PT関数の逆関数を定義する条件付き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

nishida@sakabe.nuie.nagoya-u.ac.jp
sakai@nuie.nagoya-u.ac.jp
sakabe@nuie.nagoya-u.ac.jp

あらまし　本稿では、パターン照合の機能を持ち、右辺に入れ子の関数呼び出しを持たない pure treeless 関数定義から指定された関数の逆関数の定義を持つ条件付き項書換え系を生成するアルゴリズムを提案する。

本アルゴリズムでは、関数 f の逆関数 f' を求める際に、f を定義するすべての規則に対し、左辺と右辺を入れ換えた後、フュージョン変換の考え方を応用して、関数の定義規則として成立するように規則を変更する。提案したアルゴリズムにより得られた関数が逆関数であることを証明する。

キーワード　逆関数、プログラム変換、関数型プログラム

Generation of a Conditional TRS Implementing the Inverses of Pure Treeless Functions

Naoki Nishida　Masahiko Sakai　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, where pure treeless functions are those which are defined by a pure treeless definition, i.e., rewriting rules such that patterns are allowed only in the first argument of the left hand side of a rule and all arguments of any defined symbol are variables in the right hand side of a rule.

Our algorithm constructs conditional rules from a pure treeless definition in such a way that each rule of the definition 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 definition.

key words　inverse function, program transformation, functional programming

1 はじめに

方程式を解く、すなわち、未知変数の解を求める問題は、方程式に現れる変数への代入を求める問題と言い換えることができる。これを解くもっとも単純な方法は、変数に順に値を代入し、方程式が成り立つか調べる方法である。代入する値は、成り立っている関係式などから予想することはできるが、あらゆる値について順に調べなければならない。このような代入を求める方法として、関数型言語には、“E-unification”[BN98, Plot72] や “Narrowing”[BN98, Sla74, Lank75]、“Inversion Algorithm”[Der99] などの方法がある。これらの方法は、規則集合と計算結果を与えることによって代入を求めるアルゴリズムであり、解を求める際に、毎回アルゴリズムが実行される。また、計算に対して適用されうるあらゆる規則から代入の可能性のある値を取り出していく。そのため、これらの方法は効率的であるとはいい難いが、2 つ以上の未知変数に対してもあらゆる解を求めることができる。

方程式を解く方法として、与えられた方程式を $x = \cdots$ の形式に変形して解を得る方法が一般的である。その際の両辺の変形は逆向き（逆関数）の計算を行っていると言える。例えば、$x + 3 = 5$ という方程式を解く場合、両辺から 3 を引く。これは、+3 という計算に対する逆向きの計算をすることになる。この場合は、-3 という $+3$ の逆関数を両辺に施して式を変形している。その結果、$x = 2$ という解が得られる。一般に、関数が現れる線形方程式は、その関数の逆関数があれば、それを利用して効率的に解くことができる。例えば、$f(x) = n$ の場合には、関数 f の逆関数 f^{-1} を両辺に施して、解は $f^{-1}(f(x)) = x = f^{-1}(n)$ となる。

関数型言語においても逆関数を利用できれば、あらかじめ逆関数を求めておき、それを利用して簡単に何度でも計算を行える。先の例 $add(x, 3) = 5$ では、+3 の関数を add_3 とすれば、$add_3(x) = 5$ と表現でき、+3 の逆関数 -3 を add_3^{-1} とすると、両辺に add_3^{-1} を施して、$add_3^{-1}(add_3(x)) = add_3^{-1}(5)$ となる。両辺を計算すると、$x = 2$ が得られる。逆関数を求めることができればこのような計算により方程式を解くことが可能となる。

本稿では、パターン照合の機能を持ち、規則の右辺に入れ子の関数呼び出しがない pure treeless 関数定義 [Chin92] を対象として、関数の効率化を行うフュージョン変換と呼ばれるプログラム変換の考え方を発展させて、プログラム変換により、関数型プログラムから指定された関数の逆関数を持つ条件付き項書換え系を生成するアルゴリズムを提案し、このアルゴリズムが必ず停止し、条件付き項書換え系を出力することを示す。また、提案したアルゴリズムにより得られた関数がその逆関数になっていることを証明する。

