自動検証のためのプログラム変換法

佐藤 洸一　菊池 健太郎　青戸 等人　外山 芳人

プログラムの自動検証を容易にすることを目的としたプログラム変換法として，Giesl(2000) により文脈移動法および文脈分割法が提案されている．これらの手法は，文脈の左交換律や結合律などを利用して，末尾再帰プログラムを自動検証に適した単純再帰プログラムへと変換する．本報告では，文脈移動法と文脈分割法に基づく項書き換えシステムの変換法を提案し，その正当性を証明するとともに，計算機上での自動変換実験について説明する．

1　はじめに

末尾再帰プログラムは，再帰呼び出しの結果をそのまま返り値とし，呼び出し時の実行状態を保持する必要のない効率的なプログラムとして知られている．末尾再帰プログラムの多くは計算の途中結果を保存するためアキュムレータとよばれる引数をもつ．しかし，アキュムレータの値は再帰呼び出しで変化するため，末尾再帰プログラムに対して帰納法に基づく自動証明を適用することが困難である．

従来のプログラム変換の研究は，単純再帰を末尾再帰に変換するなど，効率的なプログラムへの変換が主な目的であり，プログラムの検証を容易にすることを目的とした研究はあまり知られていない．しかし，高度なプログラム自動検証のためには，後者のような，検証に適したプログラムへの変換が必要である．

そのようなプログラム変換法として，文脈移動法と文脈分割法 [4] が提案されている．これらは，適当な条件のもとで末尾再帰プログラムを等価な単純再帰プログラムへと変換する手法である．単純再帰プログラムは，計算にアキュムレータを使用しないため，帰納法による自動証明に適したプログラムとなっている．

項書き換えシステムは，関数型言語の形式的な計算モデルであり，プログラムに関する様々な性質や操作を言語の制約に捉われず柔軟に取り扱うことが可能となる．したがって，変換対象を項書き換えシステムとすることで，変換の本質に迫った考察が可能となる．

本報告では，文脈移動法と文脈分割法のそれぞれに基づいた項書き換えシステムの変換法を提案するとともに，その正当性を示す．また，計算機上での自動変換実験を通してその有効性を確認する．

2　準備

本節では，項書き換えシステム (TRS) に関する用語や概念を文献 [2] [5] に従って定義する．

関数記号と変数記号の有限集合をそれぞれ $\mathcal{F}, \mathcal{V}$ とする．$\mathcal{F}, \mathcal{V}$ で構成できる項全体の集合を $\mathcal{T}(\mathcal{F}, \mathcal{V})$ とする．項 t に含まれる変数全体の集合を $\mathcal{V}(t)$，関数記号全体の集合を $\mathcal{F}(t)$ で表す．$t_1, t_2, \ldots, t_n$ のような項の列を $\bar{t}$ で表す．$\mathcal{V}(t) = \emptyset$ となる項 t を基底項とよび，その集合を $\mathcal{T}(\mathcal{F})$ で表す．項 t の根記号 $root(t)$ は，$t \in \mathcal{V}$ のとき t，$t = f(t_1, \ldots, t_n)$ のとき f である．ホールとよばれる特別な定数記号 $\Box$ を含む項を文脈とよび，$\Box$ を一つだけ含む文脈を $C[\,]$ と表す．$C[\,]$ の $\Box$ を項 t で置き換えて得られる項を $C[t]$ で表す．また，C の複数のホール $\Box_i (1 \leq i \leq n)$ を t_i で置き換えた項を $C[t_1, t_2, \ldots, t_n]$ のように表す．$t = C[u]$

Program transformation for automated verification.

Koichi Sato, Kentaro Kikuchi, Takahito Aoto, Yoshihito Toyama, 東北大学電気通信研究所, Research Institute of Electrical Communication, Tohoku University.

