Argument Filtering Transformation

Keiichirou Kusakari Masaki Nakamura Yoshihito Toyama

School of Information Science, JAIST
Tatsunokuchi, Ishikawa 923-1292, Japan
email : {kusakari, masaki-n, toyama}@jaist.ac.jp

Abstract. To simplify the task of proving termination of term rewriting systems, several elimination methods, such as the dummy elimination, the distribution elimination, the general dummy elimination and the improved general dummy elimination, have been proposed. In this paper, we show that the argument filtering method combining with the dependency pair technique is essential in all the above elimination methods. We present remarkable simple proofs for the soundness of these elimination methods based on this observation. Moreover, we propose a new elimination method, called the argument filtering transformation, which is not only more powerful than all the other elimination methods but also especially useful to make clear the essential relation hidden behind these methods.

Keywords: Term Rewriting System, Termination, Elimination Method, Dependency Pair, Argument Filtering

1 Introduction

Term Rewriting Systems (TRSs) can be regarded as a model for computation in which terms are reduced, using a set of directed equations. They are used 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 [5, 9].

Termination is one of the most fundamental properties of term rewriting systems. While in general termination of TRSs is an undecidable property, several methods for proving termination have been developed. To simplify the task of proving termination of TRSs to which these methods cannot be directly applied, several elimination methods have been proposed. Elimination methods try to transform a given TRS into a TRS whose termination is easier to prove than the original one. The dummy elimination [6], the distribution elimination [13, 16], the general dummy elimination [7] and the improved general dummy elimination [15] are examples of elimination methods.

Recently, Arts and Giesl proposed the notion of the dependency pair, which can offer an effective method for analyzing an infinite reduction sequence [1–3]. Using dependency pairs, we can easily show the termination property of TRSs to which traditional techniques cannot be applied. Since this method compares rewrite rules and dependency pairs by a weak reduction pair instead of a reduction order, to find an appropriate weak reduction pair for a given TRS is necessary. The argument filtering method introduced [4, 8] allows us to make a weak reduction pair from an arbitrary reduction order.

In this paper, we first extend the argument filtering method by combining subterm relation. Next, we study the relation between the argument filtering method and various elimination methods. The key of our result is the observation that the argument filtering method combining with the dependency pair technique is essential in all the above elimination methods. Indeed, we present remarkable simple proofs for the soundness of these elimination methods based on this observation, though the original proofs presented in the literatures [6, 7, 13, 15, 16] are complicated and treated as rather different methods respectively. This observation also leads us to a new powerful elimination method, called the argument filtering transformation, which is not only more powerful than all the other elimination methods but also especially useful to make clear the essential relation hidden behind these methods. The main contributions of this paper are as follows:

(1) We show that the argument filtering method combining with the dependency pair technique can clearly explain in a uniform framework why various elimination methods work well.
(2) 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 these elimination methods.
(3) 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.

The remainder of this paper is organized as follows. The next section gives the definition of term rewriting systems. In section 3, we explain the dependency pair technique and introduce a new argument filtering method. Using these results, we show a general and essential property for elimination methods to be sound with respect to termination. In section 4, we propose the argument filtering transformation and show the soundness of this transformation. In section 5, we compare various elimination methods with the argument filtering transformation, and give simple proofs for the soundness of these elimination methods.

2 Preliminaries

We assume that the reader is familiar with notions of term rewriting systems [5].

A signature Σ is a finite set of function symbols, where each $f \in \Sigma$ is associated with a non-negative integer 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 constructed from Σ and $\mathcal{V}$ is written by $\mathcal{T}(\Sigma, \mathcal{V})$. $Var(t)$ is the set of variables in t. We define $root(f(t_1, \ldots, t_n)) = f$. Identity of terms is denoted by $\equiv$. A substitution $\theta : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is a mapping. A substitution over terms is defined as a homomorphic extension. We write $t\theta$ instead of $\theta(t)$. A context is a term which has a special constant $\Box$, called a hole. A term $C[t]$ denotes the result of replacing t in the hole of C.

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 set of rules. The set of defined symbols in a TRS R is denoted by $DF(R) = \{root(l) \mid l \to r \in R\}$. A reduction relation $\underset{R}{\to}$ is defined as follows: $s \underset{R}{\to} t \overset{\text{def}}{\iff} \exists\, l \to r \in R, \exists C[\ ], \exists\theta\ (s \equiv C[l\theta] \wedge t \equiv C[r\theta])$. We often omit the subscript R whenever no confusion arises. A TRS R is terminating if there is no infinite sequence such that $t_0 \underset{R}{\to} t_1 \underset{R}{\to} t_2 \underset{R}{\to} \cdots$. The transitive-reflexive closure and the transitive closure of $\to$ are denoted by $\overset{*}{\to}$ and $\overset{+}{\to}$, respectively.

A strict order $>$ is a reduction order if $>$ is well-founded, monotonic ($s > t \Rightarrow C[s] > C[t]$) and stable ($s > t \Rightarrow s\theta > t\theta$). Note that a TRS R is terminating iff there exists a reduction order $>$ that satisfies $l > r$ for all $l \to r \in R$. A reduction order $>$ is a simplification order if $C[t] > t$ for all t and C ($\not\equiv \Box$). A TRS R is simply terminating if there exists a simplification order $>$ that satisfies $l > r$ for all $l \to r \in R$.

3 Soundness Condition for Transformation

In this section, we first explain the dependency pair and the argument filtering method, whose notions greatly extend the provable class of termination. Using these notions, we show a theorem, which makes a general and essential property clear for transformations of TRSs to be sound with respect to termination.

Definition 3.1. *[1–3] $\Sigma^\# = \{f^\# \mid f \in \Sigma\}$ is the set of marked symbols disjoint from $\Sigma \cup \mathcal{V}$. We define the root-marked terms by $(f(t_1, \ldots, t_n))^\# = f^\#(t_1, \ldots, t_n)$. The set of the dependency pairs of R, written by $DP^\#(R)$, is $\{\langle u^\#, v^\# \rangle \mid u \to C[v] \in R,\ root(v) \in DF(R)\}$. The set of the unmarked dependency pairs of R, written by $DP(R)$, is obtained by erasing marks of symbols in $DP^\#(R)$.*

Example 3.2. *Let $R = \{add(x, 0) \to x,\ add(x, s(y)) \to s(add(x, y))\}$. Then,*
$DP^\#(R) = \{\langle add^\#(x, s(y)), add^\#(x, y)\rangle\}$. and
$DP(R) = \{\langle add(x, s(y)), add(x, y)\rangle\}$

Definition 3.3. *A pair $(\gtrsim, >)$ of binary relations on terms is a weak reduction pair if it satisfies the following three conditions:*

- $\gtrsim$ is monotonic and stable

– $>$ is stable and well-founded
– $\gtrsim \cdot > \ \subseteq \ >$ or $> \cdot \gtrsim \ \subseteq \ >$

In the above definition, we do not assume that $\gtrsim$ is a quasi-order (reflexive and transitive) or $>$ is a strict order (irreflexive and transitive). 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 the 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.

Theorem 3.4. *For any TRS R, the following three properties are equivalent.*

1. TRS R is terminating.
2. There exists a weak reduction pair $(\gtrsim, >)$ such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u, v \rangle \in DP(R).\ u > v$.
3. There exists a weak reduction pair $(\gtrsim, >)$ such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u^\#, v^\# \rangle \in DP^\#(R).\ u^\# > v^\#$.

