.Kusakari, M.Nakamura, Y.Toyama, Argument Filtering Transformation, In *Proc. of Int. Conf. on Principles and Practice of Declarative Programming*, LNCS 1702 (PPDP'99), pp.47–61, 1999.

[47] D.Lankford, Canonical Algebraic Simplification in Computational Logic, Technical Report ATP-25, Department of Mathematics, University of Texas, Austin, 1975.

[48] D.Lankford, A.M.Ballantyne, Decision Procedures for Simple Equational Theories with a Commutative Axiom: Complete Sets of Commutative Reductions, Technical Report, Mathematics Dep., Univ. of Texas, Austin, Texas, March 1977.

[49] D.Lankford, A.M.Ballantyne, Decision Procedures for Simple Equational Theories with Permutative Axioms: Complete Sets of Permutative Reductions, Technical Report, Mathematics Dep., Univ. of Texas, Austin, Texas, April 1977.

[50] D.Lankford, On Proving Term Rewriting Systems are Noetherian, Technical Report MTP-3, Mathematics Department, Louisiana Tech University, Ruston, 1979.

[51] Z.Manna, S.Ness, On the Termination of Markov Algorithms, In *Proc. 3rd Hawaii Int. Conf. System Science*, pp.789–792, 1970.

[52] C.Marché, X.Urbain, Termination of Associative-Commutative Rewriting by Dependency Pairs, In *Proc. of 9th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1379 (RTA'98), pp.241–255, 1998.

[53] A.Middeldorp, H.Ohsaki, H.Zantema, Transforming Termination by Self-Labeling, In *Proc. of 13th Int. Conf. on Automated Deduction*, LNCS 1104 (CADE-13), pp.373–387, 1996.

[54] A.Middeldorp, H.Zantema, Simple Termination of Rewrite Systems, *Theoretical Computer Science* 175(1), pp. 127–158, 1997.

[55] T.Nagaya, M.Sakai, Y.Toyama, Index Reduction of Overlapping Strongly Sequential Systems, *IEICE Trans. Inf. & Syst.*, vol.E81-D, No.5, pp.419–426, 1998.

[56] T.Nagaya, Reduction Strategies for Term Rewriting Systems, Ph.D. thesis, Japan Advanced Institute of Science and Technology, Japan, 1999.

[57] M.Nakamura, K.Kusakari, Y.Toyama, On Proving Termination by General Dummy Elimination, *IEICE Transactions on Information and Systems*, vol. J82-D-I, No.10, pp.1225–1231, 1999. (in Japanese)

[58] H.Ohsaki, A.Middeldorp, J.Giesl, Equational Termination by Semantic Labeling, Technical Report TR-98-29, Electrotechnical Laboratory, 1998.

[59] M. Oyamaguchi, NV-Sequentiality: A decidable condition for call-by-need computations in term rewriting systems, *SIAM J. Computation*, vol.22, no.1, pp.112–135, 1993.

[60] G.Peterson, M.Stickel, Complete Sets of Reductions for some Equational Theories, *Journal of the Association for Computing Machinery*, vol.28, No.2, pp.233–264, 1981.

[61] A.Rubio, R.Nieuwenhuis, A Total AC-compatible Ordering based on RPO, *Theoretical Computer Science*, vol.142, pp.209–227, 1995.

[62] A.Rubio, A Fully Syntactic AC-RPO, In *Proc. of 10th Int. Conf. on Rewriting Techniques and Applications*, LNCS 1631 (RTA'99), pp.133–147, 1999.

[63] J.Steinbach, Termination of Rewriting-Extension, Comparison and Automatic Generation of Simplification orderings, Ph.D. Thesis, University of Kaiserslautern, 1994.

[64] J.Steinbach, Simplification Ordering: History of Results, *Fundamenta Informaticae*, vol.24, pp.44–87, 1995.

[65] A.M.Turing, On Computable Numbers with an Application to the Entscheidungsproblem, In *Proceedings of the London Mathematical Society*, vol.42, pp.230–265; vol.43, pp.544–546, 1936.

[66] A.M.Turing, Computability and λ-definability, *J.Symbolic Logic*, vol.2, pp.153–163, 1937.

[67] H.Zantema, Termination of Term Rewriting: Interpretation and Type Elimination, *Journal of Symbolic Computation* 17, pp.23–50, 1994.

Publications

[1] K.Kusakari, M.Sakai, Y.Toyama, Church-Rosser Property of Non-Linear Term Rewriting Systems, In *Proc. the Joint Conf. of Hokuriku Chapters of Institutes of Electrical Engineers*, Japan, E–26, p.312, 1995 (in Japanese).

[2] K.Kusakari, M.Sakai, Y.Toyama, Church-Rosser Property of Non-Linear Term Rewriting Systems, Tech. Rep. of IEICE, COMP95–86(1996–01), pp.123–129, 1996 (in Japanese).