となる文脈 C が存在するとき，u は t の部分項であるという．変数集合 $\mathcal{V}$ から項全体の集合 $\mathcal{T}(\mathcal{F},\mathcal{V})$ への写像 θ を代入とよび，θ を $\mathcal{T}(\mathcal{F},\mathcal{V})$ から $\mathcal{T}(\mathcal{F},\mathcal{V})$ への写像へ自然に拡張する．以下では $\theta(t)$ を $t\theta$ と表記する．

項の対 (l,r) が $l \notin \mathcal{V}$ かつ $\mathcal{V}(l) \supseteq \mathcal{V}(r)$ をみたすとき (l,r) を書き換え規則といい，$l \to r$ と表す．項書き換えシステム R は書き換え規則の集合である．項書き換えシステム R における書き換え関係 $\to_R$ は以下のように定義される．

$s \to_R t$
$\overset{\text{def}}{\iff} \exists l \to r \in R.\exists C[\ ].\exists \theta. s = C[l\theta] \wedge t = C[r\theta]$

このとき，s は t に書き換えられるという．文脈から R が明らかな場合，$\to_R$ を $\to$ と略記する．なお，以下では他の記法についても添字 R の省略を同様に行う．また，項の列について $\bar{t} \to \bar{s}$ は $t_1 \to s_1, \cdots, t_n \to s_n$ を表す．$s \to t$ となる t が存在しないとき s を正規形という．項 t の項書き換えシステム R で得られる正規形を $t{\downarrow}$ で表す．項の列に対しては $\bar{t}{\downarrow} = t_1{\downarrow}, \ldots, t_n{\downarrow}$ と定義される．また，代入 θ に対して $\theta{\downarrow}$ を $\theta{\downarrow}(x) = (\theta(x)){\downarrow}$ と定める．$\to$ の反射推移閉包を $\overset{*}{\to}$ で表す．R を項書き換えシステムとし，任意の基底項 t, t_1, t_2 について $t \overset{*}{\to} t_1$ かつ $t \overset{*}{\to} t_2$ ならばある項 s が存在し $t_1 \overset{*}{\to} s$ かつ $t_2 \overset{*}{\to} s$ ならば R は基底合流性をもつという．項書き換えシステム R の規則の左辺の根記号を定義関数記号という．R の定義関数記号の集合 $\{root(l) \mid l \to r \in R\}$ を $\mathcal{D}$ と表す．項 t に現れる定義関数記号の集合を $\mathcal{D}(t)$ と記す．このとき，$f \in \mathcal{F} \setminus \mathcal{D}$ を構成子記号といい，R の構成子記号集合を $\mathcal{C}$ と記す．項 $t \in \mathcal{T}(\mathcal{C})$ を基底構成子項という．基底項と基底構成子項への代入をそれぞれ θ_g, θ_{gc} 等で表す．規則 $l \to r$ において，$root(l) = f$，$f \in \mathcal{F}(r)$ であるとき，$l \to r$ は f の再帰規則という．とくに，$root(l) = root(r)$ のとき $l \to r$ は末尾再帰規則という．項書き換えシステム R によって，任意の基底項 s に対してある基底構成子項 t が存在して $s \overset{*}{\to} t$ が成立するとき，R は十分完全性をもつという．本報告では，変換対象として十分完全性と基底合流性をもつ項書き換えシステムを考える．

3 項書き換えシステムに対する文脈移動法

3.1 文脈移動法による項書き換えシステムの変換

関数型プログラムの文脈移動法 [4] は，末尾再帰規則のアキュムレータの文脈をその再帰呼び出しの外に移動する．項書き換えシステムでは，再帰呼び出しは左辺と同じ根記号をもつ右辺の部分項に相当するため，この部分項におけるアキュムレータを含む文脈を再帰呼び出しの外へ移動することで項書き換えシステムの文脈移動法による変換を定める．

定義 3.1 (項書き換えシステムに対する文脈移動法)**.** 項書き換えシステムに対する文脈移動法は，以下の項書き換えシステム R から R' への変換規則である．

$$R = \begin{cases} f(\bar{l}_i, z) \to f(\bar{r}_i, C_i[z]) & (1 \le i \le m) \\ f(\bar{l}_j, z) \to C_j[z] & (m+1 \le j \le n) \\ l_k \to r_k & (n+1 \le k \le p) \end{cases}$$

$$R' = \begin{cases} f'(\bar{l}_i, z) \to C_i[f'(\bar{r}_i, z)] & (1 \le i \le m) \\ f'(\bar{l}_j, z) \to C_j[z] & (m+1 \le j \le n) \\ l_k \to r_k & (n+1 \le k \le p) \end{cases}$$

変換対象の関数記号を f とし，変数 z をアキュムレータとよぶ．このとき，変換対象の f と z の出現について以下の条件が成立するものとする．

$\forall i. f \notin \mathcal{F}(\bar{l}_i, \bar{r}_i, l_i, r_i, C_i[\ ]) \wedge z \notin \mathcal{V}(\bar{l}_i, \bar{r}_i, C_i[\ ])$

