Generation of Inverse Term Rewriting Systems for Pure Treeless Functions

Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe

Graduate School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
nishida@sakabe.nuie.nagoya-u.ac.jp
sakai@nuie.nagoya-u.ac.jp
sakabe@nuie.nagoya-u.ac.jp

Abstract. In this paper, we present an algorithm for generating a conditional TRS that implements the inverses of pure treeless functions defined by a pure treeless TRS which is a restricted orthogonal TRS. Our algorithm constructs conditional rules from a pure treeless TRS in such a way that each rule is reversed and then is modified according to the idea of the fusion transformation. We prove that the conditional TRS constructed by our algorithm implements the inverses of those functions defined by a given pure treeless TRS. Moreover, we show that the conditional TRS generated by our algorithm can be transformed to an equivalent TRS if the input pure treeless TRS of the algorithm is right-linear.

1 Introduction

For solving an equation over the functions defined by a functional program, we can use "E-unification"[BN98,Plot72], "Narrowing"[BN98,Sla74,La75] and "Inversion Algorithm" [Der99]. These methods search a large space for finding a solution everytime when given a functional program and an equation, and hence they are not efficient in general. In contrast, if we can get a functional program which computes the inverse functions, we may solve the equation efficiently by using the inverses in a same manner as solving algebraic equations.

This paper proposes an algorithm which generates for an input functional program a conditional TRS that computes the inverses of functions of the input functional program. The inputs of our algorithm are restricted to pure treeless TRSs which is restricted orthogonal TRSs. Our algorithm is based on the idea used in the fusion transformation[Chin92]. We prove that our algorithm eventually terminates, and that the conditional TRS produced by our algorithm implements the inverses of those functions defined by a given pure treeless TRS. Moreover, we show that the conditional TRS generated by our algorithm is transformed to an equivalent TRS if the input pure treeless TRS is right-linear.

2 Preparation

In this paper, we mainly follow the notation of [BN98,Klop92]. A *reduction system* is a structure $\mathcal{A} = (\mathcal{S}, \rightarrow)$ where $\mathcal{S}$ is a set and $\rightarrow$ is a binary relation over $\mathcal{S}$, which is called a reduction relation. A *reduction* is a finite sequence $x_0 \to x_1 \to \cdots \to x_n$ $(n \geq 0)$ 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 $\overset{k\geq}{\to}$ 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 as follows.

- A variable $x \in X$ is a term.
- 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.

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 $C[\ ]$.

A substitution is a mapping $\sigma : X \to T(F, X)$ such that $\sigma(x) \not\equiv x$ for finitely many variables 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 *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 $l_1 \to r_1 \wedge \cdots \wedge l_n \to r_n$ $(n \geq 1)$ with $\mathcal{V}ar(l) \cup (\bigcup_{i=1}^{n} \mathcal{V}ar(l_i) \cup \mathcal{V}ar(r_i)) \supseteq \mathcal{V}ar(r)$. For a substitution σ and a reduction relation $\to$ on terms, if $l_i\sigma \overset{*}{\to} r_i\sigma$ 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 *conditional term rewriting system* (CTRS), where $\to_R = \bigcup_{n=0}^{\infty} \underset{n}{\to}$ and $\underset{n}{\to}$, called the n-level reduction, are defined as follows.

- $\underset{0}{\to} = \emptyset$.
- If $s \underset{n}{\to} t$, then $s \underset{n+1}{\to} t$.
- 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 $(T(F,X), \rightarrow_R)$ if that causes no confusion. We abbreviate a rule $l \rightarrow r \Leftarrow true$ as $l \rightarrow r$, and call it a *rewrite rule.* A *term rewriting system* (TRS) is a CTRS that has only rewrite rules. We have $\rightarrow_R = \underset{i}{\rightarrow}$ 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). $l \rightarrow r \Leftarrow \mathcal{C}ond$ and $l' \rightarrow r' \Leftarrow \mathcal{C}ond'$ are *overlapping* if there exist a subterm s 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