2 準備

本稿では、項書換え系に関する一般的な記法に従う [BN98, Klop92]。

抽象書換え系は $\mathcal{A} = (\mathcal{S}, \rightarrow)$ である。ここで、$\mathcal{S}$ はある集合であり、$\rightarrow$ は $\mathcal{S}$ 上の 2 項関係（書換え関係）である。書換えとは、書換えステップにおける有限のシーケンス $x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_n$ $(n \geq 0)$ 、あるいは、無限のシーケンス $x_0 \rightarrow x_1 \rightarrow \cdots$ である。$x, y \in \mathcal{S}$ が同一であるとき、$x \equiv y$ と書く。$\rightarrow$ の推移反射閉包を $\xrightarrow{*}$ と書く。$\xrightarrow{k}$ 、$\xrightarrow{k\geq}$ はそれぞれ k ステップの書換え、k ステップ以下の書換えを表す。任意の x , y , $y' \in \mathcal{S}$ において、$x \xrightarrow{*} y$ かつ $x \xrightarrow{*} y'$ ならば、$z \in \mathcal{S}$ が存在して $y \xrightarrow{*} z$ かつ $y' \xrightarrow{*} z$ が成立するとき、$\mathcal{A}$ は合流性を持つという。

関数記号、変数の集合をそれぞれ F 、X とする。$F \cup X$ の記号から構成される項は写像 $arity : F \rightarrow N$（N は自然数の集合）を用いて、次のように再帰的に定義される。

- 変数 x $(\in X)$ は項である。
- $f \in F$, $arity(f) = n$ で $t_1, ..., t_n$ が項のとき、$f(t_1, ..., t_n)$ は項である。ただし、$arity(f) = 0$ のときは f を項とする。

$arity(f) = 0$ である関数記号 f を定数と呼ぶ。項の集合を $T(F \cup X)$（単に T）で表す。また、項 t に出現する変数の集合を $\mathcal{V}ar(t)$ で表す。一般に n 個の関数記号 f の入れ子 $\overbrace{f(\cdots f}^{n}(t)\cdots)$ を $f^n(t)$ と書く。

$\Box$ を特別な定数とする。$C[, ...,]$ と記述される項 $C \in T(F \cup X \cup \{\Box\})$ を文脈と呼ぶ。n 個の $\Box$ を含む $C[, ...,]$ と $t_1, ..., t_n \in T(F \cup X)$ に対して、$\Box$ を左から順に $t_1, ..., t_n$ に置き換えて得られる項を $C[t_1, ..., t_n]$ で表す。$\Box$ が 1 個の文脈 C を $C[\ ]$ と書く。

代入は変数から項への写像である。変数 $x_1, ..., x_n$ をそれぞれ項 $u_1, ..., u_n$ に移す代入 σ を $\sigma = \{x_1 \mapsto u_1, ..., x_n \mapsto u_n\}$ と表す。項 t 中の各変数 x_i を項 u_i に置き換えて得られる項を $t\sigma$ で表す。

$l \to r \Leftarrow \mathcal{C}ond$ は条件付き書換え規則と呼ばれる。ここで、$l \notin X$ と r は項、$\mathcal{C}ond$ は $true$ もしくは $l_1 \to r_1 \wedge \cdots \wedge l_n \to r_n$ $(n \geq 1)$ の形式であり、任意の i $(1 \leq i \leq n)$ について、$\mathcal{V}ar(l) \supseteq \mathcal{V}ar(l_i)$, $\mathcal{V}ar(l) \cup \bigcup_{i=1}^{n} \mathcal{V}ar(r_i) \supseteq \mathcal{V}ar(r)$ を満たす。代入 σ、項上の関係 $\to$ に対して、すべての i について $l_i\sigma \xrightarrow{*} r_i\sigma$ ならば、σ と $\to$ は $\mathcal{C}ond$ を満たすといい、$\mathcal{C}ond(\sigma, \to)$ と書く。条件付き書換え規則の集合を R で表し、$(T(F \cup X), \underset{R}{\longrightarrow})$ を**条件付き項書換え系**（CTRS）と呼ぶ。以下では混乱がない限り、CTRS $(T(F \cup X), \underset{R}{\longrightarrow})$ を条件付き書換え規則の集合 R のみで表す。ここで、$\underset{R}{\longrightarrow}$ は次のように定義されるレベル n の書換え $\underset{n}{\longrightarrow}$ を用いて $\underset{R}{\longrightarrow} = \bigcup_{n=0}^{\infty} \underset{n}{\longrightarrow}$ と定義される。

