社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

指定した引数を固定した逆関数を定義するTRSの生成

西田　直樹†　　酒井　正彦†　　坂部　俊樹†

† 名古屋大学大学院工学研究科
〒 464–8603 名古屋市千種区不老町

E-mail: †nishida@sakabe.nuie.nagoya-u.ac.jp, ††{sakai,sakabe}@nuie.nagoya-u.ac.jp

あらまし　本稿では，指定した引数を固定した逆関数を定義するTRSを生成するアルゴリズムを提案する．最初に，constructor TRSの逆TRSを生成するアルゴリズム $\mathcal{InvCTRS}_0$ を提案し，次に $\mathcal{InvCTRS}_0$ を指定した引数を固定した関数を対象としたアルゴリズム $\mathcal{InvCTRS}_I$ に拡張する．最後に，Ohlebuschの変換を利用し，生成したCTRSをTRSに変換できることを示す．

キーワード　逆関数，プログラム変換，関数型プログラム，TRS

Generation of a TRS Implementing the Inverses of the Functions with Specified Arguments Fixed

Naoki NISHIDA†, Masahiko SAKAI†, and Toshiki SAKABE†

† Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464–8603 Japan

E-mail: †nishida@sakabe.nuie.nagoya-u.ac.jp, ††{sakai,sakabe}@nuie.nagoya-u.ac.jp

Abstract　In this paper, we propose an algorithm for generating a TRS implementing the inverses of the functions with specified arguments fixed. At first, we propose an algorithm $\mathcal{InvCTRS}_0$ for generating inverse CTRSs for constructor TRSs. Next, we extend $\mathcal{InvCTRS}_0$ to the algorithm $\mathcal{InvCTRS}_I$ for the functions with specified arguments fixed. Finally, we show that our inverse CTRSs can be transformed to TRSs by using Ohlebusch's transformation.

Key words　inverse function, program transformation, functional program, TRS

1 Introduction

In some cases, we need value of some arguments, giving value for the rest of arguments, i.e., consider the following TRS:

$$\begin{cases} 0+y \to y \\ s(x)+y \to s(x+y) \\ gcd(x+y,y) \to gcd(x,y) \\ gcd(x,0) \to x \\ gcd(x,y) \to gcd(y,x) \Leftarrow x<y \end{cases}$$

It is difficult that some value $n\ (=x+y)$ is devided to x and y in standerd calculation. But it is easy to devide n to x and y by using inverse functions. And in this case, it is more easy to calculate the inverse function of $+$ since y is already given. If we already have the inverse function $-$ of $+$ which takes n and y as input and outputs x as follows, it is very convenient.

$$\begin{cases} gcd(x,y) \to gcd(z,y) \Leftarrow x-y \to z \\ gcd(x,0) \to x \\ gcd(x,y) \to gcd(y,x) \Leftarrow x<y \\ \quad\vdots \end{cases}$$

As other case, there is a problem to solve equations in ground convergent left-linear rewrite systems as like the follows[Der99].

$$n(d,m,y) = 360\times y + 30\times m + d \ (\ 0\le d<30,\ 0\le m<12\ )$$

As general method for solving equations, we use "Narrowing"[BN98]. But simple Narrowing is not terminating. If inverse functions are terminating, a method using them is useful since searching space is finite.

We have proposed an algorithm for generating inverse TRSs for pure treeless TRSs[NN01]. Since pure treeless TRSs are quite restricted class, it is hard to say that our algorithm for pure treeless TRSs is useful. To treat the larger classes than pure treeless TRSs, we need to extend our algorithm to one for constructor TRSs.

In this paper, we propose an algorithm for generating a TRS implementing the inverses of the functions with specified arguments fixed. At first, we extend the algorithm for pure treeless TRSs[NN01] to the algorithm $\mathcal{I}nv\mathcal{CTRS}_0$ for generating inverse CTRSs for constructor TRSs. Next, we extend $\mathcal{I}nv\mathcal{CTRS}_0$ to the algorithm $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ for the functions with specified arguments fixed. Finally, we show that our inverse CTRSs can be transformed to TRSs by using Ohlebusch's transformation.

2 Preparation

In this paper, we mainly follow the notation of [BN98,Klop92]. A *reduction system* is a structure $\mathcal{A}=(\mathcal{S},\to)$ where $\mathcal{S}$ is a set and $\to$ is a binary relation over $\mathcal{S}$, which is called a reduction relation. A *reduction* is a finite sequence $x_0\to x_1\to\cdots\to x_n$ $(n\ge 0)$ or an infinite sequence $x_0\to x_1\to\cdots$ of reduction steps. The identity of elements $x,y\in\mathcal{S}$ is denoted by $x\equiv y$. If there is no element y such that $x\to y$, then we say x is a *normal form* (with respect to $\mathcal{A}$). $\overset{*}{\to}$ is the reflexive and transitive closure of $\to$. $\overset{k}{\to}$ and $\overset{k\ge}{\to}$ denote a reduction of k steps and a reduction of steps less than k, respectively. $\mathcal{A}$ is *confluent* if and only if $x\overset{*}{\to}y\wedge x\overset{*}{\to}y'$ imply $\exists z\in\mathcal{S}$, $y\overset{*}{\to}z\wedge y'\overset{*}{\to}z$ for all x , y , $y'\in\mathcal{S}$.