[3] K.Kusakari, M.Sakai, Y.Toyama, Church-Rosser Property of Finite Ranked Terms of Non-Linear Term Rewriting Systems, In *Proc. LA Symposium '96*, Summer, pp. 160–165, 1996.

[4] K.Kusakari, Y.Toyama, Designing a Termination Prover for Term Rewriting Systems, In *Proc. 14th Conf. on Japan Society for Software Science and Technology*, pp. 361–364, 1997 (in Japanese).

[5] K.Kusakari, Y.Toyama, On Proving AC-Termination by Dependency Pairs, Tech. Rep. of IEICE, COMP97–112(1998–03), pp. 47–54, 1998.

[6] K.Kusakari, Y.Toyama, On Proving AC-Termination by AC-Dependency Pairs, Res. Rep. IS-RR-98-0026F, School of Information Science, JAIST, 1998.

[7] K.Kusakari, Y.Toyama, On Proving Termination of Term Rewriting Systems by Pruning Method, Tech. Rep. of IEICE, COMP98-432(1998-11), pp. 49-56, 1998.

[8] K.Kusakari, Y.Toyama, On Proving AC-Termination by Argument Filtering Method, Res. Rep. IS-RR-99-0006F, School of Information Science, JAIST, 1999.

[9] K.Kusakari, Y.Toyama, The Hierarchy of Dependency Pairs, Res. Rep. IS-RR-99-0007F, School of Information Science, JAIST, 1999.

[10] K.Kusakari, M.Nakamura, Y.Toyama, Argument Filtering Transformation, Res. Rep. IS-RR-99-0008F, School of Information Science, JAIST, 1999.

[11] K.Kusakari, Y.Toyama, Designing a Weak AC-Reduction Pair by Argument Filtering Method, In *LA Symposium '99*, Summer, pp.12.1–12.6, 1999.

[12] M.Nakamura, K.Kusakari, Y.Toyama, On Proving Termination by General Dummy Elimination, *IEICE Transactions on Information and Systems*, Vol. J82–D–I, No.10, pp.1225–1231, 1999. (in Japanese)

[13] K.Kusakari, M.Nakamura, Y.Toyama, Argument Filtering Transformation, In *Proc. of Int. Conf. on Principles and Practice of Declarative Programming*, LNCS 1702 (PPDP'99), pp.47–61, 1999.

Index

AFT_{π_1}, 54
$AFT_{\pi}^{C_i}(R)$, 55
$arity(f)$, 9
$A/\sim$, 6
$\Box$, 10
$C[\ ]$, 10
$\mathcal{O}_{hd}(t)$, 29
$\mathcal{T}$, 48
$\mathcal{T}_{\Omega}$, 48
$\mathcal{V}$, 9
cap_e, 56
cap_i, 60
$C[t_1, \ldots, t_n]$, 10
$C[t]_p$, 10
dec, 60
dec', 62
dec_e, 56
dec_{π}, 54
DE_e, 56
$DF(R)$, 10
DIS_e, 58
$DP_{AC}(R)$, 29
$DP^{\#}_{AC}(R)$, 33
$DP(R)$, 15
$DP^{\#}_{AC}(R)$, 32
$DP^{\#}(R)$, 15
$DP^{\#}(R^{AC})$, 33
E, 60
E', 62
E_e, 58
E_i, 60
Emb, 12
ε, 9
$\underset{AC}{\sim}$, 10
$\equiv$, 9
$>$, 6
$>_A$, 11, 21, 47
GDE_e, 60
$\geq$, 6
$\gg$, 7
$\gg^{sub}_{AC}$, 42
$\gtrsim$, 8
$\gtrsim_{AC}$, 42
$\gtrsim^{sub}_{AC}$, 42
$>_{lex}$, 9
$>_{lpo}$, 13
$\underset{\mathcal{R}}{>}$, 6
$\underset{\mathcal{R}}{>}^{sub}_{AC}$, 36
$\underset{\Omega}{\geq}$, 47
$>_{\pi}$, 19, 37
$>^{mul}_{\pi}$, 43
$>_{rpo}$, 12
$>^{flat}_{rpo}$, 13
$\underset{\sim}{>}$, 6
$\underset{\sim}{>}_A$, 21, 47
$\underset{\sim}{>}_{AC}$, 36
$\underset{\sim}{>}^{sub}_{AC}$, 36
$\underset{\sim}{>}_E$, 8
$\underset{\sim}{>}_{\pi}$, 19, 37
$\underset{\sim}{>}^{mul}_{\pi}$, 43
$>^{sub}$, 19
$IGDE_e$, 62
$isConnect$, 49
$isConnect_n$, 49
$[\![x]\!]$, 6
$<$, 6
$\leq$, 6
$\underset{\sim}{<}$, 6
$M(A)$, 6
$\#$, 15
$NF_{\Omega}(t)$, 48
Ω, 47
Ω-term, 47
π, 19
$pick_{\pi}$, 54
$\hat{\pi}$, 42
$\hat{\pi}(\theta)$, 42
$\hat{\pi}(\theta)(T)$, 42