Proof. $(1 \Rightarrow 2)$ We define $s \gtrsim t$ by $s \xrightarrow[R]{*} t$, and $s > t$ by $s \xrightarrow[R]{+} C[t]$ for some C. Then, it is trivial that $(\gtrsim, >)$ is a weak reduction pair such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u, v \rangle \in DP(R).\ u > v$. $(2 \Rightarrow 3)$ It is easily shown by identifying $f^\#$ with f. $(3 \Rightarrow 1)$ This case has already been shown in [1–4, 8]. □

Note that the proofs for $(2 \Rightarrow 1)$ and for $(1 \Rightarrow 3)$ have already been shown in [10, 14] and [4] [1], respectively.

The above theorem shows that the weak reduction pair plays an important role. To design a weak reduction pair, the argument filtering method introduced in [4, 8] is very useful, which is defined as recursive program schemata [9]. In the next definition, we introduce a new argument filtering method by combining the subterm relation, which is more effective in our framework than original one.

Definition 3.5. *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). We can naturally extend π over terms as follows:*

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

For any argument filtering function π and any reduction order $>$, we define the pair $(\gtrsim_\pi, >_\pi)$ as follows:

$$s \gtrsim_\pi t \overset{\text{def}}{\iff} \pi(s) > \pi(t) \ \text{ or } \ \pi(s) \equiv \pi(t)$$
$$s >_\pi t \overset{\text{def}}{\iff} \exists C.\ \pi(s) > C[\pi(t)] \ \text{ or } \ \exists C \not\equiv \square.\ \pi(s) \equiv C[\pi(t)]$$

[1] The proof for $(1 \Rightarrow 3)$ in [4] is based on the claim that if R is terminating then so is $R \cup DP^\#(R)$. However, the same proof method can not work well for $(1 \Rightarrow 2)$, because the termination of R does not ensure that of $R \cup DP(R)$ [11].

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

Example 3.6. *Let* $t \equiv f(e(e'(0,1),2), e''(3,4,5))$, $\pi(e) = 1$, $\pi(e') = []$ *and* $\pi(e'') = [1,3]$. *Then,* $\pi(t) = f(e', e''(3,5))$.

Theorem 3.7. *Let* $>$ *be a reduction order and* π *an argument filtering function. Then, the pair* $(\gtrsim_\pi, >_\pi)$ *is a weak reduction pair.*

Proof. We define the substitution $\pi(\theta)$ as $\pi(\theta)(x) = \pi(\theta(x))$. Then, the claim $\pi(t\theta) \equiv \pi(t)\pi(\theta)$ is easily proved by induction on t. Using this claim, the stability of $\gtrsim_\pi$ and $>_\pi$ is easily proved. The other conditions are routine. □

Definition 3.8. *We define the including relation* $\sqsubseteq$ *as follows:*

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

Theorem 3.9. *Let* R *be a TRS,* R' *a terminating TRS and* π *an argument filtering function. If* $\pi(R) \subseteq R'$ *and* $\pi(DP(R)) \sqsubseteq R'$ *then* R *is terminating.*

Proof. We define $>$ by $\xrightarrow[R']{+}$. The termination of R' ensure that $>$ is a reduction order. Using the argument filtering method, we design the weak reduction pair $(\gtrsim_\pi, >_\pi)$. It is obvious that $\forall l \to r \in R.\ l \gtrsim_\pi r$ and $\forall \langle u, v\rangle \in DP(R).\ u >_\pi v$. From theorem 3.4, R is terminating. □

Taking R and R' as a given TRS and a transformed TRS in elimination methods respectively, the above simple theorem can uniformly explain why elimination methods work well. This fact is very interesting because in the original literatures the soundness of these elimination methods were proved by complicated different methods. In the following sections, we will explain how theorem 3.9 simplifies the requirement conditions in elimination methods into acceptable one.

Corollary 3.10. *Let* R *be a TRS,* R' *a simply terminating TRS and* π *an argument filtering function. If* $\pi(R \cup DP(R)) \sqsubseteq R'$ *then* R *is terminating.*

Proof. As similar to theorem 3.9 by defining $>$ as $\xrightarrow[R' \cup Emb]{+}$. □

4 Argument Filtering Transformation

In this section, we design a new elimination method, called the argument filtering transformation. This transformation is designed based on theorem 3.9, which is essential for elimination methods.

Definition 4.1. *(Argument Filtering Transformation) Let* π *be an argument filtering function. The argument filtering transformation* (AFT_π) *is defined as follows:*

– $\begin{cases} dec_\pi(x) = \emptyset \\ dec_\pi(f(t_1,\ldots,t_n)) = \bigcup_{i\neq\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & if \ \ \pi(f)=i \\ dec_\pi(f(t_1,\ldots,t_n)) = \bigcup_{i\notin\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & otherwise \end{cases}$

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

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

Example 4.2. *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* $\pi(e) = []$, $\pi(e') = [1,3]$, $\pi(e'') = 2$ *and* $r \equiv f(e(e'(0,1,2),3), e''(f(4,5),6))$. *Then, we obtain* $AFT_\pi(R)$ *as follows (Fig.1):*

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

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order [5]. Thus,* R *is terminating, if the argument filtering transformation is sound. The soundness is showed in this section. Note that the termination of* R *is not easily proved, because* R *is not simply terminating.*

Lemma 4.3. *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. $\Box$

Theorem 4.4. *If* $AFT_\pi(R)$ *is terminating then* R *is 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 4.3, 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 $\overline{\pi}$, $root(\overline{\pi}(v))$ 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 $\pi(u) \to D[\pi(v)] \in \pi(R) \subseteq AFT_\pi(R)$.

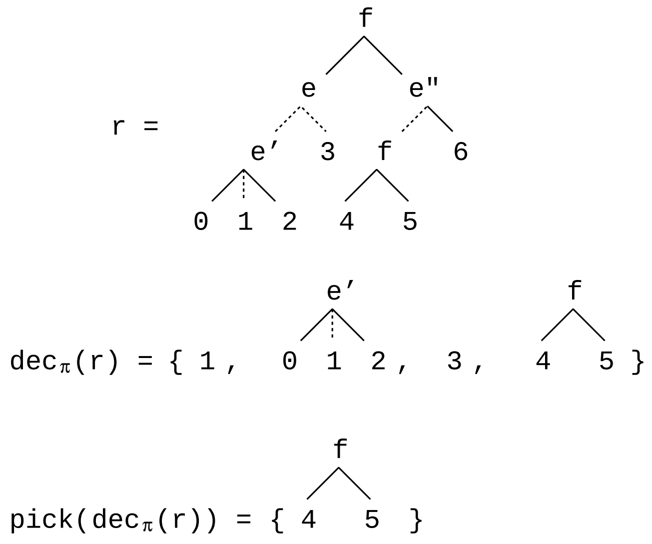

Fig. 1. Argument Filtering Transformation

From theorem 3.9, 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 4.5. *If $AFT_\pi^{\vec{C}_i}(R)$ is terminating then R is terminating.*

5 Comparison with Other Eliminations

For proving termination, several transformation methods, which simplify that task, have been proposed. As examples of such transformations the dummy elimination [6], the distribution elimination [13, 16], the general dummy elimination [7] and the improved general dummy elimination [15], were proposed. In this section, we compare these elimination methods to the argument filtering transformation. As a result, we conclude that the argument filtering transformation is a generalization of these elimination methods.