Let F be a set of function symbols accompanied with a mapping *arity* from F to the set of natural numbers, which is called a *signature*. Let X be a set of variables. We use x,y,z as variables. Terms over F and X are recursively defined by (1) and (2): (1) A variable $x\in X$ is a term, (2) If $f\in F$ with $arity(f)=n$ and $t_1,...,t_n$ are terms then $f(t_1,...,t_n)$ is a term. If $arity(f)=0$ then we write f instead of $f()$. Function symbols f with $arity(f)=0$ are called *constants*. $T(F,X)$ (or simply T) denotes the set of all terms over F and X. The set of variables which appears in a term t is denoted by $\mathcal{V}ar(t)$. For any term t we use $f^n(t)$ to denote $\overbrace{f(\cdots f}^{n}(t)\cdots)$. A term t is called *linear* if every variable occurs in t at most once. $top(t)$ represents a top symbol of t.

Letting $\Box$ be an extra constant, a term $C\in T(F\cup\{\Box\},X)$ is called a *context* denoted by $C[,...,]$. For a $C[,...,]$ which contains n $\Box$'s and for $t_1,...,t_n\in T(F,X)$, $C[t_1,...,t_n]$ denotes the term obtained by replacing $\Box$'s with $t_1,...,t_n$ from left to right. A context that possesses exactly one $\Box$ is denoted by $C[\ ]$. If $t\equiv C[s]$, we write $top(s)\in t$ and also $s\trianglelefteq t$.

A substitution is a mapping $\sigma: X\to T(F,X)$ such that $\mathcal{D}om(\sigma)$ is finite, where $\mathcal{D}om(\sigma)=\{x\mid\sigma(x)\not\equiv x\}$. Let $\{x_1\mapsto u_1,...,x_n\mapsto u_n\}$ denote a substitution σ such that $\sigma(x_i)\equiv u_i$ for all i and $\sigma(x)\equiv x$ for the other variables x. Then $t\sigma$ denotes the term obtained by replacing variables $x_1,...,x_n$ in t with terms $u_1,...,u_n$, respectively.

A *(oriented) conditional rewrite rule* is a triple $(l, r, \mathcal{C}ond)$, denoted by $l \to r \Leftarrow \mathcal{C}ond$, where the left-hand side $l(\notin X)$ and the right-hand side r are terms and $\mathcal{C}ond$ is either *true* or $l_1 \to r_1 \wedge \cdots \wedge l_n \to r_n$ $(n \geq 1)$[1]. For a substitution σ and a reduction relation $\to$ on terms, if $l_i\sigma \xrightarrow{*} r_i\sigma$ for all i then we say that σ and $\to$ satisfy $\mathcal{C}ond$ and write $\mathcal{C}ond(\sigma, \to)$. Letting R be a set of conditional rewrite rules, $(T(F, X), \to_R)$ is called a *(oriented) conditional term rewriting system* ((oriented) CTRS), where $\to_R = \bigcup_{n=0}^{\infty} \underset{n}{\to}$ and the n-level reduction $\underset{n}{\to}$, are defined by (1), (2) and (3): (1) $\underset{0}{\to} = \emptyset$, (2) If $s \underset{n}{\to} t$, then $s \underset{n+1}{\to} t$, (3) If $l \to r \Leftarrow \mathcal{C}ond \in R$ and $\mathcal{C}ond(\sigma, \underset{n}{\to})$, then $C[l\sigma] \underset{n+1}{\to} C[r\sigma]$. In the sequel, we identify R with $(T(F, X), \to_R)$ if that causes no confusion. For a conditional rewrite rule $l \to r \Leftarrow Cond$, we call the variables in $(\mathcal{V}ar(r) \cup \mathcal{V}ar(Cond)) - \mathcal{V}ar(l)$ *extra variables*. $\mathcal{EV}ar(t)$ denotes the set of extra variables in t. A *1-CTRS* has no extra variables in each rule, a *2-CTRS* has no extra variables on the right-hand side of each rule, a *3-CTRS* may contain extra variables on the right-hand sides of the rules provided that these also occur in the corresponding conditional part (i.e., $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l) \cup \mathcal{V}ar(c)$), and a *4-CTRS* contains extra variables without any restriction. We abbreviate a rule $l \to r \Leftarrow true$ as $l \to r$, and call it a *rewrite rule*. A *term rewriting system* (TRS) is a CTRS that has only rewrite rules. We have $\to_R = \underset{i}{\to}$ for all $i \geq 1$ on TRS R. We say that a conditional rewrite rule is *left-linear* (and *right-linear*, respectively) if any variable occurs in the left-hand side l (and the right-hand side r) at most once. A CTRS is called *left-linear* (and *right-linear*, respectively) if every rule is left-linear (and right-linear). The rules $l \to r \Leftarrow \mathcal{C}ond$ and $l' \to r' \Leftarrow \mathcal{C}ond'$ *overlap* if there exist a subterm s $(\notin X)$ of l' and substitutions σ and σ' such that $l\sigma \equiv s\sigma'$. A CTRS is *orthogonal* if it is left-linear and has no overlapping rules. An orthogonal TRS is known to be confluent.

3 Generation of Inverse TRSs

3.1 Inverse CTRSs

In this paper, we suppose that a signature F is devided to disjoint sets F_D of *defined symbols* and F_C of *constructors*, that is $F = F_D \oplus F_C$. We call $t \in T(F_C, X)$ a *constructor term*. We say that CTRS R is defined over $T(F, X)$ if $top(l) \in F_D$ for every rule $l \to r \Leftarrow Cond \in R$. We introduce the notion of *tuple* as special constructors for representing the return value of the inverses of multi-argument functions. A tuple of terms $t_1, ..., t_n$ is denoted by $tp_n(t_1, ..., t_n)$ with the special constructor tp_n. A tuple with one element $tp_1(t)$ is sometimes simply denoted by t.