Let R be a CTRS over a signature F and a set X of variables. The set F_D of *defined symbols* of R is $\{f \mid f(...) \rightarrow r \Leftarrow \mathcal{C}ond \in R\}$. A function symbol in $F - F_D$ is called a *constructor.* F_C denotes a set of constructors of R. We use f, g, h as defined symbols and c, d, e as constructors. R_f denotes the set of rewrite rules of $f \in F_D$ in R: $\{l \rightarrow r \Leftarrow \mathcal{C}ond \mid l \equiv f(...)\}$. We introduce the notion of *tuple* as special constructors for representing the return value of the inverses of n-ary functions. A tuple of tems $t_1, ..., t_n$ is denoted by $tp_n(t_1, ..., t_n)$ with the constructor tp_n. A tuple with one element $tp_1(t)$ is simply denoted by t.

Let R be a CTRS over a signature F and a set X of variables. 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 $f(t_1, ..., t_n) \overset{*}{\rightarrow}_R s$ for some $s \in T(F_C, \emptyset)$.

Definition 1. *Let R be a CTRS over a signature F, F_0 be a subset of defined symbols F_D and R' be a CTRS over a signature G 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 with respect to 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 $g(f(t_1, ..., t_n)) \overset{*}{\rightarrow}_{R \cup R'} tp_n(t_1, ..., t_n)$.* □

Let R be a CTRS over a signature F and a set X of variables. Let F_0 be a subset of defined symbols F_D. We call R' an *inverse CTRS* if R' is inverse of R w.r.t. F_0.

3.2 Pure Treeless TRS

An orthogonal TRS R is *pure treeless* if R_f is in the form of either `TYPE1` or `TYPE2` for all $f \in F_D$:

– `TYPE1`
$$R_f = \{f(x_1, ..., x_n) \rightarrow tt_0\}$$

– TYPE2

$$\begin{aligned} R_f = \{ \ & f(c_1(x'_1, ..., x'_{n_1}, x_2, ..., x_n) \to tt_1, \\ & \vdots \\ & f(c_m(x'_1, ..., x'_{n_m}, x_2, ..., x_n) \to tt_m \ \} \end{aligned}$$

where each term tt_i satisfies that the arguments of every defined symbol in tt_i are variables and $c_1, ..., c_m$ are different from each other. A term as like tt_i and a term $c(x_1, ..., x_n)$ are called a *pure treeless term* (*PT term*) and a *pattern*, respectively. A orthogonal TRS which is pure treeless is called a *pure treeless term rewriting system* (*PT TRS*). Note that a PT term is represented in the following form:

$$C[f_1(x_{1,1}, ..., x_{1,n_1}), ..., f_m(x_{m,1}, ..., x_{m,n_m})]$$

where $C \in T(F_C \cup \{\Box\}, X)$ is a context that contains no defined symbols.

A defined symbol in a PT TRS is called a *pure treeless function* [1] (*PT function*).

Example 1. The PT function *add* that calculates addition is defined as follows:

$$\begin{aligned} R_1 = \ \{ \ & add(0, x_2) \to x_2, \\ & add(s(x_1), x_2) \to s(add(x_1, x_2)) \ \ \}. \end{aligned}$$

$\Box$

3.3 Basic Idea

We explain our basic idea by using a TYPE1 PT function with one argument. Let's consider the PT TRS consisting of the rule:

$$f(x) \to C[f_1(x), ..., f_m(x)]$$

where $C \in T(F_C \cup \{\Box\}, X)$.

Firstly, we replace the defined symbol $f_i(x)$ in the right-hand side with a fresh variable y_i, for each i. Secondly, by applying the inverse function symbol f^{-1} of f to both hand sides, the left-hand side is transformed to x, since "$f^{-1}(f(n)) = n$" from the desired property of inverse functions. Then, we obtain the following rule:

$$f^{-1}(C[y_1, ..., y_m]) \to x.$$

In order to relate x to y_i's, we need to add equations $x \equiv f_1^{-1}(y_1)$, ..., $x \equiv f_m^{-1}(y_m)$ as the condition of the rule. Thus, we obtain the following conditional rewrite rule that defines the inverse function of f.

$$f^{-1}(C[y_1, ..., y_m]) \to x \ \Leftarrow \ \bigwedge_{i=1}^{m} f_i^{-1}(y_i) \to x$$

[1] In [Chin92], functions defined by TYPE1 and TYPE2 are called pure treeless functions and g-type pattern matching, respectively. In this paper, we call the both pure treeless functions.