5.1 Dummy Elimination

Definition 5.1. *(Dummy Elimination)[6] Let e be a function symbol, called an eliminated symbol. The dummy elimination (DE_e) is defined as follows:*

- $\begin{cases} cap_e(x) = x \\ cap_e(e(t_1, \ldots, t_n)) = \diamond \\ cap_e(f(t_1, \ldots, t_n)) = f(cap_e(t_1), \ldots, cap_e(t_n)) \quad \mathit{if} \;\; f \neq e \end{cases}$
- $\begin{cases} dec_e(x) = \emptyset \\ dec_e(e(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} (\{cap_e(t_i)\} \cup dec_e(t_i)) \\ dec_e(f(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} dec_e(t_i) \quad \mathit{if} \;\; f \neq e \end{cases}$
- $DE_e(R) = \{cap_e(l) \to r' \mid l \to r \in R, \; r' \in \{cap_e(r)\} \cup dec_e(r)\}$

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

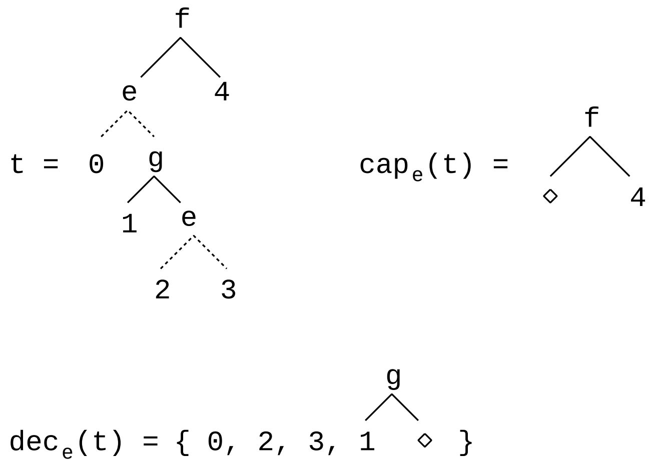

Fig. 2. Dummy Elimination

Proposition 5.3. *[6] If $DE_e(R)$ is terminating then R is terminating.*

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

Theorem 5.4. *For $\pi(e) = []$, $AFT_\pi(R) \subseteq DE_e(R)$.*

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 $t \equiv f(t_1,\dots,t_n)$. In the case $f \neq e$, $\pi(dec_\pi(f(t_1,\dots,t_n))) = \bigcup_{i=1}^n \pi(dec_\pi(t_i)) = \bigcup_{i=1}^n dec_e(t_i) = dec_e(f(t_1,\dots,t_n))$. In the case $f = e$, $\pi(dec_\pi(e(t_1,\dots,t_n))) = \bigcup_{i=1}^n (\{\pi(t_i)\} \cup \pi(dec_\pi(t_i))) = \bigcup_{i=1}^n (\{cap_e(t_i)\} \cup dec_e(t_i)) = dec_e(e(t_1,\dots,t_n))$. □

This theorem means that the argument filtering transformation is a proper extension of the dummy elimination, because $AFT_\pi(R)$ is terminating whenever $DE_e(R)$ is terminating.

5.2 Distribution Elimination

Definition 5.5. *(Distribution Elimination)[16]*
A rule $l \to r$ is a distribution rule for e if $l \equiv C[e(x_1,\dots,x_n)]$ and $r \equiv e(C[x_1],\dots,C[x_n])$ for some non-empty context C in which e does not occur and pairwise different variables $x_1,\dots,x_n$. Let e be an eliminated symbol. The distribution elimination (DIS_e) is defined as follows:

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

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

Example 5.6. *Let* $t \equiv f(e(0, g(1, e(2,3))), 4)$.
Then, $E_e(t) = \{f(0,4),\; f(g(1,2),4),\; f(g(1,3),4)\}$ *(Fig.3).*

In general, the distribution elimination is not sound with respect to termination, i.e., termination of $DIS_e(R)$ does not ensure termination of R. Thus, the distribution elimination requires suitable restrictions to ensure the soundness.

Proposition 5.7. *[13, 16]*

(a) *If $DIS_e(R)$ is terminating and right-linear then R is terminating.*
(b) *If $DIS_e(R)$ is terminating and non-constant symbol e does not occur in the left-hand sides of R then R is terminating.*

Lemma 5.8. *Let $\pi(e) = 1$. Under the condition (b), for any $l \to r \in AFT_\pi(R)$, there exists a context C such that $l \to C[r] \in DIS_e(R)$.*

Proof. 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'. □

Theorem 5.9. *Let $\pi(e) = 1$. Under the condition (b), the following properties hold.*

(i) $AFT_\pi^{\vec{C}_i}(R) \subseteq DIS_e(R)$ for some $\vec{C}_i$.

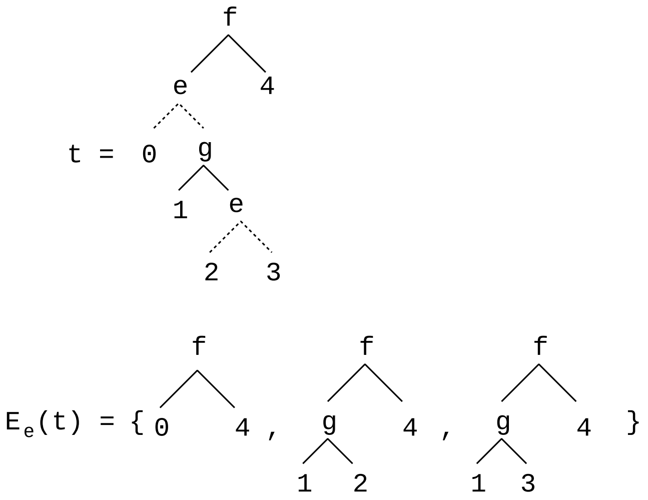

Fig. 3. Distribution Elimination

(ii) If $DIS_e(R)$ is simply terminating then $AFT_\pi(R)$ is simply terminating.

Proof. (i) It is trivial from lemma 5.8.

(ii) From the assumption and (i), $AFT_\pi^{\vec{C}_i}(R)$ is simply terminating. Thus $AFT_\pi(R)$ is simply terminating since $AFT_\pi(R) \sqsubseteq AFT_\pi^{\vec{C}_i}(R)$. $\square$

5.3 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 5.10. *(General Dummy Elimination)[7]*
Let e be an eliminated symbol and $\tau(e) = (I, i)$. The general dummy elimination (GDE_e) is defined as follows:

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

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

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

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

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

Example 5.11. *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* $dec(t) = \{5,\ 3\}$ *(Fig.4).*

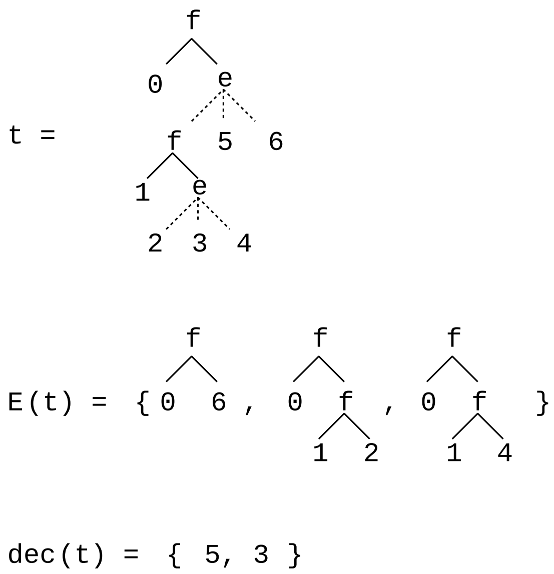

Fig. 4. General Dummy Elimination

Proposition 5.12. *[7] If* $GDE_e(R)$ *is terminating then* R *is terminating.*

Theorem 5.13. *Let* $\tau(e) = (I, i)$. *In the case* $\tau(e) = (\emptyset, 0)$, *we define* $\pi(e) = []$. *In the case* $\tau(e) = (I, i)$ *with* $i \in I$, *we define* $\pi(e) = i$. *Then, the following properties hold.*

(i) $AFT_{\pi}^{\vec{C}_i}(R) \subseteq GDE_e(R)$ *for some* $\vec{C}_i$.
(ii) If $GDE_e(R)$ *is simply terminating then* $AFT_{\pi}(R)$ *is simply terminating.*

Proof. (i) 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 $\tau(e) = (I, i)$ with $i \in I$, as similar to theorem 5.9 (i) by replacing $E_e(r)$ with $dec(r) \cup E(r)$. (ii) As similar to theorem 5.9 (ii). $\square$

We give the two following examples that the argument filtering transformation can be applied to, but the general dummy elimination can not be; the former is for showing that removing the unnecessary rules is effective and the latter is for showing that we can take well a defined function symbol as an eliminated symbol.