Let R be a CTRS over $T(F, X)$. For a defined symbol $f \in F_D$ and $t_1, ..., t_n \in T(F_C, \emptyset)$, we say that $f(t_1, ..., t_n)$ is *defined* if $f(t_1, ..., t_n) \xrightarrow{*}_R s$ for some $s \in T(F_C, \emptyset)$.

Definition 1. *Let R be a CTRS over $T(F, X)$, F_0 be a subset of defined symbols F_D and R' be a CTRS over $T(G, X')$ such that $G_C = F_C \cup \{tp_i \mid 1 \leq i \leq n\}$ and $F_D \cap G_D = \emptyset$, where n is the maximum number of arity of $f \in F_D$. R' is an* inverse *of R w.r.t. F_0 if for all $f \in F_0$ there exists $g \in G_D$ such that for all $t_1, ..., t_n \in T(F_C, \emptyset)$ if $f(t_1, ..., t_n)$ is defined then $g(f(t_1, ..., t_n)) \xrightarrow{*}_{R \cup R'} tp_n(t_1, ..., t_n)$. And we say that g is an inverse function of f.* □

3.2 Input TRSs

This paper treats *constructor TRSs* (*constructor systems*) as the input of our algorithm. Every rule $f(t_1, ..., t_n) \to r$ of a constructor TRS over $T(F, X)$ satisfy $t_1, ..., t_n \in T(F_C, X)$. We use p, q as constructor terms. Note that the input constructor TRSs are not orthogonal in general.

Example 1. The function *mult* that calculates multiplication is defined as follows:

$$R_1 = \{\ add(0, x_2) \to x_2, \qquad mult(0, x_2) \to 0,$$
$$add(s(x_1), x_2) \to s(add(x_1, x_2)),\ mult(s(x_1), x_2) \to add(mult(x_1, x_2), x_2)\ \}$$

□

3.3 Algorithm for Generating Inverse CTRSs

First, we define the set of defined symbols on which specified function symbols depend.

Definition 2. *Let R be a constructor TRS over $T(F, X)$. Let F_0 be a subset of defined symbols F_D. Define $\mathcal{D}ep_0(F_0)$ to be the smallest set satisfying the followings:*

– $\mathcal{D}ep_0(F_0) \supseteq F_0$,

[1] Although conditional rewrite rules satisfy some conditions for variables and so on in general definitions, in this paper we treat more general one.

– $\mathcal{D}ep_0(F_0) \supseteq \{\ f \mid top(l) \in \mathcal{D}ep_0(F_0), l \to r \in R, f \in r, f \in F_D\ \}$. □

We show an algorithm $\mathcal{I}nv\mathcal{CTRS}_0$ that is an extension of the algorithm $\mathcal{I}nv$ that works on pure treeless TRSs[NN01]. The algorithm $\mathcal{I}nv\mathcal{CTRS}_0$ is a set of conditional rewrite rules generated by the algorithm $\mathcal{I}nv\mathcal{R}ule_0$, and $\mathcal{I}nv\mathcal{R}ule_0$ generates a CTRS from a constructor TRS R over $T(F, X)$, and a subset F_0 of the defined symbols F_D. $\mathcal{I}nv\mathcal{CTRS}_0$ and $\mathcal{I}nv\mathcal{R}ule_0$ are defined as follows:

$$\mathcal{I}nv\mathcal{CTRS}_0(R, F_0) = \{\ \mathcal{I}nv\mathcal{R}ule_0(\ l \to r\ ) \mid l \to r \in R,\ top(l) \in \mathcal{D}ep_0(F_0)\ \},$$

$$\mathcal{I}nv\mathcal{R}ule_0(f(p_1, ..., p_n) \to r) = f^{\#}(r') \to tp_n(p_1, ..., p_n) \Leftarrow Cond, \quad \text{where } \mathcal{T}_0(\ r\ ) = \langle\ r'\ ;\ Cond\ \rangle.$$

The procedure $\mathcal{T}_0$ that takes a term t as input and outputs the pair $\langle\ t'\ ;\ Cond\ \rangle$ is defined as follows.

$$\begin{cases} - \ \mathcal{T}_0(\ x\ ) = \langle\ x\ ;\ true\ \rangle. \\ - \ \mathcal{T}_0(\ c\ ) = \langle\ c\ ;\ true\ \rangle. \\ - \ \mathcal{T}_0(\ c(t_1, ..., t_n)\ ) = \langle\ c(t'_1, ..., t'_n)\ ;\ \mathcal{C}nd_1 \wedge \cdots \wedge \mathcal{C}nd_n\ \rangle\ (n \geq 1), \\ \qquad\qquad \text{where } \mathcal{T}_0(\ t_i\ ) = \langle\ t'_i\ ;\ \mathcal{C}nd_i\ \rangle \text{ for each } i. \\ - \ \mathcal{T}_0(\ f(t_1, ..., t_n)\ ) = \langle\ y\ ;\ f^{\#}(y) \to tp_n(t'_1, ..., t'_n) \wedge \mathcal{C}nd_1 \wedge \cdots \wedge \mathcal{C}nd_n\ \rangle, \\ \qquad\qquad \text{where } y \text{ is a fresh variable and } \mathcal{T}_0(\ t_i\ ) = \langle\ t'_i\ ;\ \mathcal{C}nd_i\ \rangle \text{ for each } i. \end{cases}$$

It is clear that the procedure $\mathcal{T}_0$ always terminates and the set $\mathcal{I}nv\mathcal{CTRS}_0(R, F_0)$ is finite.