- $\underset{0}{\longrightarrow} = \emptyset$。
- $s \underset{n}{\longrightarrow} t$ ならば、$s \underset{n+1}{\longrightarrow} t$。
- $l \to r \Leftarrow \mathcal{C}ond \in R$ かつ $\mathcal{C}ond(\sigma, \underset{n}{\longrightarrow})$ ならば、$C[l\sigma] \underset{n+1}{\longrightarrow} C[r\sigma]$。

特に、条件付き書換え規則 $l \to r \Leftarrow true$ は、単に $l \to r$ と書くことがあり、これを書換え規則と呼ぶ。R が書換え規則のみからなる CTRS を項書換え系（TRS）と呼ぶ。TRS においては、任意の $i \geq 1$ について、$\underset{R}{\longrightarrow} = \underset{i}{\longrightarrow}$ が成り立つ。

3 逆関数の生成

3.1 Pure Treeless 関数

関数記号の集合 F は、被定義記号の集合 F_D と構成子記号の集合 F_C に分割されているものとする。すなわち、$F = F_D \oplus F_C$ とする。本稿では、被定義記号として f, g, h、構成子記号として c, d, e、変数として x, y, z を使用する。

次のように再帰的に定義される項を **pure treeless 項**（PT 項）と呼ぶ。

$$PT\text{項} ::= \text{変数} \mid c(PT\text{項}, ..., PT\text{項}) \mid f(\text{変数}, ..., \text{変数})$$

PT 項中の被定義記号の引数はすべて変数であるので、PT 項は被定義記号が現れない文脈 $C \in T(F_C \cup X \cup \{\Box\})$ を用いて、$C[f_1(\text{変数}, ..., \text{変数}), ..., f_n(\text{変数}, ..., \text{変数})]$ と表現できる。

変数、もしくは、1 つの構成子記号と変数からなる項をパターン項と呼び、次のように定義する。

$$\text{パターン項} ::= \text{変数} \mid c(\text{変数}, ..., \text{変数})$$

pure treeless 関数定義（PT 関数定義）は次のタイプ 1 もしくはタイプ 2 の形式の書換え規則の集合である [1]。

- タイプ 1
$$f(x_1, ..., x_n) \to t$$
- タイプ 2
$$g(p_1, x_1, ..., x_n) \to t_1$$
$$\vdots$$
$$g(p_m, x_1, ..., x_n) \to t_m$$

ここで、$t, t_1, ..., t_n$ は PT 項、$p_1, ..., p_m$ はパターン項である。このように、PT 関数定義は制限付きの TRS である。PT 関数定義で定められる被定義記号を PT 関数と呼ぶ。

例 1 自然数を *2* 倍する *PT* 関数 $double$ は次のように定義される。

$$R_1 = \{ \; double(0) \to 0, \quad double(s(x)) \to s(s(double(x))) \; \}$$

□

3.2 逆関数生成の基本となる考え方

例 1 で定義した $double$ の逆関数 $double^{-1}$ は、引数が偶数であるとき、その半分の値を返す。例えば、$double^{-1}$ は次のように定義できる。

$$R_2 = \{ \; double^{-1}(0) \to 0, \quad double^{-1}(s(s(x))) \to s(double^{-1}(x)) \; \}$$

[1] 文献 [Chin92] では、タイプ 1 を pure treeless 関数、タイプ 2 を g-type パターン照合関数と呼んでいるが、本稿ではこれらをまとめて pure treeless 関数と呼ぶ。

R_1 の各々の規則から R_2 の規則を生成する方法を考える。

$double(0) \to 0$ については、両辺に $double^{-1}$ を施すと、逆関数の一般的性質「 $f^{-1}(f(n)) = n$ 」から、左辺が 0 となるので、左辺と右辺を入れ換えて、

$$double^{-1}(0) \to 0$$

が得られる。

$double(s(x)) \to s(s(double(x)))$ については、$double(x) = y$ と置いて、両辺に $double^{-1}$ を施すと、右辺からは、$double^{-1}(s(s(y)))$ が得られる。左辺からは、逆関数の性質により、$s(x)$ が得られる。さらに、$double(x) = y$ の両辺に $double^{-1}$ を施すことによって、$x = double^{-1}(y)$ が得られるので、左辺は $s(double^{-1}(y))$ と書ける。したがって、両辺を入れ換えることによって、

$$double^{-1}(s(s(y))) \to s(double^{-1}(y))$$

となり、R_2 の各規則が得られる。

次に、1 引数のタイプ 1 で定義される PT 関数の一般的な規則

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

について考える。ここで、$C \in T(F_C \cup X \cup \{\Box\})$ とする。

まず、すべての右辺の $f_i(x)$ $(1 \le i \le m)$ をそれぞれ新しい変数 y_i $(1 \le i \le m)$ に置き換えて、両辺に f^{-1} を施して変形し、左右を入れ換えると、

