# 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 how to construct an inverse term rewriting system that implements the inverses of in a pure treeless TRS. At the first step, we propose an algorithm for generating a conditional TRS that computes the inverses of pure treeless functions. We prove that the conditional TRS constructed by our algorithm implements the inverses of those functions defined by a given pure treeless TRS. Next, we show that the conditional TRS generated by our algorithm can be transformed to an equivalent TRS. Moreover, if the input pure treeless TRS of the algorithm is right-linear then the resulting TRS is left-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 every time 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 presents how to construct an inverse term rewriting system that implements the inverses of in a pure treeless TRS. At the first step, we propose an algorithm that generates a conditional TRS which is the inverse of the input pure treeless TRS. In the algorithm, each rule in the input TRS is transformed to a conditional rule in such a way that the right-hand side is modified according to the idea of the fusion transformation[Chin92], the both hand sides are exchanged and then the condition is added. 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. Next, we show that the conditional TRS generated by our algorithm can be transformed to an equivalent TRS. Moreover, if the input pure treeless TRS of the algorithm is right-linear then the resulting TRS is left-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 $x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_n$ $(n \geq 0)$ or an infinite sequence $x_0 \rightarrow x_1 \rightarrow \cdots$ 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 $x \rightarrow y$, then we say $x$ is a *normal form* (with respect to $\mathcal{A}$). $\stackrel{*}{\rightarrow}$ is the reflexive and transitive closure of $\rightarrow$. $\stackrel{k}{\rightarrow}$ and $\stackrel{k\geq}{\rightarrow}$ denote a reduction of $k$ steps and a reduction of steps less than $k$, respectively. $\mathcal{A}$ is *confluent* if and only if $x \stackrel{*}{\rightarrow} y \wedge x \stackrel{*}{\rightarrow} y'$ imply $\exists z \in \mathcal{S}$, $y \stackrel{*}{\rightarrow} z \wedge y' \stackrel{*}{\rightarrow} z$ 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 $\sigma : X \rightarrow T(F, X)$ 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 $\sigma$ 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 $l \rightarrow r \Leftarrow \mathcal{C}ond$, where the left-hand side $l (\notin X)$ and the right-hand side $r$ are terms and $\mathcal{C}ond$ is either *true* or $l_1 \rightarrow r_1 \wedge \cdots \wedge l_n \rightarrow r_n$ $(n \geq 1)$. For a substitution $\sigma$ and a reduction relation $\rightarrow$ on terms, if $l_i\sigma \stackrel{*}{\rightarrow} r_i\sigma$ for all $i$ then we say that $\sigma$ and $\rightarrow$ satisfy $\mathcal{C}ond$ and write $\mathcal{C}ond(\sigma, \rightarrow)$. Letting $R$ be a set of conditional rewrite rules, $(T(F, X), \rightarrow_R)$ is called a *conditional term rewriting system* (CTRS), where $\rightarrow_R = \bigcup_{n=0}^{\infty} \underset{n}{\rightarrow}$ and $\underset{n}{\rightarrow}$, called the $n$-level reduction, are defined as follows.

- $\underset{0}{\rightarrow} = \emptyset$.
- If $s \underset{n}{\rightarrow} t$, then $s \underset{n+1}{\rightarrow} t$.

– 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 $(T(F,X), \to_R)$ if that causes no confusion. We abbreviate a rule $l \to r \Leftarrow \mathit{true}$ 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). $l \to r \Leftarrow \mathcal{C}ond$ and $l' \to r' \Leftarrow \mathcal{C}ond'$ are *overlapping* if there exist a subterm $s$ ($\not\equiv x$) of $l'$ and substitutions $\sigma$ and $\sigma'$ 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 $\{f \mid f(...) \to r \Leftarrow \mathcal{C}ond \in R\}$. 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 $R$: $\{l \to r \Leftarrow \mathcal{C}ond \mid l \equiv f(...)\}$. We introduce the notion of *tuple* as special constructors for representing the return value of the inverses of multi-argument functions. A tuple of terms $t_1, ..., t_n$ 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 $f(t_1, ..., t_n) \xrightarrow{*}_R s$ for some $s \in T(F_C, \emptyset)$.