$\pi(\theta)$, 19
$\prec$, 9
$\xrightarrow[\Omega]{bd}$, 48
$\xrightarrow[\Omega_V]{bd}$, 48
$\xrightarrow[\Omega_V]{bd}{}^n$, 48
R^{AC}, 33
$\xrightarrow[R]{}$, 10
$\xrightarrow[R/AC]{}$, 10
$\xrightarrow[R]{}{}_p$, 10
$\xrightarrow{bd}$, 29
$\xrightarrow{hd}$, 29
$\xrightarrow{\#}$, 32
$\rhd$, 12, 13
$R^{\#}$, 32
Σ, 9
Σ_{AC}, 10
$\Sigma^{\#}_{AC,\pi}$, 36
$\Sigma^{\#}_{AC}$, 32
$\Sigma^{\#}_{\pi}$, 19
$\Sigma^{\#}$, 15
$\sim$, 6
$\sqsubseteq$, 52
$|t|$, 9
τ, 60
$T_{bd}(t)$, 29
$t \downarrow_{\#}$, 32
$\mathcal{T}(\Sigma, \mathcal{V})$, 9
t_{Ω}, 48
$t^{\#}$, 15, 32
$t|_p$, 9
$(t)_p$, 9
$\trianglerighteq_{hd}$, 29
$\uparrow$, 48
Υ^*, 6
Υ^+, 6
Υ^{-1}, 5
$\Upsilon^=$, 6
Υ^n, 5
$Var(t)$, 9

AC-compatible, 11, 12, 34, 35
AC-compatible simplification order, 12
AC-condition, 36
AC-deletion property, 34, 35
AC-dependency chain, 33
AC-dependency graph, 47
AC-dependency pair, 32
AC-extension, 36
AC-function symbol, 10
AC-marked condition, 34, 35
AC-multiset extension, 42
AC-reduction order, 11
AC-reduction relation, 10
AC-term rewriting system, 10
AC-terminating, 11
AC-TRS, 10
AC-unifiable, 10
antisymmetric, 5
approximated dependency graph, 27
arc, 25
argument filtering function, 19
argument filtering method, 19
argument filtering transformation, 54, 55
arity, 9

binary relation, 5
    composition, 5
    inverse relation, 5
body AC-Ω-reduction relation, 48
body AC-Ω_V-reduction, 48

closure, 6
cluster, 25
compatible, 5, 48
congruence relation, 10
context, 10
cycle, 25

defined symbol, 10
dependency chain, 16
dependency graph, 25
dependency pair, 15
directed graph, 25
distribution elimination, 57
dummy elimination, 56

e-status, 60
embedding TRS, 12
equivalence class, 6
equivalence part, 6
equivalence relation, 6

finite multiset, 6

flattening term, 13
function symbol, 9

general dummy elimination, 60

head position, 29

irreflexive, 5

lexicographic argument filtering method, 40
lexicographic extension, 9
lexicographic path order, 13

monomial, 11
monotone polynomial, 11
monotone polynomial strict order, 11
monotonic, 10
multiset, 6
  operation, 7, 8
  relation, 7, 8
multiset extension
  for quasi-orders, 8
  for strict orders, 7

n-approximated AC-dependency graph, 49
node, 25

partial order, 6
path, 25
polynomial, 11
polynomial interpretation, 11
polynomial quasi-order, 47
polynomial strict order, 11
precedence, 12, 13
prefix order, 9
proper subterm, 10

quasi-order, 6
quotient set, 6

recursive path order, 12
recursive program schema, 20
reduction order, 11
reduction relation, 10
reflexive, 5
reflexive closure, 6
reflexive-transitive closure, 6
rewrite order, 11
rewrite rule, 10
root reduction, 10
RPS, 20

signature, 9
simplification order, 12
simply terminating, 12
size, 9
stable, 10
strict order, 6
strict part, 6
strongly normalizing, 6
substitution, 9
subterm, 10
subterm property, 12
symmetric, 5

term, 9
term position, 9
term rewriting system, 10
terminating, 6
transitive, 5
transitive closure, 6
TRS, 10

unifiable, 9
unmarked AC-dependency chain, 30
unmarked AC-dependency pair, 29
unmarked extended dependency pair, 29

variable, 9

weak AC-reduction order, 34
weak AC-reduction pair, 35
weak reduction order, 16
weak reduction pair, 18
well-founded, 6