Example 2. Consider the constructor TRS R_1 of Example 1. $R_2 = \mathcal{I}nv\mathcal{CTRS}_0(R_1, \{mult\})$ is as follows:

$$\begin{aligned} R_2 = \ \{\ & add^{\#}(x_2) \to tp_2(0, x_2), \\ & add^{\#}(s(y)) \to tp_2(s(x_1), x_2) \Leftarrow add^{\#}(y) \to tp_2(x_1, x_2), \\ & mult^{\#}(0) \to tp_2(0, x_2), \\ & mult^{\#}(y) \to tp_2(s(x_1), x_2) \Leftarrow add^{\#}(y) \to tp_2(z, x_2) \wedge mult^{\#}(z) \to tp_2(x_1, x_2)\ \ \} \end{aligned}$$

□

CTRSs obtained by $\mathcal{I}nv\mathcal{CTRS}_0$ are generally 4-CTRSs (see Example 2).

3.4 Correctness of the Algorithm

In this section, we prove that the CTRS generated by our algorithm is an inverse of the input.

Theorem 1. *Let R be a constructor TRS over $T(F, X)$, F_0 be a subset of defined symbols F_D and R' be $\mathcal{I}nv\mathcal{CTRS}_0(R, F_0)$. For all $f \in \mathcal{D}ep_0(F_0)$ and all $t_1, ..., t_n, s \in T(F_C, \emptyset)$, $f(t_1, ..., t_n) \xrightarrow{*}_R s$ if and only if $f^{\#}(s) \to_{R'} tp_n(t_1, ..., t_n)$.* □

Thanks to Theorem 1, a CTRS obtained from R by $\mathcal{I}nv\mathcal{CTRS}_0$ is the inverse CTRS of R in the sence of Definition 1.

Even if the input constructor TRS of our algorithm are confluent, the inverse CTRS of the input are not always confluent. Consider a many-to-one function f; $f(s) \equiv f(s')$ $(\equiv t)$ for some $s, s' \in T(F_C, \emptyset)$ with $s \not\equiv s'$. Since we have $f^{\#}(t) \to s$ and $f^{\#}(t) \to s'$ from Theorem 1, the inverses CTRS is not confluent. Moreover, a normal form of $f^{\#}(t_1, ..., t_n)$ with $t_i \in T(F_C, \emptyset)$ is not always a ground constructor term.

4 Inverses of Functions with Specified Arguments Fixed

In this section, we extend $\mathcal{I}nv\mathcal{CTRS}_0$ to $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ that generates a TRS implementing the inverses of Indexed functions.

4.1 Definition of Indexed Inversion

Letting f be a defined symbol with $arity(f) = n$, I is an *index* of f if $I \subseteq \{1, ..., n\}$. $|I|$ denotes a number of elements in I. We use I_k for representing the k-th smallest element of I where $1 \leq k \leq |I|$. We use I, J as index. By using a index, we fix some arguments of defined symbols, i.e., if f_I then each I_k-th argument is fixed for all k. For an index I of f, $\hat{I}$ denotes the index containing all elements not in I; $I \oplus \hat{I} = \{1, ..., arity(f)\}$. We call f_J an *indexed defined symbol*. Letting R be a constructor TRS over $T(F, X)$, $F_{\mathcal{I}}$ denotes a set of all indexed defined symbols for F; $F_{\mathcal{I}} = \{\ f_I \mid f \in F_D, I \subseteq \{1, ..., arity(f)\}\ \}$.

Definition 3. *Let R be a constructor TRS over $T(F,X)$ and F_0 be a subset of $F_{\mathcal{I}}$. Define $\mathcal{D}ep_{\mathcal{I}}(F_0)$ to be the smallest set satisfying the following.*

$$
\begin{aligned}
&-\mathcal{D}ep_{\mathcal{I}}(F_0) \supseteq F_0.\\
&-\mathcal{D}ep_{\mathcal{I}}(F_0) \supseteq \{\, h_I \mid f_J \in \mathcal{D}ep_{\mathcal{I}}(F_0), h \in F_D, f(p_1,...,p_n) \to C[...,h(t_1,...,t_m),...] \in R,\\
&\qquad\qquad I = \{\, i \mid \mathcal{V}ar(t_i) \cap \bigcup_{j=1}^{|\hat{J}|} \mathcal{V}ar(p_{\hat{J}_j}) = \emptyset \,\}, |I| < m \qquad\qquad \}.\\
&-\mathcal{D}ep_{\mathcal{I}}(F_0) \supseteq \{\, h, h_1, ..., h_k \mid f_J \in \mathcal{D}ep_{\mathcal{I}}(F_0), h \in F_D, f(p_1,...,p_n) \to C[...,h(t_1,...,t_m),...] \in R,\\
&\qquad\qquad \forall i, \mathcal{V}ar(t_i) \cap \bigcup_{j=1}^{|\hat{J}|} \mathcal{V}ar(p_{\hat{J}_j}), h_i \in \mathcal{D}ep_0(\{h\}) \qquad\qquad \}.
\end{aligned}
$$

□

Definition 4. *Let R be a constructor TRS over $T(F,X)$, F_0 be a subset of $F_{\mathcal{I}}$ and R' be a CTRS over $T(G,X')$ such that $G_C = F_C \cup \{tp_i \mid 1 \le i \le n\}$ and $F_D \cap G_D = \emptyset$, where n is the maximum number of arity of $f \in F_D$. R' is an* indexed-inverse *of R w.r.t. F_0 if for all $f_J \in F_0$ there exists $g \in G_D$ such that for all $t_1,...,t_n \in T(F_C,\emptyset)$ if $f(t_1,...,t_n)$ is defined then $g(f(t_1,...,t_n), tp_{|J|}(t_{J_1},...,t_{J_{|J|}})) \xrightarrow{*}_{R\cup R'} tp_{|\hat{J}|}(t_{\hat{J}_1},...,t_{\hat{J}_{|\hat{J}|}})$. And we say that g is an indexed-inverse function of f w.r.t. J. Especially, g is an inverse function of f if $|J| = 0$. Hence, this definition is an extension of Definition 1.* □