Example 5.14. *Consider TRS*

$$R = \begin{cases} f(a) \to f(b) \\ \quad b \to g(a) \end{cases}$$

Let $\pi(g) = []$. *Then,*

$$AFT_\pi(R) = \begin{cases} f(a) \to f(b) \\ \quad b \to \diamond \end{cases}$$

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order [5]. 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.*

$\tau(g)$	$GDE_g(R)$	
$(\emptyset, 0)$	$f(a) \to f(b)$ $b \to \diamond$ $b \to a$	$(= DE_g(R))$
$(\{1\}, 1)$	$f(a) \to f(b)$ $b \to a$	$(= DIS_g(R))$

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

Example 5.15. *Consider TRS*

$$R = \begin{cases} f(f(x)) \to f(g(f(x), x)) \\ g(x, y) \to y \end{cases}$$

Let $\pi(g) = [2]$. *Then,*

$$AFT_\pi(R) = \begin{cases} f(f(x)) \to f(g(x)) \\ f(f(x)) \to f(x) \\ \quad g(y) \to y \end{cases}$$

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order. We easily observe that the general dummy elimination can not be applied. Indeed, the* *following systems are clearly not terminating.*

$\tau(g)$	$GDE_g(R)$
$(\emptyset, 0)$	$f(f(x)) \to f(\diamond)$ $f(f(x)) \to f(x)$ $f(f(x)) \to x$ $\diamond \to y$
$(\{1\}, 1)$	$f(f(x)) \to f(x)$ $f(f(x)) \to x$ $x \to y$
$(\{2\}, 2)$	$f(f(x)) \to f(x)$ $y \to y$
$(\{1,2\}, 1)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $x \to y$
$(\{1,2\}, 2)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $y \to y$

The dummy elimination and the distribution elimination cannot be applied, too. Note that termination of R is not easily proved since R is not simply terminating.

5.4 Improved General Dummy Elimination

Definition 5.16. *(Improved General Dummy Elimination)[15]*
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 5.17. *[15] If $IGDE_e(R)$ is terminating then R is terminating.*

Theorem 5.18. *Let $\tau(e) = (I, i)$. In the case $\tau(e) = (\emptyset, 0)$, we define $\pi(e) = []$. In the case $\tau(e) = (I, i)$ with $i \in I$, we define $\pi(e) = i$. Then, the following properties hold.*

(i) $AFT_\pi^{\vec{C}_i}(R) \subseteq IGDE_e(R)$ for some $\vec{C}_i$.
(ii) If $IGDE_e(R)$ is simply terminating then $AFT_\pi(R)$ is simply terminating.

Proof. As similar to theorem 5.13 □

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

Example 5.19. *Consider TRS*

$$R = \begin{cases} f(f(x)) \to f(g(f(x), x)) \\ f(g(x, y)) \to f(y) \end{cases}$$

Let $\pi(g) = [2]$. *Then,*

$$AFT_\pi(R) = \{f(f(x)) \to f(g(x)),\ f(f(x)) \to f(x),\ f(g(y)) \to f(y)\}.$$

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order. We easily observe that the improved general dummy elimination can not be applied. Indeed, the following systems are clearly not terminating.*

$\tau(g)$	$IGDE_g(R)$
$(\emptyset, 0)$	$f(f(x)) \to f(\diamond)$ $f(f(x)) \to f(x)$ $f(\diamond) \to f(y)$
$(\{1\}, 1)$	$f(f(x)) \to f(x)$ $f(x) \to f(y)$
$(\{2\}, 2)$	$f(f(x)) \to f(x)$ $f(y) \to f(y)$
$(\{1, 2\}, 1)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $f(x) \to f(y)$
$(\{1, 2\}, 2)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $f(y) \to f(y)$

The dummy elimination, the distribution elimination and the general dummy elimination cannot be applied, too. Note that termination of R *is not easily proved since* R *is not simply terminating.*

Acknowledgments

We would like to thank Jürgen Giesl for introducing us the papers [4, 8] and discussion on this work, Masahiko Sakai and the referees for their useful comments.

This work is partially supported by Grants from Ministry of Education, Science and Culture of Japan, #10139214 and #10680346.

References

1. T.Arts, Automatically Proving Termination and Innermost Normalization of Term Rewriting Systems, PhD thesis, Utrecht University, 1997.
2. T.Arts, J.Giesl, Automatically Proving Termination where Simplification Orderings Fail, LNCS 1214 (TAPSOFT 97), 261-272, 1997.
3. T.Arts, J.Giesl, Proving Innermost Normalization Automatically, LNCS 1232 (RTA 97), 157-171, 1997.

4. T.Arts, J.Giesl, Termination of Term Rewriting Using Dependency Pairs, *to appear in Theoretical Computer Science.*
5. F.Baader, T.Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
6. M.Ferreira, H.Zantema, Dummy Elimination: Making Termination Easier, LNCS 965 (FCT 95), 243-252, 1995.
7. M.Ferreira, Termination of Term Rewriting, Well-foundedness, Totality and Transformations, PhD thesis, Utrecht University, 1995.
8. J.Giesl, E.Ohlebusch, Pushing the Frontiers of Combining Rewrite Systems Farther Outwards, In Proceedings of the Second International Workshop on Frontiers of Combining Systems, FroCos '98, Applied Logic Series, Amsterdam, The Netherlands, October 1998.
9. J.W.Klop, Term Rewriting Systems, Handbook of Logic in Computer Science II, 1-112, Oxford University Press, 1992.
10. K.Kusakari, Y.Toyama, On Proving AC-Termination by AC-Dependency Pairs, Research Report IS-RR-98-0026F, School of Information Science, JAIST, 1998.
11. K.Kusakari, Y.Toyama, The Hierarchy of Dependency Pairs, Research Report IS-RR-99-0007F, School of Information Science, JAIST, 1999.
12. K.Kusakari, M.Nakamura, Y.Toyama, Argument Filtering Transformation, Research Report IS-RR-99-0008F, School of Information Science, JAIST, 1999.
13. A.Middeldorp, H.Ohsaki, H.Zantema, Transforming Termination by Self-Labelling, LNCS 1104 (CADE-13), 373-387, 1996.
14. C.Marche, X.Urbain, Termination of Associative-Commutative Rewriting by Dependency Pairs, LNCS 1379 (RTA 98), 241-255, 1998.
15. M.Nakamura, K.Kusakari, Y.Toyama, On Proving Termination by General Dummy Elimination, to appear in Trans. of IEICE (in Japanese).
16. H.Zantema, Termination of Term Rewriting: Interpretation and Type Elimination, Journal of Symbolic Computation 17, 23-50, 1994.