以下では，適当な名前替えを行うことにより各書き換え規則は互いに異なる変数をもつものと仮定する．

また，R および R' の分割を以下のように定義する．

$R_A = \{l \to r \in R \mid root(l) = root(r) = f\}$
$R_B = \{l \to r \in R \mid root(l) = f \wedge f \notin \mathcal{F}(r)\}$
$R_C = \{l \to r \in R \mid f \notin \mathcal{F}(l, r)\}$
$R'_A = \{l \to r \in R' \mid root(l) = f' \wedge f' \in \mathcal{F}(r)\}$
$R'_B = \{l \to r \in R' \mid root(l) = f' \wedge f' \notin \mathcal{F}(r)\}$
$R'_C = R_C$

R_A, R'_A は R, R' それぞれの f, f' の再帰規則，R_B, R'_B は f, f' の非再帰規則，R_C は補助関数に対する規則である．

例 3.2 (文脈移動法による変換例)**.** 以下の項書き換えシステム R を考える．

$$R = \begin{cases} Mult(S(x), y, z) \to Mult(x, y, Add(y, z)) \\ Mult(0, y, z) \to z \\ Add(S(x), y) \to S(Add(x, y)) \\ Add(0, y) \to y \end{cases}$$

変換対象の関数記号を $Mult$，アキュムレータを z として文脈移動法を適用すると，以下の項書き換えシステム R' が得られる．

$$R' = \begin{cases} Mult'(S(x), y, z) \to Add(y, Mult'(x, y, z)) \\ Mult'(0, y, z) \to z \\ Add(S(x), y) \to S(Add(x, y)) \\ Add(0, y) \to y \end{cases}$$ □

3.2 文脈移動法による変換の正当性

本節では，文脈移動法による変換の正当性，つまり変換前の R と変換後の R' の等価性を示す．

変換の正当性を保証するため，以下の条件を考える．

定義 3.3 (文脈交換律)．$C_1, \cdots, C_n$ を文脈移動法の変換規則に表れる文脈とする．このとき，以下の条件を文脈交換律とよぶ．

$$\forall i(1 \le i \le m).\forall j(1 \le j \le n).\forall \theta_{gc}.$$
$$C_i[C_j[z]]\theta_{gc}\downarrow = C_j[C_i[z]]\theta_{gc}\downarrow$$

ただし，$i = j$ のとき，$C_i[\,], C_j[\,]$ は共通変数をもたないよう名前替えされているものとする．

補題 3.4. R から R' への文脈移動法による変換において文脈交換律を仮定する．このとき，任意の基底項 s と基底構成子項 v に対して，

$$s \xrightarrow{*}_R v \;\Rightarrow\; s' \xrightarrow{*}_{R'} v$$

ここで、s' は s に出現する f を f' で置き換えた項である．

証明．任意の基底構成子代入 θ_{gc} と基底構成子項 v に対して以下を示せば十分である．

$$f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v \;\Rightarrow\; f'(\bar{x}, z)\theta_{gc} \xrightarrow{*}_{R'} v$$

$f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v$ と仮定する．$f(\bar{x}, z)\theta_{gc}$ から v への書き換え列は以下のようにおくことができる．

$$\begin{aligned} f(\bar{x}, z)\theta_{gc} &= f(\bar{l}_{i_1}\theta_1, u_1) \\ &\to_{R_A} f(\bar{r}_{i_1}\theta_1, C_{i_1}\theta_1[u_1]) \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_2}\theta_2, u_2) \\ &\to_{R_A} f(\bar{r}_{i_2}\theta_2, C_{i_2}\theta_2[u_2]) \\ &\;\;\vdots \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_n}\theta_n, u_n) \\ &\to_{R_B} C_{i_n}\theta_n[u_n] \\ &\xrightarrow{*}_R v \end{aligned}$$