4.2 Algorithm of Indexed-Inverse TRSs

Let R be a signature F and a set X of variables, F_0 be a subset of $F_{\mathcal{I}}$. We extend $\mathcal{I}nv\mathcal{CTRS}_0$ and $\mathcal{I}nv\mathcal{R}ule_0$ to $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ and $\mathcal{I}nv\mathcal{R}ule_{\mathcal{I}}$, respectively:

$$
\begin{aligned}
\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R,F_0) = &\{\, \mathcal{I}nv\mathcal{R}ule_{\mathcal{I}}(f_J(p_1,...,p_n) \to r) \mid f(p_1,...,p_n) \to r \in R, f_J \in \mathcal{D}ep_{\mathcal{I}}(F_0) \,\}\\
&\cup \{\, l \to r \mid l \to r \in R, top(l) \in \mathcal{D}ep_{\mathcal{I}}(F_0) \cap F_D \,\},
\end{aligned}
$$

$$
\mathcal{I}nv\mathcal{R}ule_{\mathcal{I}}(f_J(p_1,...,p_n) \to r) = \begin{cases}
f_J^{\#}(C, tp_{|J|}(p_{J_1},...,p_{J_{|J|}})) \to tp_{|\hat{J}|}(p_{\hat{J}_1},...,p_{\hat{J}_{|\hat{J}|}}) \Leftarrow \mathcal{C}ond,\\
\qquad \text{where } \mathcal{T}_{\mathcal{I}}(\, r, X_0\,) = \langle\, C \,;\, \mathcal{C}ond \,\rangle,\ C \in T(F_C, X \cup X')\\
\qquad\quad \text{and } X' \text{ is a set of variables generated by } \mathcal{T}_{\mathcal{I}}(\, r, X'\,).\\
f_J^{\#}(C[y_1,...,y_m], tp_{|J|}(p_{J_1},...,p_{J_{|J|}})) \to tp_{|\hat{J}|}(p_{\hat{J}_1},...,p_{\hat{J}_{|\hat{J}|}})\\
\qquad\qquad \Leftarrow \mathcal{C}ond \wedge \bigwedge_{i=1}^{m} f_i(t_{i,1},...,t_{i,n_i}) \to y_i,\\
\qquad \text{where } \mathcal{T}_{\mathcal{I}}(\, r, X_0\,) = \langle\, C[f_1(t_{1,1},...,t_{1,n_1}),...,f_m(t_{m,1},...,t_{m,n_m})] \,;\, \mathcal{C}ond \,\rangle,\\
\qquad\quad C \in T(F_C, X \cup X'), \text{ and } X' \text{ is a set of variables generated by } \mathcal{T}_{\mathcal{I}}(\, r, X'\,).
\end{cases}
$$

where $X_0 = \bigcup_{i=1}^{|\hat{J}|} \mathcal{V}ar(p_{\hat{J}_i})$. Moreover, $\mathcal{T}_{\mathcal{I}}$ is defined as follow:

- $\mathcal{T}_{\mathcal{I}}(\, x, X''\,) = \langle\, x \,;\, true \,\rangle$.
- $\mathcal{T}_{\mathcal{I}}(\, c, X''\,) = \langle\, c \,;\, true \,\rangle$.
- $\mathcal{T}_{\mathcal{I}}(\, c(t_1,...,t_n), X''\,) = \langle\, c(t'_1,...,t'_n) \,;\, \mathcal{C}nd_1 \wedge \cdots \wedge \mathcal{C}nd_n \,\rangle\ (n \ge 1)$, where $\mathcal{T}_{\mathcal{I}}(\, t_i, X''\,) = \langle\, t'_i \,;\, \mathcal{C}nd_i \,\rangle$ for each i.
- $\mathcal{T}_{\mathcal{I}}(\, f(t_1,...,t_n), X''\,) = \begin{cases}
\langle\, y \,;\, f_J^{\#}(y, tp_{|J|}(t_{J_1},...,t_{J_{|J|}})) \to tp_{|\hat{J}|}(t'_{\hat{J}_1},...,t'_{\hat{J}_{|\hat{J}|}}) \wedge \mathcal{C}nd_{\hat{J}_1} \wedge \cdots \wedge \mathcal{C}nd_{\hat{J}_n} \,\rangle,\\
\qquad \text{where } y \text{ is a fresh variable, } J = \{\, i \mid \mathcal{V}ar(t_i) \cap X'' = \emptyset \,\},\ |J| < n\\
\qquad\quad \text{and } \mathcal{T}_{\mathcal{I}}(\, t_{\hat{J}_i}, X''\,) = \langle\, t'_{\hat{J}_i} \,;\, \mathcal{C}nd_{\hat{J}_i} \,\rangle \text{ for each } i.\\
\langle\, f(t_1,...,t_n) \,;\, true \,\rangle,\\
\qquad \text{where } J = \{\, i \mid \mathcal{V}ar(t_i) \cap X'' = \emptyset \,\} \text{ and } |J| = n.
\end{cases}$

where X'' is a set of varialbes. It is clear that the procedure $\mathcal{T}_{\mathcal{I}}$ always terminates and $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R,F_0)$ is finite.