Argument Filtering Transformation

Keiichirou Kusakari Masaki Nakamura Yoshihito Toyama

School of Information Science, JAIST
Tatsunokuchi, Ishikawa 923-1292, Japan
email : {kusakari, masaki-n, toyama}@jaist.ac.jp

Abstract. To simplify the task of proving termination of term rewriting systems, several elimination methods, such as the dummy elimination, the distribution elimination, the general dummy elimination and the improved general dummy elimination, have been proposed. In this paper, we show that the argument filtering method combining with the dependency pair technique is essential in all the above elimination methods. We present remarkable simple proofs for the soundness of these elimination methods based on this observation. Moreover, we propose a new elimination method, called the argument filtering transformation, which is not only more powerful than all the other elimination methods but also especially useful to make clear the essential relation hidden behind these methods.

Keywords: Term Rewriting System, Termination, Elimination Method, Dependency Pair, Argument Filtering

1 Introduction

Term Rewriting Systems (TRSs) can be regarded as a model for computation in which terms are reduced, using a set of directed equations. They are used 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 [5, 9].

Termination is one of the most fundamental properties of term rewriting systems. While in general termination of TRSs is an undecidable property, several methods for proving termination have been developed. To simplify the task of proving termination of TRSs to which these methods cannot be directly applied, several elimination methods have been proposed. Elimination methods try to transform a given TRS into a TRS whose termination is easier to prove than the original one. The dummy elimination [6], the distribution elimination [13, 16], the general dummy elimination [7] and the improved general dummy elimination [15] are examples of elimination methods.

Recently, Arts and Giesl proposed the notion of the dependency pair, which can offer an effective method for analyzing an infinite reduction sequence [1–3]. Using dependency pairs, we can easily show the termination property of TRSs to which traditional techniques cannot be applied. Since this method compares rewrite rules and dependency pairs by a weak reduction pair instead of a reduction order, to find an appropriate weak reduction pair for a given TRS is necessary. The argument filtering method introduced [4, 8] allows us to make a weak reduction pair from an arbitrary reduction order.

In this paper, we first extend the argument filtering method by combining subterm relation. Next, we study the relation between the argument filtering method and various elimination methods. The key of our result is the observation that the argument filtering method combining with the dependency pair technique is essential in all the above elimination methods. Indeed, we present remarkable simple proofs for the soundness of these elimination methods based on this observation, though the original proofs presented in the literatures [6, 7, 13, 15, 16] are complicated and treated as rather different methods respectively. This observation also leads us to a new powerful elimination method, called the argument filtering transformation, which is not only more powerful than all the other elimination methods but also especially useful to make clear the essential relation hidden behind these methods. The main contributions of this paper are as follows:

(1) We show that the argument filtering method combining with the dependency pair technique can clearly explain in a uniform framework why various elimination methods work well.
(2) 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 these elimination methods.
(3) 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.

The remainder of this paper is organized as follows. The next section gives the definition of term rewriting systems. In section 3, we explain the dependency pair technique and introduce a new argument filtering method. Using these results, we show a general and essential property for elimination methods to be sound with respect to termination. In section 4, we propose the argument filtering transformation and show the soundness of this transformation. In section 5, we compare various elimination methods with the argument filtering transformation, and give simple proofs for the soundness of these elimination methods.

2 Preliminaries

We assume that the reader is familiar with notions of term rewriting systems [5].

A signature Σ is a finite set of function symbols, where each $f \in \Sigma$ is associated with a non-negative integer 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 constructed from Σ and $\mathcal{V}$ is written by $\mathcal{T}(\Sigma, \mathcal{V})$. $Var(t)$ is the set of variables in t. We define $root(f(t_1, \ldots, t_n)) = f$. Identity of terms is denoted by $\equiv$. A substitution $\theta : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is a mapping. A substitution over terms is defined as a homomorphic extension. We write $t\theta$ instead of $\theta(t)$. A context is a term which has a special constant $\Box$, called a hole. A term $C[t]$ denotes the result of replacing t in the hole of C.

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 set of rules. The set of defined symbols in a TRS R is denoted by $DF(R) = \{root(l) \mid l \to r \in R\}$. A reduction relation $\underset{R}{\to}$ is defined as follows: $s \underset{R}{\to} t \overset{\mathrm{def}}{\iff} \exists\, l \to r \in R, \exists C[\;], \exists\theta\ (s \equiv C[l\theta] \wedge t \equiv C[r\theta])$. We often omit the subscript R whenever no confusion arises. A TRS R is terminating if there is no infinite sequence such that $t_0 \underset{R}{\to} t_1 \underset{R}{\to} t_2 \underset{R}{\to} \cdots$. The transitive-reflexive closure and the transitive closure of $\to$ are denoted by $\overset{*}{\to}$ and $\overset{+}{\to}$, respectively.

A strict order $>$ is a reduction order if $>$ is well-founded, monotonic ($s > t \Rightarrow C[s] > C[t]$) and stable ($s > t \Rightarrow s\theta > t\theta$). Note that a TRS R is terminating iff there exists a reduction order $>$ that satisfies $l > r$ for all $l \to r \in R$. A reduction order $>$ is a simplification order if $C[t] > t$ for all t and C ($\not\equiv \Box$). A TRS R is simply terminating if there exists a simplification order $>$ that satisfies $l > r$ for all $l \to r \in R$.

3 Soundness Condition for Transformation

In this section, we first explain the dependency pair and the argument filtering method, whose notions greatly extend the provable class of termination. Using these notions, we show a theorem, which makes a general and essential property clear for transformations of TRSs to be sound with respect to termination.

Definition 3.1. *[1–3] $\Sigma^\# = \{f^\# \mid f \in \Sigma\}$ is the set of marked symbols disjoint from $\Sigma \cup \mathcal{V}$. We define the root-marked terms by $(f(t_1, \ldots, t_n))^\# = f^\#(t_1, \ldots, t_n)$. The set of the dependency pairs of R, written by $DP^\#(R)$, is $\{\langle u^\#, v^\# \rangle \mid u \to C[v] \in R,\ root(v) \in DF(R)\}$. The set of the unmarked dependency pairs of R, written by $DP(R)$, is obtained by erasing marks of symbols in $DP^\#(R)$.*

Example 3.2. *Let $R = \{add(x, 0) \to x,\ add(x, s(y)) \to s(add(x, y))\}$. Then,*
$DP^\#(R) = \{\langle add^\#(x, s(y)), add^\#(x, y)\rangle\}$. and
$DP(R) = \{\langle add(x, s(y)), add(x, y)\rangle\}$

Definition 3.3. *A pair $(\gtrsim, >)$ of binary relations on terms is a weak reduction pair if it satisfies the following three conditions:*

- $\gtrsim$ is monotonic and stable

– $>$ is stable and well-founded
– $\gtrsim \cdot > \;\subseteq\; >$ or $> \cdot \gtrsim \;\subseteq\; >$

In the above definition, we do not assume that $\gtrsim$ is a quasi-order (reflexive and transitive) or $>$ is a strict order (irreflexive and transitive). 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 the 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.

Theorem 3.4. *For any TRS R, the following three properties are equivalent.*

1. TRS R is terminating.
2. There exists a weak reduction pair $(\gtrsim, >)$ such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u, v\rangle \in DP(R).\ u > v$.
3. There exists a weak reduction pair $(\gtrsim, >)$ such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u^\#, v^\#\rangle \in DP^\#(R).\ u^\# > v^\#$.