**Definition 1.** *Let $R$ be a CTRS over a signature $F$ and a set $X$ of variables, $F_0$ be a subset of defined symbols $F_D$ and $R'$ be a CTRS over a signature $G$ and a set $X'$ of variables such that $G_C = F_C \cup \{tp_i \mid 1 \leq i \leq m\}$ and $F_D \cap G_D = \emptyset$, where $m$ 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 $g(f(t_1, ..., t_n)) \xrightarrow{*}_{R \cup R'} tp_n(t_1, ..., t_n)$.* □

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 $R$ over a signature $F$ and a set $X$ of variables is *pure treeless* if $R_f$ is in the form of either `TYPE1` or `TYPE2` for all $f \in F_D$:

– `TYPE1`

$$R_f = \{f(x_1, ..., x_n) \to tt_0\}$$

– `TYPE2`

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

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

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:

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

□

### 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, applying the inverse function symbol $f^{-1}$ of $f$ to the 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 have $x \to f^{-1}(C[f_1(x), ..., f_m(x)])$. Secondly, we exchange the both hand sides and replace the defined symbol $f_i(x)$ in the right-hand side with a fresh variable $y_i$ for each $i$. Then, we have $x \to f^{-1}(C[y_1, ..., y_m])$. Finally, 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 in order to relate $x$ to $y_i$'s. 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:

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

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.

We say that a tuple is *$n$-variable* if it is in form of either $tp_n(x_1, ..., x_n)$ or $tp_n(c(x'_1, ..., x'_j), x_2, ..., x_n)$.

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

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

□

### 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 $t_1, ..., t_n, s \in T(F_C, \emptyset)$,*

$$f(t_1, ..., t_n) \xrightarrow{*}_R s \iff f^{\#}(s) \to_{R'} tp_n(t_1, ..., t_n).$$

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

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

– 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 $C \in T(F_C \cup \{\Box\}, X)$:

$$u \equiv C[f_1(x_{1,1}, ..., x_{1,n_1}), ..., f_l(x_{l,1}, ..., x_{l,n_l})]$$

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:

$$t \equiv u\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\} \equiv C'[f_1(t_{1,1}, ..., t_{1,n_1}), ..., f_l(t_{l,1}, ..., t_{l,n_l})]$$

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 $t \equiv C'[f_1(t_{1,1}, ..., t_{1,n_1}), ..., f_l(t_{l,1}, ..., t_{l,n_l})] \xrightarrow{k-1}_R s$ 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:

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

Hence, we have the following reduction by induction hypothesis:

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

Since $f(x_1, ..., x_n) \to u \in R$ and $C$ has no defined symbol, we have:

$$\mathcal{T}[\![C[f_1(x_{1,1}, ..., x_{1,n_1}), ..., f_l(x_{l,1}, ..., x_{l,n_l})]\!] = \langle\, C[y_1, ..., y_l]\, ;\, Cond\,\rangle$$

where $Cond = \bigwedge_{i=1}^{l} (f_i^{\#}(y_i) \to tp_{n_i}(x_{i,1}, ..., x_{i,n_i})\,)$. 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 $\sigma$ be $\{x_1 \mapsto t_1, ..., x_n \mapsto t_n, y_1 \mapsto s_1, ..., y_l \mapsto s_l\}$. Then we have $Cond(\sigma, \underset{m}{\to}_{R'})$. Therefore, the following reduction exists:

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

– 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 of the reduction $f^{\#}(s) \to_{R'} tp_n(t_1, ..., t_n)$. We suppose that the level of this reduction is $k$ as follows:

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

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

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

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 $f(x_1, ..., x_n) \to u \in R$ by the definition of $\mathcal{I}nv(R, F_0)$. Therefore, the following reduction holds:
  $$f(t_1, ..., t_n) \equiv f(x_1, ..., x_n)\sigma \to_R u\sigma \equiv s.$$
  - If $Cond = \bigwedge_{i=1}^{m} f_i^{\#}(y_i) \to tp_{n_i}(x_{i,1}, ..., x_{i,n_i})$ 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 $f(x_1, ..., x_n) \to C[f_1(x_{1,1}, ..., x_{1,n_1}), ..., f_m(x_{m,1}, ..., x_{m,n_m})] \in R$ by the definition of $\mathcal{I}nv(R, F_0)$. From $Cond(\sigma, \to_{R'})$ and the form of rules in $R'$, the following reduction holds:
  $$f^{\#}(y_i\sigma) \underset{k_i}{\to}_{R'} tp_{n_i}(x_{i,1}\sigma, ..., x_{i,n_i}\sigma) \quad (1 \leq i \leq m).$$
  Hence, we have the following reduction by induction hypothesis:
  $$f_i(x_{i,1}\sigma, ..., x_{i,n_i}\sigma) \overset{*}{\to}_R y_i\sigma \quad (1 \leq i \leq m).$$
  Therefore, the following reduction holds:
  $$\begin{aligned} f(t_1, ..., t_n) &\equiv f(x_1, ..., x_n)\sigma \\ &\to_R C[f_1(x_{1,1}, ..., x_{1,n_1}), ..., f_m(x_{m,1}, ..., x_{m,n_m})]\sigma \\ &\equiv C[f_1(x_{1,1}, ..., x_{1,n_1})\sigma, ..., f_m(x_{m,1}, ..., x_{m,n_m})\sigma] \\ &\equiv C[f_1(x_{1,1}\sigma, ..., x_{1,n_1}\sigma), ..., f_m(x_{m,1}\sigma, ..., x_{m,n_m}\sigma)] \\ &\overset{*}{\to}_R C[y_1\sigma, ..., y_m\sigma] \equiv C[y_1, ..., y_m]\sigma \equiv u\sigma \equiv s. \end{aligned}$$

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

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.

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

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

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

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

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 $f^{\#}(t) \to s$ and $f^{\#}(t) \to s'$ 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 PT TRS to an equivalent TRS, and if the input PT TRS is right-linear then the resulting TRS is left-linear.

**Proposition 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 $F_D$ and $R'$ be $\mathcal{I}nv(R, F_0)$. For any conditional rewrite rule $f^{\#}(t) \to r \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \to r_i \in R'$ such that $f, f_i \in \mathcal{D}ep(F_0)$, $t \in T(F_C, X)$, $r$ and $r_i$ are $m$-variable tuple ($m = arity(f)$) and $m_i$-variable tuple ($m_i = arity(f_i)$), respectively, the following property holds.*

- $\forall i,\ y_i \in \mathcal{V}ar(t),\ y_i \notin \mathcal{V}ar(r),\ \forall j,\ y_i \notin \mathcal{V}ar(r_j)$.

*Moreover, if $R$ is right-linear then the following two properties are also satisfied.*

- $\mathcal{V}ar(t) \cap \bigcup_{i=1}^{n} \mathcal{V}ar(r_i) = \emptyset$.
- $\forall i, j,\ i \neq j \Rightarrow \mathcal{V}ar(r_i) \cap \mathcal{V}ar(r_j) = \emptyset$.

*Proof.* It is trivial from the construction of $R'$ and that right-linearity of $R$. □

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 $\mathcal{I}nv(R, F_0)$. Then, $R'$ can be transformed to a TRS $R''$ by the following way.

1. $R'' := \{\ l \to r \mid l \to r \Leftarrow true \in R'\ \}$.

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

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

where $\{z_1, ..., z_j\} = \mathcal{V}ar(t) - \{y_1, ..., y_n\}$.

$\mathcal{D}el\mathcal{C}ond(R')$ denotes the TRS $R''$ obtained from a CTRS $R'$ by the transformation above. Note that $\mathcal{D}el\mathcal{C}ond(R')$ is left-linear if $R$ is right-linear by Proposition 2.

*Example 4.* Since the PT TRS $R_1$ of Example 1 is right-linear, we obtain the following left-linear CTRS $R_3 = \mathcal{D}el\mathcal{C}ond(R_2)$ from $R_2$ of Example 2:

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

□

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 a signature $F$ 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 $m = arity(f)$,*

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

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

- 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 some substitution $\sigma$, and also have the following reduction:
$$f_i^{\#}(y_i)\sigma \underset{k-1}{\to}_{R'} r_i\sigma \quad (1 \leq i \leq n).$$
Hence, the following reduction holds by induction hypothesis.
$$f_i^{\#}(y_i)\sigma \overset{*}{\to}_{R''} r_i\sigma \quad (1 \leq i \leq n) \tag{3}$$

We have $f^{\#}(u) \to f_{new}(z_1, ...z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n)) \in R''$, $f_{new}(z_1, ..., z_j, r_1, ..., r_n) \to r \in R''$ from the construction of $R''$ where $\{z_1, ..., z_j\} = \mathcal{V}ar(u) - \{y_1, ..., y_n\}$. Therefore, by these rules and the expression (3), the following holds:

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

$\Leftarrow$). We prove this direction by induction on the steps $k$ of the reduction $f^{\#}(t) \overset{*}{\to}_{R''} tp_m(t_1, ..., t_m)$. 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 applied at the first step is $f^{\#}(u) \to r \in R''$.