項 $f(\bar{t}, t)$ に適用できる R_A の規則は $\bar{t}$ によってのみ定まり，t に影響されない．このことから，上の書き換え列で適用する R_A の規則の順番を変えず，かつ R_B を適用するまで f の最後の引数 u を書き換えない以下の書き換え列が得られる．

$$\begin{aligned} f(\bar{x}, z)\theta_{gc} &= f(\bar{l}_{i_1}\theta_1, u) \\ &\to_{R_A} f(\bar{r}_{i_1}\theta_1, C_{i_1}\theta_1[u]) \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_2}\theta_2, C_{i_1}\theta_1[u]) \\ &\to_{R_A} f(\bar{r}_{i_2}\theta_2, C_{i_2}\theta_2[C_{i_1}\theta_1[u]]) \\ &\;\;\vdots \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_n}\theta_n, C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u]\cdots]) \\ &\to_{R_B} C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u]\cdots]] \end{aligned}$$

この書き換え列に対応する R' による書き換えを考える．R' の規則の左辺は f が f' に置き換わっている以外は同じ項であり，$R_C = R'_C$ であるため，以下の書き換え列が得られる．

$$\begin{aligned} f'(\bar{x}, z)\theta_{gc} &= f'(\bar{l}_{i_1}\theta_1, u) \\ &\to_{R'_A} C_{i_1}\theta_1[f'(\bar{r}_{i_1}\theta_1, u)] \\ &\xrightarrow{*}_{R'_C} C_{i_1}\theta_1[f'(\bar{l}_{i_2}\theta_2, u)] \\ &\to_{R'_A} C_{i_1}\theta_1[C_{i_2}\theta_2[f'(\bar{r}_{i_2}\theta_2, u)]] \\ &\;\;\vdots \\ &\xrightarrow{*}_{R'_C} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[f'(\bar{l}_{i_n}\theta_n, u)]\cdots] \\ &\to_{R'_B} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u]]\cdots] \end{aligned}$$

ここで，$u \in \mathcal{T}(\mathcal{C})$，$\forall i.f, f' \notin \mathcal{F}(C_i\theta_i[\,])$，$\theta_i\downarrow_R = \theta_i\downarrow_{R_C} = \theta_i\downarrow_{R'}$ となり，R の十分完全性より $\theta_i\downarrow_{R_C}$ は基底構成子代入となるので，文脈交換律と基底合流性より以下が成り立つ．

$$
\begin{aligned}
v &= C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u]\cdots]]\downarrow_R \\
&= C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u]\cdots]]\downarrow_{R_C} \\
&= C_{i_n}(\theta_n\downarrow_{R_C})[\cdots C_{i_1}(\theta_1\downarrow_{R_C})[u]\cdots]\downarrow_{R_C} \\
&= C_{i_1}(\theta_1\downarrow_{R_C})[\cdots C_{i_n}(\theta_n\downarrow_{R_C})[u]\cdots]\downarrow_{R_C} \\
&= C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u]]\cdots]\downarrow_{R_C} \\
&= C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u]]\cdots]\downarrow_{R'}
\end{aligned}
$$

よって，以下が成立する．

$$
\begin{aligned}
f'(\bar{x},z)\theta_{gc}\downarrow_R &\xrightarrow{*}_{R'} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u]]\cdots] \\
&\xrightarrow{*}_{R'} v
\end{aligned}
$$

□

系 3.5. R' は十分完全性をもつ．

証明. 補題 3.4 と R の十分完全性より成立する． □

補題 3.6. R から R' への文脈移動法による変換において文脈交換律を仮定する．このとき，任意の基底項 s' と基底構成子項 v に対して，

$$s' \xrightarrow{*}_{R'} v \;\Rightarrow\; s \xrightarrow{*}_R v$$

ここで、s は s' に出現する f' を f で置き換えた項である．

証明. 系 3.5 より R' は十分完全性をもつので，補題 3.4 と同様にして証明できる． □

系 3.7. R' は基底合流性をもつ．

証明. $t' \xrightarrow{*}_{R'} t'_1, t' \xrightarrow{*}_{R'} t'_2$ とする．このとき，系 3.5 より $t' \xrightarrow{*}_{R'} t'_1 \xrightarrow{*}_{R'} t'_1\downarrow_{R'}$，$t' \xrightarrow{*}_{R'} t'_2 \xrightarrow{*}_{R'} t'_2\downarrow_{R'}$ および $t'_1\downarrow_{R'}, t'_2\downarrow_{R'} \in \mathcal{T}(\mathcal{C})$ が成立する．t' 中の f' を f に置き換えた項 t を考えると，補題 3.6 より以下が成立する．

$$t \xrightarrow{*}_R t_1\downarrow_R = t'_1\downarrow_{R'} \;,\; t \xrightarrow{*}_R t_2\downarrow_R = t'_2\downarrow_{R'}$$