Proof. $(1 \Rightarrow 2)$ We define $s \gtrsim t$ by $s \xrightarrow[R]{*} t$, and $s > t$ by $s \xrightarrow[R]{+} C[t]$ for some C. Then, it is trivial that $(\gtrsim, >)$ is a weak reduction pair such that $\forall l \to r \in R.\ l \gtrsim r$ and $\forall \langle u, v\rangle \in DP(R).\ u > v$. $(2 \Rightarrow 3)$ It is easily shown by identifying $f^\#$ with f. $(3 \Rightarrow 1)$ This case has already been shown in [1–4, 8]. $\square$

Note that the proofs for $(2 \Rightarrow 1)$ and for $(1 \Rightarrow 3)$ have already been shown in [10, 14] and [4] [1], respectively.

The above theorem shows that the weak reduction pair plays an important role. To design a weak reduction pair, the argument filtering method introduced in [4, 8] is very useful, which is defined as recursive program schemata [9]. In the next definition, we introduce a new argument filtering method by combining the subterm relation, which is more effective in our framework than original one.

Definition 3.5. *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). We can naturally extend π over terms as follows:*

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

For any argument filtering function π and any reduction order $>$, we define the pair $(\gtrsim_\pi, >_\pi)$ as follows:

$$s \gtrsim_\pi t \overset{\text{def}}{\iff} \pi(s) > \pi(t) \text{ or } \pi(s) \equiv \pi(t)$$
$$s >_\pi t \overset{\text{def}}{\iff} \exists C.\ \pi(s) > C[\pi(t)] \text{ or } \exists C \not\equiv \square.\ \pi(s) \equiv C[\pi(t)]$$

[1] The proof for $(1 \Rightarrow 3)$ in [4] is based on the claim that if R is terminating then so is $R \cup DP^\#(R)$. However, the same proof method can not work well for $(1 \Rightarrow 2)$, because the termination of R does not ensure that of $R \cup DP(R)$ [11].

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

Example 3.6. *Let* $t \equiv f(e(e'(0,1),2), e''(3,4,5))$*,* $\pi(e) = 1$*,* $\pi(e') = []$ *and* $\pi(e'') = [1,3]$*. Then,* $\pi(t) = f(e', e''(3,5))$*.*

Theorem 3.7. *Let* $>$ *be a reduction order and* π *an argument filtering function. Then, the pair* $(\gtrsim_\pi, >_\pi)$ *is a weak reduction pair.*

Proof. We define the substitution $\pi(\theta)$ as $\pi(\theta)(x) = \pi(\theta(x))$. Then, the claim $\pi(t\theta) \equiv \pi(t)\pi(\theta)$ is easily proved by induction on t. Using this claim, the stability of $\gtrsim_\pi$ and $>_\pi$ is easily proved. The other conditions are routine. □

Definition 3.8. *We define the including relation* $\sqsubseteq$ *as follows:*

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

Theorem 3.9. *Let* R *be a TRS,* R' *a terminating TRS and* π *an argument filtering function. If* $\pi(R) \subseteq R'$ *and* $\pi(DP(R)) \sqsubseteq R'$ *then* R *is terminating.*

Proof. We define $>$ by $\xrightarrow[R']{+}$. The termination of R' ensure that $>$ is a reduction order. Using the argument filtering method, we design the weak reduction pair $(\gtrsim_\pi, >_\pi)$. It is obvious that $\forall l \to r \in R.\ l \gtrsim_\pi r$ and $\forall \langle u, v\rangle \in DP(R).\ u >_\pi v$. From theorem 3.4, R is terminating. □

Taking R and R' as a given TRS and a transformed TRS in elimination methods respectively, the above simple theorem can uniformly explain why elimination methods work well. This fact is very interesting because in the original literatures the soundness of these elimination methods were proved by complicated different methods. In the following sections, we will explain how theorem 3.9 simplifies the requirement conditions in elimination methods into acceptable one.

Corollary 3.10. *Let* R *be a TRS,* R' *a simply terminating TRS and* π *an argument filtering function. If* $\pi(R \cup DP(R)) \sqsubseteq R'$ *then* R *is terminating.*

Proof. As similar to theorem 3.9 by defining $>$ as $\xrightarrow[R' \cup Emb]{+}$. □

4 Argument Filtering Transformation

In this section, we design a new elimination method, called the argument filtering transformation. This transformation is designed based on theorem 3.9, which is essential for elimination methods.

Definition 4.1. *(Argument Filtering Transformation) Let* π *be an argument filtering function. The argument filtering transformation (*AFT_π*) is defined as follows:*

$$- \begin{cases} dec_\pi(x) = \emptyset \\ dec_\pi(f(t_1,\ldots,t_n)) = \bigcup_{i\neq\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & \text{if } \pi(f) = i \\ dec_\pi(f(t_1,\ldots,t_n)) = \bigcup_{i\notin\pi(f)}\{t_i\} \ \cup \ \bigcup_{i=1}^n dec_\pi(t_i) & \text{otherwise} \end{cases}$$

– $pick_\pi(T) = \{t \in T \mid \overline{\pi}(t)$ *includes some defined symbols of* $R\}$

$$\text{where} \quad \overline{\pi}(f) = \begin{cases} [i] & \text{if } \pi(f) = i \\ \pi(f) & \text{otherwise} \end{cases}$$

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

Example 4.2. *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* $\pi(e) = []$, $\pi(e') = [1,3]$, $\pi(e'') = 2$ *and* $r \equiv f(e(e'(0,1,2),3), e''(f(4,5),6))$. *Then, we obtain* $AFT_\pi(R)$ *as follows (Fig.1):*

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

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order [5]. Thus,* R *is terminating, if the argument filtering transformation is sound. The soundness is showed in this section. Note that the termination of* R *is not easily proved, because* R *is not simply terminating.*

Lemma 4.3. *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. $\Box$

Theorem 4.4. *If* $AFT_\pi(R)$ *is terminating then* R *is 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 4.3, 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 $\overline{\pi}$, $root(\overline{\pi}(v))$ 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 $\pi(u) \to D[\pi(v)] \in \pi(R) \subseteq AFT_\pi(R)$.

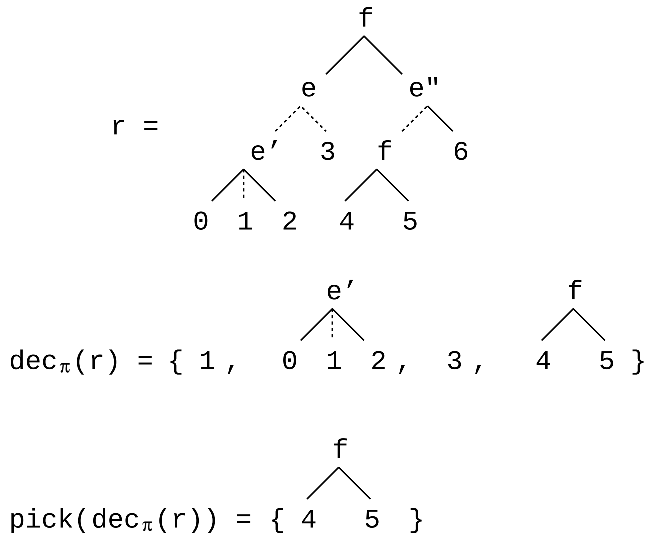

Fig. 1. Argument Filtering Transformation

From theorem 3.9, R is terminating. $\square$

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 4.5. *If $AFT_\pi^{\vec{C}_i}(R)$ is terminating then R is terminating.*

5 Comparison with Other Eliminations

For proving termination, several transformation methods, which simplify that task, have been proposed. As examples of such transformations the dummy elimination [6], the distribution elimination [13, 16], the general dummy elimination [7] and the improved general dummy elimination [15], were proposed. In this section, we compare these elimination methods to the argument filtering transformation. As a result, we conclude that the argument filtering transformation is a generalization of these elimination methods.