Example 3. For TRS R_1 of Example 1, we obtain $R_3 = \mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R_1, \{mult_{\{2\}}\})$ as follows:

$$
\begin{aligned}
R_3 = \{\ & add^{\#}_{\{2\}}(x_2,x_2) \to 0,\\
& add^{\#}_{\{2\}}(s(y),x_2) \to s(x_1) \Leftarrow add^{\#}_{\{2\}}(y,x_2) \to x_1,\\
& mult^{\#}_{\{2\}}(0,x_2) \to 0,\\
& mult^{\#}_{\{2\}}(y,x_2) \to s(x_1) \Leftarrow add^{\#}_{\{2\}}(y,x_2) \to z \wedge mult^{\#}_{\{2\}}(z,x_2) \to x_1,\ \}.
\end{aligned}
$$

□

CTRSs obtained by $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ are generally 4-CTRSs (see Example 3).

Theorem 2. *Let R be a constructor TRS over $T(F,X)$, F_0 be a subset of $F_{\mathcal{I}}$ and R' be $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R,F_0)$. For all $f_J \in \mathcal{D}ep_{\mathcal{I}}(F_0)$ and all $t_1,...,t_n,s \in T(F_C,\emptyset)$, $f(t_1,...,t_n)\xrightarrow{*}_R s$ if and only if $f^{\#}_J(s,tp_{|J|}(t_{J_1},...,t_{J_{|J|}}))\rightarrow_{R'} tp_{|\hat{J}|}(t_{\hat{J}_1},...,t_{\hat{J}_{|\hat{J}|}})$.* □

Thanks to Theorem 2, a CTRS obtained from R by $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ is the indexed inverse CTRS of R in the sence of Definition 4.

Corollary 1. *Let R be a constructor TRS over $T(F,X)$. Let R be $\mathcal{I}nv\mathcal{CTRS}_0(R,\{f\})$ and R' be $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R,\{f_\emptyset\})$ for $f \in F_D$. For all $t,t_1,...,t_n \in T(F_C,\emptyset)$, $f^{\#}(t) \rightarrow_{R''} tp_n(t_1,...,t_n)$ if and only if $f^{\#}{}_{\emptyset}(t) \rightarrow_{R'} tp_n(t_1,...,t_n)$.* □

5 Transformation of CTRSs to TRSs

E.Ohlebusch have proposed the transformation of CTRSs to TRSs[Ohl01]. Although its input class is deterministic 3-CTRSs, it can transform CTRSs (deterministic 4-CTRSs), that is deterministic 3-CTRSs if each right-hand side of each rule has no extra variables, to TRSs. Since CTRSs obtained by our algorithms are deterministic 4-CTRSs, we can use Ohlebusch's transformation. In this section, we prove our CTRSs are deterministic 4-CTRSs, introduce Ohlebusch's transformation and show some examples that the transformation is applied to our CTRSs.

5.1 Deterministic 3-CTRSs

At first we define deterministic 3-CTRSs.

Definition 5. *[Ohl01] Let R be an oriented 3-CTRS. A conditional rewrite rule $l \rightarrow r \Leftarrow s_1 \rightarrow t_1 \wedge \cdots \wedge s_k \rightarrow t_k \in R$ is called* deterministic *if for every i we have $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$. In the following, we will frequently use the notation $\mathcal{EV}ar(t_i) = \mathcal{V}ar(t_i) - (\mathcal{V}ar(l) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j))$.The CTRS R is called* deterministic *if every rewrite rule in R is deterministic.* □

Let R be a 4-CTRS. We call R a *deterministic 4-CTRS* if for each rule $l \rightarrow r \Leftarrow s_1 \rightarrow t_1 \wedge \cdots \wedge s_k \rightarrow t_k \in R$, for every i we have $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l) \cup \bigcup_{j=1}^{i-1} \mathcal{V}ar(t_j)$. The difference between deterministic 3-CTRSs and deterministic 4-CTRSs is whether r has extra variables appearing only in r.

Theorem 3. *CTRSs obtained by $\mathcal{I}nv\mathcal{CTRS}_0$ are deterministic 4-CTRSs.* □

Corollary 2. *Let R be a constructor TRS over $T(F,X)$ and F_0 be a subset of F_D. If $\mathcal{V}ar(l) = \mathcal{V}ar(r)$ for each rule $l \rightarrow r \in R$ then $\mathcal{I}nv\mathcal{CTRS}_0(R,F_0)$ is a deterministic 3-CTRS.* □

Theorem 4. *CTRSs obtained by $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ are deterministic 4-CTRSs.* □

Corollary 3. *Let R be a constructor TRS over $T(F,X)$ and F_0 be a subset of $F_{\mathcal{I}}$. $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R,F_0)$ is a deterministic 3-CTRS if the following propaties are satisfied.*

- $\mathcal{V}ar(r) = \bigcup_{i=1}^{|\hat{J}|} \mathcal{V}ar(p_{\hat{J}_i})$ *for each rule $f_J(p_1,...,p_n) \rightarrow r \in R$ where $f_J \in \mathcal{D}ep_0(F_0)$.*
- $\mathcal{V}ar(r) = \bigcup_{i=1}^{n} \mathcal{V}ar(p_i)$ *for each rule $f(p_1,...,p_n) \rightarrow r \in R$ where $f \in \mathcal{D}ep_{\mathcal{I}}(F_0)$.*

□

5.2 Ohlebusch's Transformation

We introduce Ohlebusch's transformation of a deterministic 3-CTRS R into a TRS $\mathcal{U}(R)$.