ここで，R の基底合流性より $t'_1\downarrow_{R'} = t'_2\downarrow_{R'}$ となる．

□

以上より，文脈移動法による変換の正当性が示される．

定理 3.8 (文脈移動法による変換の正当性)**.** 文脈移動法により項書き換えシステム R から R' が得られたとする．このとき，任意の基底項 s について $s\downarrow_R = s'\downarrow_{R'}$ が成立する．ここで，s' は s に出現するすべての f を f' に置き換えた項である．

証明. 補題 3.4，補題 3.6，系 3.5，系 3.7 より成立する． □

4 項書き換えシステムに対する文脈分割法

4.1 文脈分割法による項書き換えシステムの変換

関数型プログラムに対する文脈分割法 [4] は，末尾再帰呼び出し中のアキュムレータの文脈を，各分岐共通の文脈と各分岐固有の部分に分け，共通の文脈を再帰呼び出しの外へ，固有部分をアキュムレータ位置へ移動する．また，再帰呼び出しを行わない分岐については，固有部分のみを残す．文脈分割法の項書き換えシステムへの適用では，同様にアキュムレータの位置に出現する文脈を共通文脈と固有部分に分け，共通文脈を外に出し，固有部分をアキュムレータの位置に移動する．

定義 4.1 (項書き換えシステムに対する文脈分割法)**.** 項書き換えシステムに対する文脈分割法は，以下の項書き換えシステム R から R' への変換規則である．

$$
R = \begin{cases}
f(\bar{l}_i, z) \to f(\bar{r}_i, C[r_i, z]) & (1 \le i \le m) \\
f(\bar{l}_{i'}, z) \to f(\bar{r}_{i'}, z) & (m+1 \le i' \le m') \\
f(\bar{l}_j, z) \to C[r_j, z] & (m'+1 \le j \le n) \\
f(\bar{l}_{j'}, z) \to z & (n+1 \le j' \le n') \\
l_k \to r_k & (n'+1 \le k \le p)
\end{cases}
$$

$$
R' = \begin{cases}
f'(\bar{l}_i) \to C[f'(\bar{r}_i), r_i] & (1 \le i \le m) \\
f'(\bar{l}_{i'}) \to f'(\bar{r}_{i'}) & (m+1 \le i' \le m') \\
f'(\bar{l}_j) \to r_j & (m'+1 \le j \le n) \\
f'(\bar{l}_{j'}) \to e & (n+1 \le j' \le n') \\
l_k \to r_k & (n'+1 \le k \le p)
\end{cases}
$$

変換対象の関数記号を f とし，変数 z をアキュムレータとよぶ．変換規則の C を共通文脈とよぶ．このとき，変換対象の f と z の出現について以下の条件が成立するものとする．

$$\forall i. f \notin \mathcal{F}(\bar{l}_i, \bar{r}_i, l_i, r_i, C[\,]) \wedge z \notin \mathcal{V}(\bar{l}_i, \bar{r}_i, r_i, C[\,])$$

が成立する．また，項 e は共通文脈 C の単位元であり，その定義は後述する．

以下では，適当な名前替えを行うことにより各書き換え規則は互いに異なる変数をもつものと仮定する．

また，R, R' の分割を以下のように定義する．

$$R_A = \{l \to r \in R \mid root(l) = root(r) = f\}$$
$$R_B = \{l \to r \in R \mid root(l) = f \wedge f \notin \mathcal{F}(r)\}$$
$$R_C = \{l \to r \in R \mid f \notin \mathcal{F}(l, r)\}$$
$$R'_A = \{l \to r \in R' \mid root(l) = f' \wedge f' \in \mathcal{F}(r)\}$$
$$R'_B = \{l \to r \in R' \mid root(l) = f' \wedge f' \notin \mathcal{F}(r)\}$$
$$R'_C = R_C$$

さらに R_A, R_B, R'_A, R'_B を以下のように分割する．

$$R_{A0} = \{l \to r \in R_A \mid r = f(\bar{r}_i, C[r_i, z])\}$$
$$R_{A1} = \{l \to r \in R_A \mid r = f(\bar{r}_{i'}, z)\}$$
$$R_{B0} = \{l \to r \in R_B \mid r = \bar{r}_j\}$$
$$R_{B1} = \{l \to r \in R_B \mid r = z\}$$
$$R'_{A0} = \{l \to r \in R'_A \mid root(r) \neq f'\}$$
$$R'_{A1} = \{l \to r \in R'_A \mid root(r) = f'\}$$
$$R'_{B0} = \{l \to r \in R'_B \mid r = r_j\}$$
$$R'_{B1} = \{l \to r \in R'_B \mid r = e\}$$