3.4 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 PT TRS over a signature F and a set X of variables. Let F_0 be a subset of defined symbols F_D. Define $\mathcal{D}ep(F_0)$ to be the smallest set satisfying the following.*

- $\mathcal{D}ep(F_0) \supseteq F_0$.
- $\mathcal{D}ep(F_0) \supseteq \{h \in F_D \mid f \in \mathcal{D}ep(F_0), f(...) \to C[..., h(x_1, ..., x_n), ...] \in R\ \}$.

□

We show an algorithm $\mathcal{I}nv$ that is a formalization of Section 3.3. The algorithm $\mathcal{I}nv$ generates a CTRS from a PT TRS R over a signature F and a set X of variables, and a subset F_0 of the defined symbols F_D. $\mathcal{I}nv$ is defined as follows:

$$\begin{aligned}\mathcal{I}nv(R, F_0) = \{\ & h^{\#}(s) \to tp_n(t_1, x_2, ..., x_n) \ \Leftarrow\ Cond \\ & \mid h \in \mathcal{D}ep(F_0),\ h(t_1, x_2, ..., x_n) \to t \in R,\ \mathcal{T}[\![t]\!] = \langle\, s\,;\, Cond\,\rangle\ \}.\end{aligned}$$

Note that t_1 is a variable or a pattern. The procedure $\mathcal{T}$ takes a PT term t as input, and outputs the pair $\langle\, s\,;\, Cond\,\rangle$. $\mathcal{T}$ is defined as follows.

$$\begin{cases} -\ \mathcal{T}[\![x]\!] = \langle\, x\,;\, true\,\rangle. \\ -\ \mathcal{T}[\![c]\!] = \langle\, c\,;\, true\,\rangle. \\ -\ \mathcal{T}[\![c(t_1, ..., t_n)]\!] = \langle\, c(t'_1, ..., t'_n)\,;\, \mathcal{C}nd_1 \wedge \cdots \wedge \mathcal{C}nd_n\,\rangle\ (n \geq 1), \\ \qquad\qquad \text{where } \mathcal{T}[\![t_i]\!] = \langle\, t'_i\,;\, \mathcal{C}nd_i\,\rangle \text{ for each } i. \\ -\ \mathcal{T}[\![f(x_1, ..., x_n)]\!] = \langle\, y\,;\, f^{\#}(y) \to tp_n(x_1, ..., x_n)\,\rangle, \\ \qquad\qquad \text{where } y \text{ is a fresh variable.} \end{cases}$$

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

A tuple whose arguments are only variables $tp_n(x_1, ..., x_n)$ or whose first argument is a pattern and the rests are variables $tp_n(c(x'_1, ..., x'_j), x_2, ..., x_n)$, is called a *$n$-variable tuple*.

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

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

□

3.5 Correctness of the Algorithm

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

Proposition 1. *Let R be a PT TRS over a signature F and a set X of variables. Let F_0 be a subset of defined symbols F_D. For all $f \in \mathcal{D}ep(F_0)$, if $f(...) \to r \in R$ then defined symbol that appears in r belongs to $\mathcal{D}ep(F_0)$.*

Proof. It is trivial from the construction of $\mathcal{D}ep(F_0)$.

Theorem 1. *Let R be a PT TRS over a signature F and a set X of variables. Let F_0 be a subset of defined symbols F_D and R' be $\mathcal{I}nv(R, F_0)$. For all $f \in \mathcal{D}ep(F_0)$ and all $t_1, ..., t_n, s \in T(F_C, \emptyset)$,*

$$f(t_1, ..., t_n) \overset{*}{\to}_R s \quad \overset{\text{iff}}{\Longleftrightarrow} \quad f^{\#}(s) \to_{R'} tp_n(t_1, ..., t_n).$$