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

が得られる。このとき、$x = f_1^{-1}(y_1)$, ..., $x = f_m^{-1}(y_m)$ であるので、$double$ の例のように x を $y_1, ..., y_m$ を用いて表せない。そこで、これらを規則の条件とすることにより、条件付き書換え規則として逆関数の規則

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

が得られる。

3.3 アルゴリズム

まず、関数 f の計算に必要となる関数の集合を定義する。

定義 1 PT 関数定義 $(T((F_D \oplus F_C) \cup X), \underset{R}{\longrightarrow})$、関数記号の集合 $F_0 \subseteq F_D$ について、F_0 中の関数が依存する関数の集合 G は、次のように再帰的に定義される。

1. $G := F_0$
2. $G := G \cup \{h \in F_D \mid$

$$f(...) \to C[..., h(x_1, ..., x_n), ...] \in R, f \in G\}$$

このとき G を $\mathcal{D}ep(F_0)$ と書く。 □

以下では、3.2 節で考案した手法をアルゴリズム $\mathcal{I}nv$ として形式化する。ここで、多引数関数の逆関数は項のタプルを返すので、項 $t_1, ..., t_n$ のタプルを導入し、$(t_1, ..., t_n)$ と書くことにする。

アルゴリズム $\mathcal{I}nv$ は、PT 関数定義 R、逆関数を求めたい関数記号の集合 $F_0 \subseteq F_D$ から、CTRS を生成する。

$$\begin{aligned}\mathcal{I}nv(R, F_0) \quad = \quad & \\ \{\ & h^{\#}(s) \to (p, x_1, ..., x_n) \ \Leftarrow \ COND \\ \mid \quad & h(p, x_1, ..., x_n) \to t \in R, \\ & h \in \mathcal{D}ep(F_0), \\ & \mathcal{I}nv\mathcal{R}ule(t) = \langle\ s; COND\ \rangle\ \}\end{aligned}$$

ここで、$\mathcal{I}nv\mathcal{R}ule$ は、PT 項 t を入力とし、項 s と書換え条件 $COND$ の並び、$\langle\ s; COND\ \rangle$ を出力する手続きである。s は、$\mathcal{C}nd := true$ を初期値として、$\mathcal{T}[\![t]\!]$ に $\mathcal{T}[\![...]\!]$ が消えるまで次の変換を適用して得られる項である。また、このときの $\mathcal{C}nd$ の値が $COND$ である。

- $t \equiv x$ のとき、

$$\mathcal{T}[\![x]\!] \ \Rightarrow \ x$$

- $t \equiv c(t_1, ..., t_n)$ のとき、

$$\mathcal{T}[\![c(t_1, ..., t_n)]\!] \ \Rightarrow \ c(\mathcal{T}[\![t_1]\!], ..., \mathcal{T}[\![t_n]\!])$$

- $t \equiv f(x_1, ..., x_n)$ のとき、

$$\begin{cases} \mathcal{T}[\![f(x_1, ..., x_n)]\!] \ \Rightarrow \ y \\ \\ \mathcal{C}nd := \mathcal{C}nd \wedge f^{\#}(y) \to (x_1, ..., x_n) \end{cases}$$

ここで、y は新しい変数とする。

条件付き書換え規則の集合 $\mathcal{I}nv(R, F_0)$ が有限であること、手続き $\mathcal{I}nv\mathcal{R}ule$ における $\mathcal{T}$ の計算が必ず停止することは明らかでる。

例 2 例 *1* の *PT* 関数定義 R_1 に対して $double$ の逆関数を求める。

まず、$F_0 = \{double\}$ とすると、被定義記号は $double$ のみであるので、逆関数を求める必要のある関数記号の集合は $\mathcal{D}ep(F_0) = \{double\}$ となる。