$R_{A0}, R_{A1}, R'_{A0}, R'_{A1}$ は f, f' の再帰規則であり，$R_{B0}, R_{B1}, R'_{B0}, R'_{B1}$ は f, f' の非再帰規則である．また，$R_{A0}, R_{B0}, R'_{A0}, R'_{B0}$ は共通文脈 C を右辺にもつ規則およびそのような規則を変換して得られた規則，$R_{A1}, R_{B1}, R'_{A1}, R'_{B1}$ は共通文脈 C が変換の前後で出現しない規則である．$R_C = R'_C$ は補助関数に対応する規則である．$R_A \cup R_B$ などの集合の和を R_{AB} などのように表記する．

例 4.2 (文脈分割法による変換例)．以下の R を文脈分割法によって R' へと変換する．

$$R = \begin{cases} Cat(Nil, z) \to z \\ Cat(Cons(x, xs), z) \to Cat(xs, App(z, x)) \\ App(Nil, y) \to y \\ App(Cons(x, xs), y) \to Cons(x, App(xs, z)) \end{cases}$$

R において変換対象の関数記号は Cat，アキュムレータは z である．共通の文脈 C は $C = App(\Box_2, \Box_1)$ で，この文脈の単位元は Nil となる．

$$R' = \begin{cases} Cat'(Nil) \to Nil \\ Cat'(Cons(x, xs)) \to App(x, Cat'(xs)) \\ App(Nil, y) \to y \\ App(Cons(x, xs), y) \to Cons(x, App(xs, z)) \end{cases}$$

□

4.2 文脈分割法による変換の正当性

本節では文脈分割法による書き換えシステム変換の正当性，つまり，R と R' の等価性を示す．

文脈分割法の変換規則における共通文脈 C について以下の 2 つの条件を考える．

定義 4.3 (文脈結合律)．

$$\forall \theta_{gc}.C[C[x, y], z]\theta_{gc}{\downarrow_R} = C[x, C[y, z]]\theta_{gc}{\downarrow_R}$$

定義 4.4 (文脈単位元)．

$$\forall \theta_g.C[z, e]\theta_{gc}{\downarrow_R} = C[e, z]\theta_{gc}{\downarrow_R} = \theta_{gc}(z){\downarrow_R}$$

補題 4.5. R から R' への文脈分割法による変換において文脈結合律と文脈単位元の条件を仮定する．このとき，任意の基底項 s と基底構成子項 v に対して，

$$s \xrightarrow{*}_R v \;\Rightarrow\; s' \xrightarrow{*}_{R'} v$$

ここで、s' は s に出現する $f(\bar{t}, t)$ を $C[f'(\bar{t}), t]$ で置き換えた項である．

証明．任意の基底構成子代入 θ_{gc} と基底構成子項 v に対して以下を示せば十分である．

$$f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v \Rightarrow C[f'(\bar{x}), z]\theta_{gc} \xrightarrow{*}_{R'} v$$

$f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v$ と仮定する．$f(\bar{x}, z)\theta_{gc}$ から v への書き換え列は以下のようにおくことができる．

$$\begin{aligned} f(\bar{x}, z)\theta_{gc} &= f(\bar{l}_{i_1}\theta_1, u_1) \\ &\to_{R_A} f(\bar{r}_{i_1}\theta_1, u'_1) \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_2}\theta_2, u_2) \\ &\to_{R_A} f(\bar{r}_{i_2}\theta_2, u'_2) \\ &\;\;\vdots \\ &\xrightarrow{*}_{R_C} f(\bar{l}_{i_n}\theta_n, u_n) \\ &\to_{R_B} u'_n \\ &\xrightarrow{*}_R v \end{aligned}$$

ここで $v = f(\bar{x}, z)\theta_{gc}{\downarrow_R}$ とし，任意の j について $u'_j = C[r_{i_j}, u_j]$ または $u'_j = u_j$ である．項 $f(\bar{t}, t)$ に適用できる R_A の規則は t に影響されない．よって上の書き換え列で f の最後の引数 u_j もしくは u'_j を一切書き換えない系列を考えることができる．このとき，

$$f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R C[r_{j_m}\theta'_m, \cdots, C[r_{j_1}\theta'_1, u_1] \cdots]$$