Proof. $\Rightarrow$). We prove this direction by induction on the steps k of the reduction $f(t_1, ..., t_n) \overset{*}{\to}_R s$. Since $k \geq 1$ from $s \in T(F_C, \emptyset)$, this reduction is written as follows:

$$f(t_1, ..., t_n) \to_R t \overset{k-1}{\to}_R s.$$

- If f is the `TYPE1` PT function, we have $f(x_1, ..., x_n) \to u \in R$ and $t \equiv u\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$. Since u is a PT term, u is represented in the following form with a context $C \in T(F_C \cup \{\Box\}, X)$:

  $$u \equiv C[f_1(x_{1,1}, ..., x_{1,a_1}), ..., f_l(x_{l,1}, ..., x_{l,a_l})]$$

  where $x_{i,j} \in \{x_1, ..., x_n\}$ for all i and j, f_i is in $\mathcal{D}ep(F_0)$ for every i from Proposition 1. Hence, t is represented as follows:

  $$t \equiv u\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\} \equiv C'[f_1(t_{1,1}, ..., t_{1,a_1}), ..., f_l(t_{l,1}, ..., t_{l,a_l})]$$

  where $C' \equiv C\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ and $t_{i,j} \equiv x_{i,j}\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ for each i and j.

  Since $t \equiv C'[f_1(t_{1,1}, ..., t_{1,a_1}), ..., f_l(t_{l,1}, ..., t_{l,a_l})] \overset{k-1}{\to}_R s$ and C' has no defined symbol in F_D, s is represented as $C'[s_1, ..., s_l]$ for some $s_i \in T(F_0, \emptyset)$ and the following reduction exists:

  $$f_i(t_{i,1}, ..., t_{i,a_i}) \overset{k-1\geq}{\to}_R s_i \quad (1 \leq i \leq l).$$

  Hence, we have the following reduction by induction hypothesis:

  $$f_i^{\#}(s_i) \overset{*}{\to}_{R'} tp_{a_i}(t_{i,1}, ..., t_{i,a_i}) \quad (1 \leq i \leq l). \tag{1}$$

  Since $f(x_1, ..., x_n) \to u \in R$ and C has no defined symbol, we have $\mathcal{T}[\![C[f_1(x_{1,1}, ..., x_{1,a_1}), ..., f_l(x_{l,1}, ..., x_{l,a_l})]]\!] = \langle\, C[y_1, ..., y_l]\ ;\ Cond\, \rangle$, where $Cond = \bigwedge_{i=1}^{l} (f_i^{\#}(y_i) \to tp_{a_i}(x_{i,1}, ..., x_{i,a_i})\ )$. Thus, R' has the following rule:

  $$f^{\#}(C[y_1, ..., y_l]) \to tp_n(x_1, ..., x_n) \ \Leftarrow\ Cond.$$

Let m be the maximum level of the reduction (1) and σ be $\{x_1 \mapsto t_1, ..., x_n \mapsto t_n, y_1 \mapsto s_1, ..., y_l \mapsto s_l\}$. Then we have $Cond(\sigma, \underset{m}{\rightarrow}_{R'})$. Therefore, the following reduction exists:

$$\begin{aligned} f^{\#}(s) &\equiv f^{\#}(C'[s_1,...,s_l]) \equiv \ f^{\#}(C[y_1,...,y_l])\sigma \\ &\underset{m+1}{\rightarrow}_{R'} tp_n(x_1,...,x_n)\sigma \equiv tp_n(t_1,...,t_n). \end{aligned}$$

- In the case that f is the `TYPE2` PT function, the claim is shown in similar to `TYPE1`.

$\Leftarrow$). We prove this direction by induction on the level k of the recution $f^{\#}(s) \rightarrow_{R'} tp_n(t_1,...,t_n)$. We suppose that the level of this reduction is k as follows:

$$f^{\#}(s) \underset{k}{\rightarrow}_{R'} tp_n(t_1,...,t_n).$$

The rules used in the reduction is in the following form from the construction of R':

$$f^{\#}(u) \rightarrow tp_n(u_1, x_2, ..., x_n) \ \Leftarrow \ Cond.$$

We suppose that $f^{\#}(s)$ is rewritten by the rule above.