- If $f^{\#}(u) \to r \in R'$ then $r$ is an $m$-variable tuple. Since $t'$ is a normal form with respect to $R'$, we have $t' \equiv tp_m(t_1, ..., t_m)$. Therefore, we have $f^{\#}(t) \to_{R'} t' \equiv tp_m(t_1, ..., t_m)$.
- 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 $f_{new}(z_1, ..., z_j, r_1, ..., r_n) \to r_0 \in R''$ is the only rule having $f_{new}$ as the top symbol. Hence, we have the following reduction for some substitution $\sigma_1$ and $\sigma_2$ such that $\mathcal{D}om(\sigma_1) = \{z_1, ..., z_j, y_1, ..., y_n\}$ and $\mathcal{D}om(\sigma_2) = \{z_1, ..., z_j\} \cup \bigcup_{i=1}^{n} \mathcal{V}ar(r_i)$ without loss of generality.

$$\begin{aligned} f^{\#}(t) &\equiv f^{\#}(u)\sigma_1 \to_{R''} f_{new}(z_1, ..., z_j, f_1^{\#}(y_1), ..., f_n^{\#}(y_n))\sigma_1 \\ &\equiv f_{new}(z_1\sigma_1, ..., z_j\sigma_1, f_1^{\#}(y_1\sigma_1), ..., f_n^{\#}(y_n\sigma_1)) \\ &\overset{*}{\to}_{R''} f_{new}(z_1\sigma_2, ..., z_j\sigma_2, r_1\sigma_2, ..., r_n\sigma_2) \\ &\equiv f_{new}(z_1, ..., z_j, r_1, ..., r_n)\sigma_2 \to_{R''} r_0\sigma_2 \equiv tp_m(t_1, ..., t_m) \end{aligned}$$

We have $z_i\sigma_1 \equiv z_i\sigma_2$ since $z_i\sigma_1$ is a normal form for each $i$. We also have $(\{y_1, ..., y_n\} \cap \bigcap_{i=1}^{n} \mathcal{V}ar(r_i)) = \emptyset$ by Proposition 2. Hence we can define $\sigma = \sigma_1\sigma_2$. Then we have $y_i\sigma \equiv y_i\sigma_1$, $r_i\sigma \equiv r_i\sigma_2$, $u\sigma \equiv u\sigma_1$ and $r_0\sigma \equiv r_0\sigma_2$ for each $i$. Moreover, it follows from $f_i^{\#}(y_i\sigma) \overset{k-2\geq}{\to}_{R''} r_i\sigma$ by induction hypothesis that $f_i^{\#}(y_i\sigma) \to_{R'} r_i\sigma$ for each $i$. Therefore, it follows from $f^{\#}(u) \to r_0 \Leftarrow \bigwedge_{i=1}^{n} f_i^{\#}(y_i) \to r_i \in R'$ that $f^{\#}(t) \equiv f^{\#}(u)\sigma \to_{R'} r_0\sigma \equiv tp_m(t_1, ..., t_m)$. □

*Example 5.* When we consider the reduction $add^{\#}(s^3(0)) \overset{*}{\to}_{R_2} tp_2(s(0), s^2(0))$, 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. Next, we showed that the CTRS generated by our algorithm can be transformed to an equivalent TRS, and if the input PT TRS is right-linear then the resulting TRS is left-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. 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. ACM Conference on Lisp and Functional Programming, San Francisco, Ca., pp.11-20, ACM, June 1992.

[Der99] N. Dershowitz, S. Mitra: Jeopardy. The 10th International Conference on Rewriting Techniques and Applications, LNCS 1631, pp.16-29, 1999.

[Klop92] J.W.Klop: Term Rewriting Systems. 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.