となっている．次にこの R の書き換え列に対応する R' による書き換え列を考える．R_{AB} で $f(\bar{t}, t)$ を書き換える場合に適用される規則は $\bar{t}$ によって決まり，R'_{AB} による $f(\bar{t})$ の書き換えで適用可能な規則と対応

する．さらに $R_C = R'_C$ であるため，以下の書き換え列が得られる．

$$\begin{aligned}
C[f'(\bar{x}), u]\theta_{gc} &\xrightarrow{*}_{R'_C} C[f'(\bar{l}_{i_1}\theta_1), u] \\
&\to_{R'_A} C[D_{i_1}[f'(\bar{r}_{i_1}\theta_1)], u] \\
&\xrightarrow{*}_{R'_C} C[D_{i_1}[f'(\bar{l}_{i_2}\theta_2)], u] \\
&\to_{R'_A} C[D_{i_1}[D_{i_2}[f'(\bar{r}_{i_2}\theta_2)]], u] \\
&\vdots \\
&\xrightarrow{*}_{R'_C} C[D_{i_1}[\cdots D_{i_{n-1}}[f'(\bar{l}_{i_n}\theta_n)]\cdots], u] \\
&\to_{R'_B} C[D_{i_1}[\cdots D_{i_{n-1}}[r'_{i_n}]\cdots], u]
\end{aligned}$$

ここで $r'_{i_n} = e$ または $r'_{i_n} = r_{i_n}$ であり，文脈 $D_{i_j} = \Box$ または $D_{i_j} = C[\Box, r_{i_j}\theta_j]$ である．この書き換え列の最終項は共通文脈 C を用いて書くと以下のようになる．

$u'_n = C[C[\cdots C[r_{j_m}\theta'_m, r_{j_{m-1}}\theta'_{m-1}], \cdots, r_{j_1}\theta'_1], u]$

ここで，$u \in \mathcal{T}(\mathcal{C}), \forall k.f, f' \notin \mathcal{F}(C, s_{j_k}\theta_k)$，文脈結合律より

$$\begin{aligned}
v &= C[r_{j_m}\theta'_m, C[r_{j_{m-1}}\theta'_{m-1}, \cdots, C[r_{j_1}\theta'_1, u]\cdots]]\downarrow_R \\
&= C[r_{j_m}\theta'_m, C[r_{j_{m-1}}\theta'_{m-1}, \cdots, C[r_{j_1}\theta'_1, u]\cdots]]\downarrow_{R_C} \\
&= C[r_{j_m}(\theta'_m\downarrow_{R_C}), \cdots, C[r_{j_1}(\theta'_1\downarrow_{R_C}), u]\cdots]\downarrow_{R_C} \\
&= C[\cdots C[r_{j_m}(\theta'_m\downarrow_{R_C}), r_{j_{m-1}}(\theta'_{m-1}\downarrow_{R_C})], \cdots, u]\downarrow_{R_C} \\
&= C[C[\cdots C[r_{j_m}\theta'_m, r_{j_{m-1}}\theta'_{m-1}], \cdots, r_{j_1}\theta'_1], u]\downarrow_{R_C} \\
&= C[C[\cdots C[r_{j_m}\theta'_m, r_{j_{m-1}}\theta'_{m-1}], \cdots, r_{j_1}\theta'_1], u]\downarrow_{R'}
\end{aligned}$$

よって，以下が成立する．

$$\begin{aligned}
&C[f'(\bar{x}, u)]\theta_{gc} \\
&\xrightarrow{*}_{R'} C[r_{j_m}\theta'_m, C[r_{j_{m-1}}\theta'_{m-1}, \cdots, C[r_{j_1}\theta'_1, u]\cdots]] \\
&\xrightarrow{*}_{R'} v
\end{aligned}$$

□

補題 4.6. R から R' への文脈分割法による変換において文脈結合律と文脈単位元の条件を仮定する．このとき，任意の基底項 s と基底構成子項 v に対して，

$$s \xrightarrow{*}_R v \;\Rightarrow\; s' \xrightarrow{*}_{R'} v$$

ここで、s' は s に出現する $f(\bar{t}, e)$ を $f'(\bar{t})$ で置き換えた項である．

証明. 補題 4.5 の証明において $\theta_{gc}(z) = e$ とおくことで同様に示せる．なお，$R_C = R'_C$ より文脈の単位元条件は $\downarrow_{R'}$ についても成立することに注意する．□