5.1 Dummy Elimination

Definition 5.1. *(Dummy Elimination)[6] Let e be a function symbol, called an eliminated symbol. The dummy elimination (DE_e) is defined as follows:*

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

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

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

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

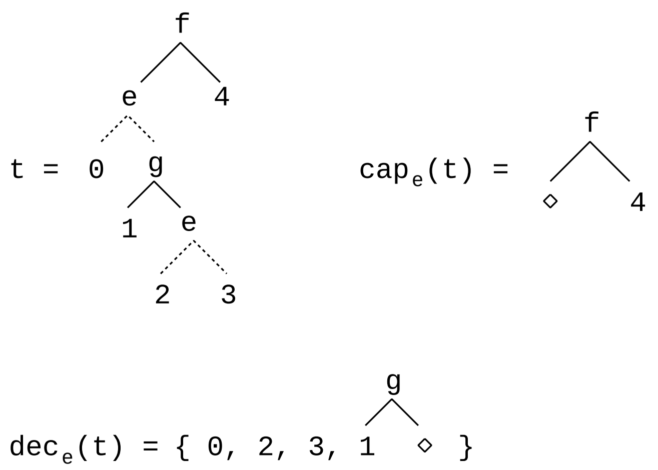

Fig. 2. Dummy Elimination

Proposition 5.3. *[6] If $DE_e(R)$ is terminating then R is terminating.*

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

Theorem 5.4. *For $\pi(e) = []$, $AFT_\pi(R) \subseteq DE_e(R)$.*

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 $f \neq e$, $\pi(dec_\pi(f(t_1,\ldots,t_n))) = \bigcup_{i=1}^n \pi(dec_\pi(t_i)) = \bigcup_{i=1}^n dec_e(t_i) = dec_e(f(t_1,\ldots,t_n))$. In the case $f = e$, $\pi(dec_\pi(e(t_1,\ldots,t_n))) = \bigcup_{i=1}^n (\{\pi(t_i)\} \cup \pi(dec_\pi(t_i))) = \bigcup_{i=1}^n (\{cap_e(t_i)\} \cup dec_e(t_i)) = dec_e(e(t_1,\ldots,t_n))$. □

This theorem means that the argument filtering transformation is a proper extension of the dummy elimination, because $AFT_\pi(R)$ is terminating whenever $DE_e(R)$ is terminating.

5.2 Distribution Elimination

Definition 5.5. *(Distribution Elimination)[16]*
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:

- $E_e(t) = \begin{cases} \{t\} & \textit{if } t \in \mathcal{V} \\ \bigcup_{i=1}^n E_e(t_i) & \textit{if } t \equiv e(t_1,\ldots,t_n) \\ \{f(s_1,\ldots,s_n) \mid s_i \in E_e(t_i)\} & \textit{if } t \equiv f(t_1,\ldots,t_n) \ \textit{ with } f \neq e \end{cases}$
- $DIS_e(R) = \{l \to r' \mid l \to r \in R \textit{ is not a distribution rule for } e,\ r' \in E_e(r)\}$

Example 5.6. *Let $t \equiv f(e(0,g(1,e(2,3))),4)$.*
Then, $E_e(t) = \{f(0,4),\ f(g(1,2),4),\ f(g(1,3),4)\}$ (Fig.3).

In general, the distribution elimination is not sound with respect to termination, i.e., termination of $DIS_e(R)$ does not ensure termination of R. Thus, the distribution elimination requires suitable restrictions to ensure the soundness.

Proposition 5.7. *[13, 16]*

(a) *If $DIS_e(R)$ is terminating and right-linear then R is terminating.*
(b) *If $DIS_e(R)$ is terminating and non-constant symbol e does not occur in the left-hand sides of R then R is terminating.*

Lemma 5.8. *Let $\pi(e) = 1$. Under the condition (b), for any $l \to r \in AFT_\pi(R)$, there exists a context C such that $l \to C[r] \in DIS_e(R)$.*

Proof. 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'. □

Theorem 5.9. *Let $\pi(e) = 1$. Under the condition (b), the following properties hold.*

(i) $AFT_\pi^{\vec{C}_i}(R) \subseteq DIS_e(R)$ for some $\vec{C}_i$.

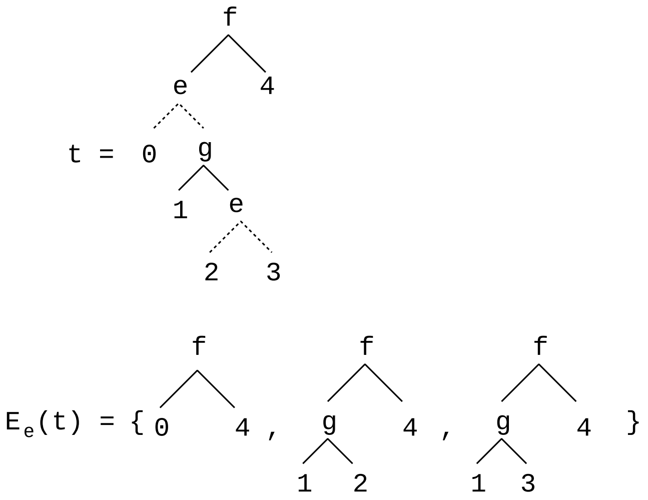

Fig. 3. Distribution Elimination

(ii) If $DIS_e(R)$ is simply terminating then $AFT_\pi(R)$ is simply terminating.

Proof. (i) It is trivial from lemma 5.8.
(ii) From the assumption and (i), $AFT_\pi^{\vec{C}_i}(R)$ is simply terminating. Thus $AFT_\pi(R)$ is simply terminating since $AFT_\pi(R) \sqsubseteq AFT_\pi^{\vec{C}_i}(R)$. $\square$

5.3 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 5.10. *(General Dummy Elimination)[7]*
Let e be an eliminated symbol and $\tau(e) = (I, i)$. The general dummy elimination (GDE_e) is defined as follows:

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

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

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

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

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

Example 5.11. *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 $dec(t) = \{5,\ 3\}$ (Fig.4).*

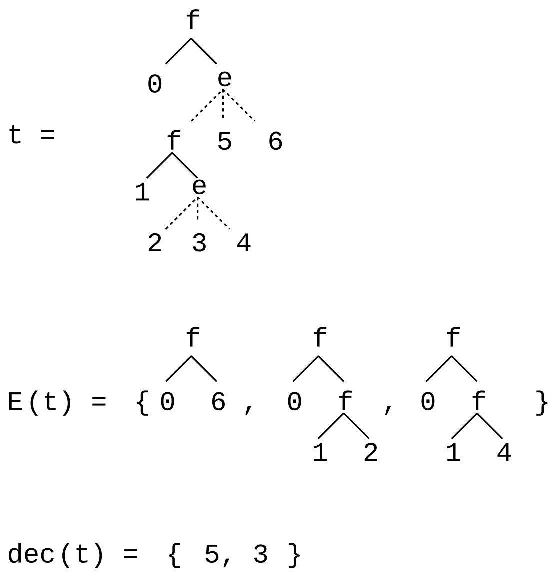

Fig. 4. General Dummy Elimination

Proposition 5.12. *[7] If $GDE_e(R)$ is terminating then R is terminating.*

Theorem 5.13. *Let $\tau(e) = (I, i)$. In the case $\tau(e) = (\emptyset, 0)$, we define $\pi(e) = []$. In the case $\tau(e) = (I, i)$ with $i \in I$, we define $\pi(e) = i$. Then, the following properties hold.*

(i) $AFT_{\pi}^{\vec{C}_i}(R) \subseteq GDE_e(R)$ for some $\vec{C}_i$.
(ii) If $GDE_e(R)$ is simply terminating then $AFT_{\pi}(R)$ is simply terminating.