Definition 6. *[Ohl01] Let R be a deterministic 3-CTRS over $T(F,X)$. We use a label for each rewrite rule in R, i.e., $\rho : l \rightarrow r \Leftarrow Cond$. For each conditional rewrite rule $l \rightarrow r \Leftarrow s_1 \rightarrow t_1 \wedge \cdots \wedge s_k \rightarrow t_k \in R$, we need k fresh defined symbols $\mathcal{U}^{\rho}_1,...,\mathcal{U}^{\rho}_k$ in the transformation. Moreover, by abuse of notation, $\mathcal{V}ar$ and $\mathcal{EV}ar$ denote functions which assigns the sequence of the variables (in some fixed order) in the set $\mathcal{V}ar(t)$ and $\mathcal{EV}ar(t)$ to a term t, respectively. We transform $l \rightarrow r \Leftarrow s_1 \rightarrow t_1 \wedge \cdots \wedge s_k \rightarrow t_k \in R$ into a set $\mathcal{U}(\rho)$ of $k+1$ rewrite rules as follows:*

$$\begin{aligned}\{\quad & l \rightarrow \mathcal{U}^{\rho}_1(s_1,\mathcal{V}ar(l)),\\ & \mathcal{U}^{\rho}_1(t_1,\mathcal{V}ar(l)) \rightarrow \mathcal{U}^{\rho}_2(s_2,\mathcal{V}ar(l),\mathcal{EV}ar(t_1)),\\ & \mathcal{U}^{\rho}_2(t_2,\mathcal{V}ar(l),\mathcal{EV}ar(t_1)) \rightarrow \mathcal{U}^{\rho}_2(s_3,\mathcal{V}ar(l),\mathcal{EV}ar(t_1),\mathcal{EV}ar(t_2)),\\ & \vdots\\ & \mathcal{U}^{\rho}_k(t_k,\mathcal{V}ar(l),\mathcal{EV}ar(t_1),...,\mathcal{EV}ar(t_{k-1})) \rightarrow r \qquad \}.\end{aligned}$$

Note that $\mathcal{U}(l \rightarrow r) = \{\, l \rightarrow r \,\}$ for every rewrite rule $l \rightarrow r$ in R. □

For a conditional rewrite rule $\rho : l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k \in R$, r appears only in the right-hand side of the $k+1$-th rewrite rule of $\mathcal{U}(\rho)$. Hence, $\mathcal{U}$ can be used for deterministic 4-CTRSs.

$\mathcal{I}nv_0(R, F_0)$ and $\mathcal{I}nv_{\mathcal{I}}(R, F_0)$ denote $\mathcal{U}(\mathcal{I}nv\mathcal{CTRS}_0(R, F_0))$ and $\mathcal{U}(\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R, F_0))$, respectively.

Theorem 5. *Let R be a constructor TRS over $T(F, X)$ and F_0 be a subset of F_D. If R is left-linear and right-linear then $\mathcal{I}nv_0(R, F_0)$ is left-linear.* □

Example 4. We obtain the following CTRS $R_4 = \mathcal{U}(\mathcal{I}nv\mathcal{CTRS}_0(R_1, \{mult\}))$:

$$\begin{aligned} R_4 = \{\ & add^{\#}(x_2) \to tp_2(0, x_2), && mult^{\#}(0) \to tp_2(0, x_2), \\ & add^{\#}(s(y)) \to \mathcal{U}_1^1(add^{\#}(y), y), && mult^{\#}(y) \to \mathcal{U}_1^2(add^{\#}(y), y), \\ & \mathcal{U}_1^1(tp_2(x_1, x_2), y) \to tp_2(s(x_1), x_2)), && \mathcal{U}_1^2(tp_2(z, x_2), y) \to \mathcal{U}_2^2(mult^{\#}(z), y, z, x_2), \\ & && \mathcal{U}_2^2(tp_2(x_1, x_2), y, z, x_2) \to tp_2(s(x_1), x_2) \quad \}. \end{aligned}$$

□

Theorem 6. *Let R be a deterministic 4-CTRS over $T(F, X)$ and R' be $\mathcal{U}(R)$. For all $f \in F_D$ and all $t, t_1, ..., t_n, \in T(F_C, \emptyset)$, $f(t_1, ..., t_n) \xrightarrow{*}_R t$ if and only if $f(t_1, ..., t_n) \xrightarrow{*}_{R'} t$.* □

Corollary 4. *Let R be a constructor TRS over $T(F, X)$. Let F_0 be a subset of defined symbols F_D and R' be $\mathcal{I}nv_0(R, F_0)$. Letting R'' be $\mathcal{I}nv_0(R', \{f^{\#} \mid f \in F_0\})$, R and R'' are equivalent such that for all $f \in F_0$ and for all $t_1, ..., t_n, t \in T(F_C, \emptyset)$, $f(t_1, ..., t_n) \xrightarrow{*}_{R'} t$ if and only if $f^{\#\#}(tp_m(t_1, ..., t_n)) \xrightarrow{*}_{R''} t$.* □

Corollary 5. *Let R be a constructor TRS over $T(F, X)$. Let F_0 be a subset of $F_{\mathcal{I}}$ and R' be $\mathcal{I}nv_{\mathcal{I}}(R, F_0)$. Letting R'' be $\mathcal{I}nv_{\mathcal{I}}(R', \{f^{\#}{}_{J\{1\}} \mid f_J \in F_0\})$, R and R'' are equivalent such that for all $f \in F_0$ and for all $t_1, ..., t_n, t \in T(F_C, \emptyset)$, $f(t_1, ..., t_n) \xrightarrow{*}_{R'} t$ if and only if $f_J^{\#\#}{}_{\{1\}}(tp_{|\hat{J}|+1}(tp_{|J|}(t_{J_1}, ..., t_{J_{|J|}}), t_{\hat{J}_1}, ..., t_{\hat{J}_{|\hat{J}|}})) \xrightarrow{*}_{R''} t$.* □

We treats $f(tp_n(t_1, ..., t_n))$ as $f(t_1, ..., t_n)$ if $arity(f) = n$.