- Consider the case that u_1 is variable x_1. Then, we have $s \equiv u\sigma$ for some substitution such that $x_i \equiv t_i$.
  - If $Cond = true$, we have $u \in T(F_C, \{x_1,...,x_n\})$ and $f(x_1,...,x_n) \rightarrow u \in R$ by the definition of $\mathcal{I}nv(R, F_0)$. Therefore, the following redution holds:
  $$f(t_1,...,t_n) \equiv f(x_1,...,x_n)\sigma \rightarrow_R u\sigma \equiv s.$$
  - If $Cond = \bigwedge_{i=1}^{m} f_i^{\#}(y_i) \rightarrow tp_{n_i}(x_{i,1},...,x_{i,n_i})$ where $m \geq 1$ and $n_i = arity(f_i)$ and $x_{i,j} \in \{x_1,...,x_n\}$ for each i and j, we have $u \equiv C[y_1,...,y_m]$ with a context $C \in T(F_C, \{x_1,...,x_n\})$ and $f(x_1,...,x_n) \rightarrow C[f_1(x_{1,1},..., x_{1,n_1}),...,f_m(x_{m,1},...,x_{m,n_m})] \in R$ by the definition of $\mathcal{I}nv(R, F_0)$. From $Cond(\sigma, \rightarrow_{R'})$ and the form of rules in R', the following reduction holds:
  $$f^{\#}(y_i\sigma) \underset{k_i}{\rightarrow}_{R'} tp_{n_i}(x_{i,1}\sigma,...,x_{i,n_i}\sigma) \quad (1 \leq i \leq m).$$
  Hence, we have the following reduction by induction hypothesis:
  $$f_i(x_{i,1}\sigma,...,x_{i,n_i}\sigma) \overset{*}{\rightarrow}_R y_i\sigma \quad (1 \leq i \leq m).$$
  Therefore, the following reduction holds:
  $$\begin{aligned} f(t_1,...,t_n) &\equiv f(x_1,...,x_n)\sigma \\ &\rightarrow_R C[f_1(x_{1,1},...,x_{1,n_1}),...,f_m(x_{m,1},...,x_{m,n_m})]\sigma \\ &\equiv C[f_1(x_{1,1},...,x_{1,n_1})\sigma,...,f_m(x_{m,1},...,x_{m,n_m})\sigma] \\ &\equiv C[f_1(x_{1,1}\sigma,...,x_{1,n_1}\sigma),...,f_m(x_{m,1}\sigma,...,x_{m,n_m}\sigma)] \\ &\overset{*}{\rightarrow}_R C[y_1\sigma,...,y_m\sigma] \equiv C[y_1,...,y_m]\sigma \equiv u\sigma \equiv s. \end{aligned}$$

- In the case that u_1 is a pattern, the claim is shown in similar to the case that u_1 is a variable.

$\square$

Thanks to Theorem 1, a CTRS obtained from R by our algorithm is the inverse CTRS of R in the sence of Definition 1.

Example 3. Consider $add(s(0), s^2(0)) \xrightarrow{*}_{R_1} s^3(0)$ where R_2 is generated from R_1 in Example 2. The following holds by Theorem 1.

$$add^{\#}(s^3(0)) \rightarrow_{R_2} tp_2(s(0), s^2(0)) \tag{2}$$

In fact, the following reduction exists by using the rule $add^{\#}(x_2) \rightarrow_{R_2} tp_2(0, x_2) \in R_2$ with $\{x_2 \mapsto s^2(0)\}$:

$$add^{\#}(s^2(0)) \rightarrow_{R_2} tp_2(0, s^2(0)).$$

Since the condition of the rule $add^{\#}(s(y)) \rightarrow tp_2(s(x_1), x_2) \Leftarrow add^{\#}(y) \rightarrow tp_2(x_1, x_2) \in R_2$ is satisfied with $\{y \mapsto s^2(0), x_1 \mapsto 0, x_2 \mapsto s^2(0)\}$, we obtain the reduction (2). $\square$

Although PT TRSs are confluent, the CTRSs generated by our algorithm 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 $f^{\#}(t) \rightarrow s$ and $f^{\#}(t) \rightarrow s'$ from Theorem 1, the inverses CTRS is not confluent.

3.6 Transformation of Inverse CTRS to Inverse TRS

In this section, we describe the way to transform a CTRS generated by our algorithm from a right-linear PT TRS to an equivalent TRS.