規則 $double(0) \to 0$ については、$\mathcal{I}nv\mathcal{R}ule(0) = \langle\ 0; true\ \rangle$ であるので、規則

$$double^{\#}(0) \to 0$$

が得られる。

また、規則 $double(s(x)) \to s(s(double(x)))$ については、

$$\mathcal{I}nv\mathcal{R}ule(s(s(double(x))))$$
$$= \langle\ s(s(y)); double^{\#}(y) \to x\ \rangle$$

であるから、規則

$$double^{\#}(s(s(y))) \to s(x) \Leftarrow double^{\#}(y) \to x$$

が得られる。

よって、次の *CTRS* R_3 が得られる。

$$\begin{aligned} R_3 = \{\ & double^{\#}(0) \to 0, \\ & double^{\#}(s(s(y))) \to s(x) \\ & \qquad \Leftarrow double^{\#}(y) \to x\ \} \end{aligned}$$

□

例 3 *PT* 関数定義 R_4 に対して加算 add の逆関数を求める。

$$\begin{aligned} R_4 = \{\ & add(0, x_2) \to x_2, \\ & add(s(x_1), x_2) \to s(add(x_1, x_2))\ \} \end{aligned}$$

まず、$F_0 = \{add\}$ とすると、被定義記号は add のみであるので、逆関数を求める必要のある関数記号の集合は $\mathcal{D}ep(F_0) = \{add\}$ となる。

規則 $add(0, x_2) \to x_2$ については、$\mathcal{I}nv\mathcal{R}ule(x_2) = \langle\ x_2; true\ \rangle$ であるので、規則

$$add^{\#}(x_2) \to (0, x_2)$$

が得られる。

また、規則 $add(s(x_1), x_2) \to s(add(x_1, x_2))$ については、

$$\mathcal{I}nv\mathcal{R}ule(s(add(x_1, x_2)))$$
$$= \langle\ s(y); add^{\#}(y) \to (x_1, x_2)\ \rangle$$

であるから、規則

$$add^{\#}(s(y)) \to (s(x_1), x_2) \Leftarrow add^{\#}(y) \to (x_1, x_2)$$

が得られる。

よって、次の *CTRS* R_5 が得られる。

$$\begin{aligned} R_5 = \{\ & add^{\#}(x_2) \to (0, x_2), \\ & add^{\#}(s(y)) \to (s(x_1), x_2) \\ & \qquad \Leftarrow add^{\#}(y) \to (x_1, x_2)\ \} \end{aligned}$$

□

3.4 アルゴリズムの正当性

まず、$\mathcal{D}ep(F_0)$ に属する関数の定義の左辺が 1 変数のみである場合のアルゴリズムの正当性を証明する。

命題 1 R を *PT* 関数定義とする。このとき、すべての $\mathcal{D}ep(F_0)$ に含まれる f に対して、$f(...) \to r \in R$ ならば、r 中の被定義記号は $\mathcal{D}ep(F_0)$ に属する。

証明 $\mathcal{D}ep(F_0)$ の作り方より明らか。 □

定理 1 R を *PT* 関数定義、$R' = \mathcal{I}nv(R, F_0)$ 、$\mathcal{D}ep(F_0)$ のすべての関数の定義の左辺が *1* 変数であるとする。このとき、任意の $f \in \mathcal{D}ep(F_0)$ と $t, s \in T(F_C)$ について、$f(t) \xrightarrow[R]{*} s$ ならば、$f^{\#}(s) \xrightarrow[R']{*} t$ である。

証明 書換え系列 $f(t) \xrightarrow[R]{k} s$ のステップ数 k に関する帰納法により証明する。

$k \geq 1$ であるので、この系列は、

$$f(t) \longrightarrow_R t' \xrightarrow[R]{k-1} s$$

と書ける。

- f がタイプ 1 の形式で定義された PT 関数のとき、$f(x) \to u \in R$ 、かつ $t' \equiv u\{x \mapsto t\}$。$u$ は PT 項