Example 5. Consider the following TRS:

$$\begin{aligned} R_5 = \{\ & add(0, 0) \to 0, && mult(0, x_2) \to 0, \\ & add(0, s(x_2)) \to s(x_2), && mult(s(x_1), 0) \to 0, \\ & add(s(x_1), x_2) \to s(sdd(x_1, x_2)), && mult(s(x_1), s(x_2)) \to s(add(mult(x_1, x_2), x_1), x_2)) \ \}. \end{aligned}$$

TRS R_6 obtained by $\mathcal{I}nv\mathcal{CTRS}_0(R_5, \{mult\})$ as follows is terminating.

$$\begin{aligned} R_6 = \{\ & add^{\#}(0) \to tp_2(0, 0),\ add^{\#}(s(x_2)) \to tp_2(0, s(x_2)), \\ & add^{\#}(s(y)) \to tp_2(s(x_1), x_2) \Leftarrow add^{\#}(y) \to tp_2(x_1, x_2), \\ & mult^{\#}(0) \to tp_2(0, x_2),\ mult^{\#}(0) \to tp_2(s(x_1), 0), \\ & mult^{\#}(s(y)) \to tp_2(s(x_1), s(x_2)) \\ & \qquad \Leftarrow add^{\#}(y) \to tp_2(z, x_2) \wedge add^{\#}(z) \to tp_2(w, x_1) \wedge mult^{\#}(w) \to tp_2(x_1, x_2) \ \}. \end{aligned}$$

□

5.3 Reversible TRSs

Definition 7. *Let R be a constructor TRS over $T(F, X)$ and F_0 be a subset of $F_{\mathcal{I}}$. We say that a CTRS R' is reversible (w.r.t. F_0) if R' satisfy the followings:*

- $R' \supseteq R$,
- $R' \supseteq \mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R, \{\ f_J \mid f \in F_D, J \subseteq \{1, ..., arity(f)\}\ \})$ *($R' \supseteq \mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}(R, F_0)$).*

An we say that a TRS R'' is reversible (w.r.t. F_0) if R'' satisfy the followings:

- $R'' \supseteq R$,
- $R'' \supseteq \mathcal{I}nv_{\mathcal{I}}(R, \{\ f_J \mid f \in F_D, J \subseteq \{1, ..., arity(f)\}\ \})$ *($R' \supseteq \mathcal{I}nv_{\mathcal{I}}(R, F_0)$).*

□

6 Conclusion

In this paper, to propose an algorithm for generating a TRS implementing the inverses of the functions with specified arguments fixed, we have proposed an algorithm $\mathcal{I}nv\mathcal{CTRS}_0$ for generating inverse CTRSs for constructor TRSs and extended $\mathcal{I}nv\mathcal{CTRS}_0$ to the algorithm $\mathcal{I}nv\mathcal{CTRS}_{\mathcal{I}}$ so that generate a TRS implementingthe inverses of the functions with specified arguments fixed. Moreover, we have shown that our inverse CTRSs can be transformed to TRSs by using Ohlebusch's transformation. Hence, we can use TRSs obatained by $\mathcal{I}nv_{\mathcal{I}}$ as follows:

$$\begin{cases} gcd(x,y) \to gcd(add^{\#}_{\{2\}}(x,y),y) \\ gcd(x,0) \to x \\ gcd(x,y) \to gcd(y,x) \Leftarrow x < y \\ \qquad\qquad \vdots \end{cases}$$

As future works, we need to discuss some properties of our inverse CTRSs and TRSs, i.e., termination, and to compare other methods by efficiency.

References

[BN98] F. Baader, T. Nipkow: Term Rewriting and All That. CAMBRIDGE Univ. Press, 1998.

[Chin92] W. Chin: Safe Fusion of Functional Expressions. In ACM Conference on Lisp and Functional Programming, San Francisco, Ca., pp.11-20, ACM, June 1992.

[Der99] N. Dershowitz, S. Mitra: Jeopardy. LNCS 1631 Rewriting Techniques and Applications, pp.16-29, 1999.

[Klop92] J.W.Klop: Term Rewriting Systems. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, Hnadbook of Logic in Computer Science, volume 2, pp.2-116. Oxford University Press, 1992.

[La75] D.Lankford: Canonical Algebraic Simplification in Computational Logic. Technical Report ATP-25, Department of Mathematics, University of Texas, Austin, 1975.

[NN01] N.Nishida, M.Sakai and T.Sakabe: Generation of Inverse Term Rewriting Systems for Pure Treeless Functions. RPC'01 Proceedings (Y.Toyama ed.), pp.188-198, Sendai, Japan, 2001.

[Ohl01] E.Ohlebusch: Termination of Logic Programs: Transformational Methods Revisited. Applicable Algebra in Engineering, Communication and Computing, Vol.12, pp.73-116, 2001.

[Plot72] G.Plotkin: Building-in Equational Theories. Machine Intelligence, volume 7, pp.73-90, 1972.

[Sla74] J.R.Slagle: Automated Theorem Proving for Theories with Simplifiers, Commutativity and Associativity. J.ACM, volume 21, pp.622-642. 1974.