Let R be a right-linear PT TRS over F and a set X of variables. Let F_0 be a subset of defined symbols F_D and R' be $\mathcal{I}nv(R, F_0)$. R' can be transformed to a TRS R'' by the following way.

1. $R'' := \{\ l \rightarrow r \Leftarrow true \mid l \rightarrow r \Leftarrow true \in R'\ \}$.
2. For each conditional rewrite rule $f^{\#}(t) \rightarrow r \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \rightarrow r_i$ $(n \geq 1)$ in R', add the rules to R'' in the following way with a fresh defined symbol f_{new} for each rule of $f \in \mathcal{D}ep(F_0)$.

$$\begin{aligned} R'' := R'' \cup \{\ & f^{\#}(t) \rightarrow f_{new}(z_1, ..., z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n)), \\ & f_{new}(z_1, ..., z_j, r_1, ..., r_n) \rightarrow r \qquad \} \end{aligned}$$

where j is a number of elements of $\mathcal{V}ar(t) - \{y_1, ..., y_n\}$ and $\{z_1, ..., z_j\} = \mathcal{V}ar(t) - \{y_1, ..., y_n\}$

$\mathcal{D}el\mathcal{C}ond(R')$ denotes the TRS R'' obtained by the transformation above from a CTRS R'.

Example 4. Since the PT TRS R_1 of Example 1 is right-linear, the following CTRS $R_3 = \mathcal{D}el\mathcal{C}ond(\mathcal{I}nv(R_1, \{add\}))$ is obtained as follows:

$$\begin{aligned} R_3 = \ \{ \ & add^{\#}(x_2) \to tp_2(0, x_2), \\ & add^{\#}(s(y)) \to add'(add^{\#}(y)), \\ & add'(tp_2(x_1, x_2)) \to tp_2(s(x_1), x_2)) \ \}. \end{aligned}$$

□

We show that $\mathcal{D}el\mathcal{C}ond(R')$ is equivalent to R'.

Theorem 2. *Let R be a right-linear PT TRS over F and a set X of variables. Let F_0 be a subset of defined symbols F_D, R' be $\mathcal{I}nv(R,\ F_0)$ and R'' be $\mathcal{D}el\mathcal{C}ond(R')$. For all $f \in \mathcal{D}ep(F_0)$ and all $t, t_1, ..., t_m \in T(F_C, \emptyset)$, where $m = arity(f)$,*

$$f^{\#}(t) \to_{R'} tp_m(t_1, ..., t_m) \ \overset{\text{iff}}{\Longleftrightarrow} \ f^{\#}(t) \xrightarrow{*}_{R''} tp_m(t_1, ..., t_m).$$

Proof. $\Rightarrow$). We prove this direction by induction on the level k of the reduction $f^{\#}(t) \to_{R'} tp_m(t_1, ..., t_m)$. We suppose that the level of this reduction is k as follows:

$$f^{\#}(t) \underset{k}{\to}_{R'} tp_m(t_1, ..., t_m).$$

- If it is rewritten by $f^{\#}(u) \to r \in R'$, the following reduction holds by $f^{\#}(u) \to r \in R''$:
$$f^{\#}(t) \to_{R''} tp_m(t_1, ..., t_m).$$
- If it is rewritten by $f^{\#}(u) \to r \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \to r_i \in R'$, we have $t \equiv u\sigma$, $r\sigma \equiv tp_m(t_1, ..., t_m)$ for a substitution σ, and the following reduction exists:
$$f_i^{\#}(y_i)\sigma \underset{k-1}{\to}_{R'} r_i\sigma \quad (1 \le i \le n).$$
The following reduction holds by induction hypothesis.
$$f_i^{\#}(y_i)\sigma \xrightarrow{*}_{R''} r_i\sigma \quad (1 \le i \le n) \qquad (3)$$
Letting $\{z_1, ..., z_j\}$ be $\mathcal{V}ar(u) - \{y_1, ..., y_n\}$, we have $f^{\#}(u) \to f_{new}(z_1, ...z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n)) \in R''$ and $f_{new}(z_1, ..., z_j, r_1, ..., r_n) \to r \in R''$. Therefore, by these rules and the expression (3), the following reduction holds.
$$\begin{aligned} f^{\#}(t) & \equiv f^{\#}(u)\sigma \to_{R''} f_{new}(z_1, ..., z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n))\sigma \\ & \equiv f_{new}(z_1\sigma, ..., z_j\sigma, f_1^{\#}(y_1)\sigma, ..., f_n^{\#}(y_n)\sigma) \\ & \xrightarrow{*}_{R''} f_{new}(z_1\sigma, ..., z_j\sigma, r_1\sigma, ..., r_n\sigma) \\ & \equiv f_{new}(z_1, ..., z_j, r_1, ..., r_n)\sigma \to_{R''} r\sigma \equiv tp_m(t_1, ..., t_m) \end{aligned}$$