であるから、文脈 $C \in T(F_C \cup X \cup \{\Box\})$ を用いて、$u \equiv C[f_1(x), ..., f_n(x)]$ と表現できる。ここで、命題 1 より、各 i について f_i は $\mathcal{D}ep(F_0)$ の要素である。よって、

$$t' \equiv u\{x \mapsto t\} \equiv C'[f_1(t), ..., f_n(t)]$$

ここで、$C' \equiv C\{x \mapsto t\}$ である。

さらに、$t' \equiv C'[f_1(t), ..., f_n(t)] \xrightarrow[R]{k-1} s$ かつ C' は F_D の関数記号を持たないので、$s \equiv C'[s_1, ..., s_n]$ と書けて、

$$f_1(t) \xrightarrow[R]{k-1\geq} s_1, \quad ... \quad , f_n(t) \xrightarrow[R]{k-1\geq} s_n$$

である。ゆえに、帰納法の仮定より、

$$f_1^{\#}(s_1) \xrightarrow[R']{*} t, \quad ..., \quad f_n^{\#}(s_n) \xrightarrow[R']{*} t \qquad (1)$$

が得られる。

また、$f(x) \to C[f_1(x), ..., f_n(x)] \in R$, $C \in T(F_C \cup X \cup \{\Box\})$ であるから、

$$\mathcal{I}nv\mathcal{R}ule(C[f_1(x), ..., f_n(x)])$$
$$= \langle\, C[y_1, ..., y_n]; COND \,\rangle$$

となる。ここで、$COND = \bigwedge_{i=1}^{n} (f_i^{\#}(y_i) \to x\ )$ である。したがって、R' に次の規則

$$f^{\#}(C[y_1, ..., y_n]) \to x \ \Leftarrow \ COND$$

が含まれる。

式 (1) の書換えの最大のレベルを m とし、$\sigma = \{ x \mapsto t,\ y_1 \mapsto s_1,\ ...,\ y_n \mapsto s_n\}$ とすると、$COND(\sigma, \xrightarrow[m\ R']{}\ )$ が成り立つ。よって、

$$\begin{aligned} f^{\#}(s) &\equiv f^{\#}(C'[s_1, ..., s_n]) \\ &\equiv f^{\#}(C[y_1, ..., y_n])\sigma \\ &\xrightarrow[m+1_{R'}]{} x\sigma \ \equiv \ t \end{aligned}$$

となる。

- f がタイプ 2 の形式で定義された PT 関数のとき、$t \equiv c(t'')$ と書けて、また、$f(c(x)) \to u \in R$、かつ $t' \equiv u\{x \mapsto t''\}$ である。u は PT 項であるから、文脈 $C \in T(F_C \cup X \cup \{\Box\})$ を用いて、$u \equiv C[f_1(x), ..., f_n(x)]$ と表現できる。ここで、命題 1 より、各 i について f_i は $\mathcal{D}ep(F_0)$ の要素である。よって、

$$t' \equiv u\{x \mapsto t''\} \equiv C'[f_1(t''), ..., f_n(t'')]$$

ここで、$C' \equiv C\{x \mapsto t''\}$ である。

さらに、$t' \equiv C'[f_1(t''), ..., f_n(t'')] \xrightarrow[R]{k-1} s$ であるので、$s \equiv C'[s_1, ..., s_n]$ と書けるので、

$$f_1(t'') \xrightarrow[R]{k-1\geq} s_1, \quad ..., \quad f_n(t'') \xrightarrow[R]{k-1\geq} s_n$$

である。ゆえに、帰納法の仮定より、

$$f_1^{\#}(s_1) \xrightarrow[R']{*} t'', \quad ..., \quad f_n^{\#}(s_n) \xrightarrow[R']{*} t'' \qquad (2)$$

が得られる。

また、$f(c(x)) \to C[f_1(x), ..., f_n(x)] \in R$, $C \in T(F_C \cup X \cup \{\Box\})$ であるから、

$$\mathcal{I}nv\mathcal{R}ule(C[f_1(x), ..., f_n(x)])$$
$$= \langle\, C[y_1, ..., y_m]; COND \,\rangle$$

となる。ここで、$COND = \bigwedge_{i=1}^{n} (f_i^{\#}(y_i) \to x)$ である。したがって、R' に次の規則

$$f^{\#}(C[y_1, ..., y_n]) \to c(x) \ \Leftarrow \ COND$$

が含まれる。

式 (2) の書換えの最大のレベルを m とし、$\sigma = \{ x \mapsto t'',\ y_1 \mapsto s_1,\ ...,\ y_n \mapsto s_n\}$ とすると、$COND(\sigma, \xrightarrow[m\ R']{}\ )$ が成り立つ。よって、

$$\begin{aligned} f^{\#}(s) &\equiv f^{\#}(C'[s_1, ..., s_n]) \\ &\equiv f^{\#}(C[y_1, ..., y_n])\sigma \\ &\xrightarrow[m+1_{R'}]{} c(x)\sigma \ \equiv c(t'') \ \equiv \ t \end{aligned}$$