Proof. (i) 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 $\tau(e) = (I, i)$ with $i \in I$, as similar to theorem 5.9 (i) by replacing $E_e(r)$ with $dec(r) \cup E(r)$. (ii) As similar to theorem 5.9 (ii). $\square$

We give the two following examples that the argument filtering transformation can be applied to, but the general dummy elimination can not be; the former is for showing that removing the unnecessary rules is effective and the latter is for showing that we can take well a defined function symbol as an eliminated symbol.

Example 5.14. *Consider TRS*

$$R = \begin{cases} f(a) \to f(b) \\ b \to g(a) \end{cases}$$

Let $\pi(g) = []$. Then,

$$AFT_\pi(R) = \begin{cases} f(a) \to f(b) \\ b \to \diamond \end{cases}$$

The termination of $AFT_\pi(R)$ is easily proved by recursive path order [5]. 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.

$\tau(g)$	$GDE_g(R)$	
$(\emptyset, 0)$	$f(a) \to f(b)$ $b \to \diamond$ $b \to a$	$(= DE_g(R))$
$(\{1\}, 1)$	$f(a) \to f(b)$ $b \to a$	$(= DIS_g(R))$

Note that termination of R is not easily proved since R is not simply terminating.

Example 5.15. *Consider TRS*

$$R = \begin{cases} f(f(x)) \to f(g(f(x), x)) \\ g(x, y) \to y \end{cases}$$

Let $\pi(g) = [2]$. Then,

$$AFT_\pi(R) = \begin{cases} f(f(x)) \to f(g(x)) \\ f(f(x)) \to f(x) \\ g(y) \to y \end{cases}$$

The termination of $AFT_\pi(R)$ is easily proved by recursive path order. We easily observe that the general dummy elimination can not be applied. Indeed, the *following systems are clearly not terminating.*

$\tau(g)$	$GDE_g(R)$
$(\emptyset, 0)$	$f(f(x)) \to f(\diamond)$ $f(f(x)) \to f(x)$ $f(f(x)) \to x$ $\diamond \to y$
$(\{1\}, 1)$	$f(f(x)) \to f(x)$ $f(f(x)) \to x$ $x \to y$
$(\{2\}, 2)$	$f(f(x)) \to f(x)$ $y \to y$
$(\{1,2\}, 1)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $x \to y$
$(\{1,2\}, 2)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $y \to y$

The dummy elimination and the distribution elimination cannot be applied, too. Note that termination of R is not easily proved since R is not simply terminating.

5.4 Improved General Dummy Elimination

Definition 5.16. *(Improved General Dummy Elimination)[15]*
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 5.17. *[15] If $IGDE_e(R)$ is terminating then R is terminating.*

Theorem 5.18. *Let $\tau(e) = (I, i)$. In the case $\tau(e) = (\emptyset, 0)$, we define $\pi(e) = []$. In the case $\tau(e) = (I, i)$ with $i \in I$, we define $\pi(e) = i$. Then, the following properties hold.*

(i) $AFT_\pi^{\vec{C}_i}(R) \subseteq IGDE_e(R)$ for some $\vec{C}_i$.
(ii) If $IGDE_e(R)$ is simply terminating then $AFT_\pi(R)$ is simply terminating.

Proof. As similar to theorem 5.13 □

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

Example 5.19. *Consider TRS*

$$R = \begin{cases} f(f(x)) \to f(g(f(x), x)) \\ f(g(x, y)) \to f(y) \end{cases}$$

Let $\pi(g) = [2]$. *Then,*

$$AFT_\pi(R) = \{f(f(x)) \to f(g(x)),\ f(f(x)) \to f(x),\ f(g(y)) \to f(y)\}.$$

The termination of $AFT_\pi(R)$ *is easily proved by recursive path order. We easily observe that the improved general dummy elimination can not be applied. Indeed, the following systems are clearly not terminating.*

$\tau(g)$	$IGDE_g(R)$
$(\emptyset, 0)$	$f(f(x)) \to f(\diamond)$ $f(f(x)) \to f(x)$ $f(\diamond) \to f(y)$
$(\{1\}, 1)$	$f(f(x)) \to f(x)$ $f(x) \to f(y)$
$(\{2\}, 2)$	$f(f(x)) \to f(x)$ $f(y) \to f(y)$
$(\{1,2\}, 1)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $f(x) \to f(y)$
$(\{1,2\}, 2)$	$f(f(x)) \to f(f(x))$ $f(f(x)) \to f(x)$ $f(y) \to f(y)$

The dummy elimination, the distribution elimination and the general dummy elimination cannot be applied, too. Note that termination of R *is not easily proved since* R *is not simply terminating.*

Acknowledgments

We would like to thank Jürgen Giesl for introducing us the papers [4, 8] and discussion on this work, Masahiko Sakai and the referees for their useful comments.

This work is partially supported by Grants from Ministry of Education, Science and Culture of Japan, #10139214 and #10680346.

References

1. T.Arts, Automatically Proving Termination and Innermost Normalization of Term Rewriting Systems, PhD thesis, Utrecht University, 1997.
2. T.Arts, J.Giesl, Automatically Proving Termination where Simplification Orderings Fail, LNCS 1214 (TAPSOFT 97), 261-272, 1997.
3. T.Arts, J.Giesl, Proving Innermost Normalization Automatically, LNCS 1232 (RTA 97), 157-171, 1997.

4. T.Arts, J.Giesl, Termination of Term Rewriting Using Dependency Pairs, *to appear in Theoretical Computer Science.*
5. F.Baader, T.Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
6. M.Ferreira, H.Zantema, Dummy Elimination: Making Termination Easier, LNCS 965 (FCT 95), 243-252, 1995.
7. M.Ferreira, Termination of Term Rewriting, Well-foundedness, Totality and Transformations, PhD thesis, Utrecht University, 1995.
8. J.Giesl, E.Ohlebusch, Pushing the Frontiers of Combining Rewrite Systems Farther Outwards, In Proceedings of the Second International Workshop on Frontiers of Combining Systems, FroCos '98, Applied Logic Series, Amsterdam, The Netherlands, October 1998.
9. J.W.Klop, Term Rewriting Systems, Handbook of Logic in Computer Science II, 1-112, Oxford University Press, 1992.
10. K.Kusakari, Y.Toyama, On Proving AC-Termination by AC-Dependency Pairs, Research Report IS-RR-98-0026F, School of Information Science, JAIST, 1998.
11. K.Kusakari, Y.Toyama, The Hierarchy of Dependency Pairs, Research Report IS-RR-99-0007F, School of Information Science, JAIST, 1999.
12. K.Kusakari, M.Nakamura, Y.Toyama, Argument Filtering Transformation, Research Report IS-RR-99-0008F, School of Information Science, JAIST, 1999.
13. A.Middeldorp, H.Ohsaki, H.Zantema, Transforming Termination by Self-Labelling, LNCS 1104 (CADE-13), 373-387, 1996.
14. C.Marche, X.Urbain, Termination of Associative-Commutative Rewriting by Dependency Pairs, LNCS 1379 (RTA 98), 241-255, 1998.
15. M.Nakamura, K.Kusakari, Y.Toyama, On Proving Termination by General Dummy Elimination, to appear in Trans. of IEICE (in Japanese).
16. H.Zantema, Termination of Term Rewriting: Interpretation and Type Elimination, Journal of Symbolic Computation 17, 23-50, 1994.