$\Leftarrow$). We prove this direction by induction on the steps k of the reduction $f^{\#}(t) \xrightarrow{*}_{R''} tp_m(t_1, ..., t_m)$. We have the following reduction:

$$f^{\#}(t) \to_{R''} t' \overset{k-1}{\to}_{R''} tp_m(t_1, ..., t_m).$$

We suppose that the rule $f^{\#}(u) \to r \in R''$ is applied at the first step.

- If r is a m-variable tuple, we have $t' \equiv tp_m(t_1, ..., t_m)$ since t' is a normal form with respect to R'. Then the following reduction holds by $f^{\#}(u) \to r \in R'$:

$$f^{\#}(t) \to_{R'} t' \equiv tp_m(t_1, ..., t_m).$$

- If $r \equiv f_{new}(z_1, ..., z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n))$, we have $f^{\#}(u) \to r_0 \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \to r_i \in R'$ and $f_{new}(z_1, ..., z_j, r_1, ..., r_n) \to r_0 \in R''$. Hence, the following reduction exists for some substitution σ.

$$\begin{aligned} f^{\#}(t) &\equiv f^{\#}(u)\sigma \ \to_{R''} f_{new}(z_1, ..., z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n))\sigma \\ &\equiv f_{new}(z_1\sigma, ..., z_j\sigma, f_1^{\#}(y_1)\sigma, ..., f_n^{\#}(y_n)\sigma) \\ &\xrightarrow{*}_{R''} f_{new}(z_1\sigma, ..., z_j\sigma, r_1\sigma, ..., r_n\sigma) \\ &\equiv f_{new}(z_1, ..., z_j, r_1, ..., r_n)\sigma \ \to_{R''} r_0\sigma \equiv tp_m(t_1, ..., t_m) \end{aligned}$$

It follows from $f_i^{\#}(y_i)\sigma \overset{k-2\geq}{\to}_{R''} r_i\sigma \ \ (1 \leq i \leq n)$ by induction hypothesis that $f_i^{\#}(y_i)\sigma \to_{R'} r_i\sigma \ \ (1 \leq i \leq n)$. Since the rule $f^{\#}(u) \to r_0 \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \to r_i \in R'$ can be applied by this reduction, the following reduction holds.

$$f^{\#}(t) \equiv f^{\#}(u)\sigma \to_{R'} r_0\sigma \equiv tp_m(t_1, ..., t_m)$$

□

Example 5. When we consider the reduction $add^{\#}(s^3(0)) \xrightarrow{*}_{R_2} tp_2(s(0), s^2(0))$, we can confirm Theorem 2 by the following reduction of R_3 of Example 4.

$$\begin{aligned} add^{\#}(s^3(0)) &\to_{R_3} add'(add^{\#}(s^2(0)) \to_{R_3} add'(tp_2(0, s^2(0))) \\ &\to_{R_3} tp_2(s(0), s^2(0)) \end{aligned}$$

□

4 Conclusion

In this paper, we proposed an algorithm for generating an inverse CTRS for a PT TRS, and proved the correctness of the algorithm. Moreover, we showed that the CTRS generated by our algorithm can be transformed to an equivalent TRS if the input PT TRS is right-linear.

It is a future work to extend the input class of our algorithm to a larger one than PT TRSs by relaxing the condition on the right-hand sides of rules. Another future work is to remove the condition on the input class of our transformation from CTRSs to equivalent TRSs. Moreover, we need to discuss the input class such that CTRSs generated by our algorithm are confluent.

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.

[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.