となる。

以上により、定理は成り立つ。 □

例 4 例 *2* で R_1 から生成した R_3 について考える。
$double(s^2(0)) \xrightarrow[R_1]{*} s^4(0)$ を考えたとき、定理 *1* より、

$$double^{\#}(s^4(0)) \xrightarrow[R_3]{*} s^2(0)$$

である。実際、$double^{\#}(0) \underset{R_3}{\longrightarrow} 0$ より、規則

$$double^{\#}(s(s(y))) \to s(x) \Leftarrow double^{\#}(y) \to x$$

が適用できるので、

$$double^{\#}(s^2(0)) \underset{R_3}{\longrightarrow} s(0)$$

が示せる。さらに、同様にして、

$$double^{\#}(s^4(0)) \underset{R_3}{\longrightarrow} s^2(0)$$

が導かれる。 □

直交な TRS は合流性を持つことが知られている [BN98]。したがって、PT 関数は合流性を持つ。しかしながら、アルゴリズムによって得られた CTRS は合流性を持つとは限らない。実際、多対 1 関数では、$s \neq s'$ かつ $s, s' \in T(F_C)$ について、$f(s) \equiv f(s')\ (\equiv t)$ であるが、定理 1 より、$f^{\#}(t) \overset{*}{\longrightarrow} s$ かつ $f^{\#}(t) \overset{*}{\longrightarrow} s'$ となり、合流性を持たない。

次に、多引数関数についても、アルゴリズムにより得られた CTRS が逆関数になっていることを証明する。

定理 2 R を PT 関数定義、$R' = \mathcal{I}nv(R, F_0)$ であるとする。このとき、任意の $f \in \mathcal{D}ep(F_0)$ と $t_1, ..., t_n, s \in T(F_C)$ について、$f(t_1, ..., t_n) \underset{R}{\overset{*}{\longrightarrow}} s$ ならば、$f^{\#}(s) \underset{R'}{\overset{*}{\longrightarrow}} (t_1, ..., t_n)$ である。

証明 書換え系列 $f(t_1, ..., t_n) \underset{R}{\overset{k}{\longrightarrow}} s$ のステップ数 k に関する帰納法により証明する。

$k \geq 1$ であるので、この系列は、

$$f(t_1, ..., t_n) \underset{R}{\longrightarrow} t \underset{R}{\overset{k-1}{\longrightarrow}} s$$

と書ける。

- f がタイプ 1 の形式で定義された PT 関数のとき、$f(x_1, ..., x_n) \to u \in R$ 、かつ $t \equiv u\{x_1 \mapsto t_1, ...x_n \mapsto t_n\}$。$u$ は PT 項であるから、文脈 $C \in T(F_C \cup X \cup \{\Box\})$ を用いて、$u \equiv C[f_1(x_{1,1}, ..., x_{1,a_1}), ..., f_l(x_{l,1}, ..., x_{l,a_l})]$ と表現できる。ただし、$x_{i,j} \in \{x_1, ..., x_n\}$ とする。ここで、命題 1 より、各 i について f_i は $\mathcal{D}ep(F_0)$ の要素である。よって、

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

ここで、$C' \equiv C\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ であり、また、各 i , j について $x_{i,j}\{x_1 \mapsto t_1, ..., x_n \mapsto t_n\}$ を $t_{i,j}$ と書く。

さらに、

$$t \equiv C'[f_1(t_{1,1}, ..., t_{1,a_1}), ..., f_l(t_{l,1}, ..., t_{l,a_l})] \underset{R}{\overset{k-1}{\longrightarrow}} s$$

かつ C' は F_D の関数記号を持たないので、$s \equiv C'[s_1, ..., s_l]$ と書けて、

$$f_1(t_{1,1}, ..., t_{1,a_1}) \underset{R}{\overset{k-1\geq}{\longrightarrow}} s_1, \quad ..., \ f_l(t_{l,1}, ..., t_{l,a_l}) \underset{R}{\overset{k-1\geq}{\longrightarrow}} s_l$$

である。ゆえに、帰納法の仮定より、

$$f_1^{\#}(s_1) \underset{R'}{\overset{*}{\longrightarrow}} (t_{1,1}, ..., t_{1,a_1}), \quad ..., \ f_l^{\#}(s_l) \underset{R'}{\overset{*}{\longrightarrow}} (t_{l,1}, ..., t_{l,a_l}) \quad (3)$$

が得られる。

また、

$$f(x_1, ..., x_n) \to C[f_1(x_{1,1}, ..., x_{1,a_1}), ..., f_l(x_{l,1}, ..., x_{l,a_l})] \in R,$$

$C \in T(F_C \cup X \cup \{\Box\})$ であるから、

$$\mathcal{I}nv\mathcal{R}ule(C[f_1(x_{1,1}, ..., x_{1,a_1}), ..., f_l(x_{l,1}, ..., x_{l,a_l})]) = \langle\ C[y_1, ..., y_l]; COND\ \rangle$$

となる。ここで、$COND = \bigwedge_{i=1}^{l}(f_i^{\#}(y_i) \to (x_{i,1}, ..., x_{i,a_i})\ )$ である。したがって、R' に次の規則

$$f^{\#}(C[y_1, ..., y_l]) \to (x_{i,1}, ..., x_{i,a_i}) \Leftarrow COND$$

が含まれる。

式 (3) の書換えの最大のレベルを m とし、$\sigma = \{x_1 \mapsto t_1, ..., x_n \mapsto t_n, y_1 \mapsto s_1, ..., y_l \mapsto s_l\}$ とすると、$COND(\sigma, \underset{m}{\longrightarrow}_{R'})$ が成り立つ。よって、

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

となる。

- f がタイプ 2 の形式で定義された PT 関数のときもタイプ 1 の形式の場合と同様に証明できる。

以上により、定理は成り立つ。 □

例 5 例 *3* で R_4 から生成した R_5 について考える。$add(s^2(0), s^3(0)) \xrightarrow[R_4]{*} s^5(0)$ を考えたとき、定理 *2* より、

$$add^{\#}(s^5(0)) \xrightarrow[R_5]{*} (s^2(0), s^3(0))$$

である。実際、$add^{\#}(x_2) \xrightarrow[R_5]{} (0, x_2)$ より、

$$add^{\#}(s^3(0)) \xrightarrow[R_5]{} (0, s^3(0))$$

が示せる。これより、規則

$$add^{\#}(s(y)) \to (s(x_1), x_2) \Leftarrow add^{\#}(y) \to (x_1, x_2)$$

が適用できるので、

$$add^{\#}(s^4(0)) \xrightarrow[R_5]{} (s(0), s^3(0))$$

がいえる。さらに、同様にして、

$$add^{\#}(s^5(0)) \xrightarrow[R_5]{} (s^2(0), s^3(0))$$

が導かれる。 □

4 まとめ

本稿では、PT 関数定義から指定した関数記号の逆関数を持つ条件付き項書換え系を生成するアルゴリズムを提案し、アルゴリズムにより得られた CTRS が、逆関数になっていることを証明した。

今後の課題として、入力の条件を緩めた場合、つまり PT 項において関数記号の内側に構成子記号が出現する場合についてアルゴリズムを拡張する。

本アルゴリズムは、入力が PT 関数定義であるのに対して、出力は CTRS である。そこで、CTRS をそれと等価な PT 関数定義に変換する方法、ならびにそれが可能なクラスを明らかにする。

また、生成された逆関数の CTRS が合流性を持つクラスについて調べることも今後の課題である。

参考文献

[BN98] Franz Baader, Tobias Nipkow: Term Rewriting and All That. CAMBRIDGE Univ. Press (1998)

[Chin92] Wei-Ngan CHIN: Safe Fusion of Functional Expressions. In ACM Conference on Lisp and Functional Programming, San Francisco, Ca.,p11-20, ACM, June 1992

[Der92] Nachum Dershowitz, Subrata Mitra: Decidable matching for convergent systems (Preliminary verson). LNAI 607 Automated Deduction-CADE-11 :589-602 (1992)

[Der99] Nachum Dershowitz, Subrata Mitra: Jeopardy. LNCS 1631 Rewriting Techniques and Applications :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, p2-116. Oxford University Press, 1992

[Lank75] 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, 7:73-90, 1972

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