系 4.7. R' は十分完全性をもつ．

証明. 補題 4.5，補題 4.6 と R の十分完全性より成立する．□

補題 4.8. R から R' への文脈分割法による変換において文脈結合律と文脈単位元の条件を仮定する．このとき，任意の基底項 s' と基底構成子項 v に対して，

$$s' \xrightarrow{*}_{R'} v \;\Rightarrow\; s \xrightarrow{*}_R v$$

ここで、s は s' に出現する $f'(\bar{t})$ を $f(\bar{t}, e)$ で置き換えた項である．

証明. 系 4.7 より R' は十分完全性を持つ．補題 4.6 と同様にして証明できる．□

系 4.9. R' は基底合流性をもつ．

証明. $t' \xrightarrow{*}_{R'} t'_1, t' \xrightarrow{*}_{R'} t'_2$ とする．このとき，系 4.7 より $t' \xrightarrow{*}_{R'} t'_1 \xrightarrow{*}_{R'} t'_1\downarrow_{R'}$，$t' \xrightarrow{*}_{R'} t'_2 \xrightarrow{*}_{R'} t'_2\downarrow_{R'}$ および $t'_1\downarrow_{R'}, t'_2\downarrow_{R'} \in \mathcal{T}(\mathcal{C})$ が成立する．ここで t' 中の $f'(\bar{t})$ を $f(\bar{t}, e)$ に置き換えた項 t を考えると，補題 4.5，補題 4.8 より

$$t \xrightarrow{*}_R t_1\downarrow_R = t'_1\downarrow_{R'}\ ,\ t \xrightarrow{*}_R t_2\downarrow_R = t'_2\downarrow_{R'}$$

である．よって，R の基底合流性より $t'_1\downarrow_{R'} = t'_2\downarrow_{R'}$ となる．□

以上より，文脈分割法による変換の正当性が示される．

定理 4.10 (文脈分割法による変換の正当性)**.** 文脈分割法により項書き換えシステム R から R' が得られたとする．このとき，任意の基底項 s について $s\downarrow_R = s'\downarrow_{R'}$ が成立する．ここで，s' は s に出現するすべての $f(\bar{t}, e)$ を $f'(\bar{t})$，$t \neq e$ である t について $f(\bar{t}, t)$ を $C[f'(\bar{t}), t]$ に置き換えた項である．

証明. R の十分完全性および補題 4.5，補題 4.8，系 4.7，系 4.9 より成立する．□

5 実装と実験

文脈移動法と文脈分割法による項書き換えシステムの変換手続きを関数型言語 Standard ML of New Jersey [1] を用いて実装した．文脈移動法については，変換を適用できる形かどうかの判定手続きと実際の変換手続きを実装した．R の十分完全性，文脈交換律の判定は現段階では未実装であり，変換実験では手作業での判定を行った．文脈分割法による項書き換えシステムの変換手続きも変換を適用できる形かの判

表 1　項書き換えシステム変換の実験結果

R の機能	文脈移動法	文脈分割法
加算	○	×
乗算	○	○
乗算 (2)	○	$\times^*$
2 倍	○	$\times^*$
1/2	○	$\times^*$
二進対数	○	$\times^*$
2 の冪	○	$\times^*$
除算	○	$\times^*$
リスト長	○	$\times^*$
総減算	○	×
総和	○	○
重み付き総和	○	○
階乗	○	○
冪乗	○	○
最大要素	○	○
最小要素	○	○
Member	○	○
Subset	○	○
スカラー積	○	○
リスト結合	×	○
Filter	×	○
分割	×	○
First	×	○
Increment	×	○
減算	×	×

定手続き，実際の変換手続きを実装し，文脈結合律，文脈単位元の条件の判定機能は未実装である．

これらのシステムによる各種項書き換えシステムの変換実験の結果を表 1 に示す．実験に用いた項書き換えシステムは文献 [3] で与えられたプログラムを自然な項書き換えシステムに置き換えたものである．○が変換成功，× が失敗を表す．$\times^*$ の例は変換には失敗したものの項書き換えシステムを変形することで成功した．

全 25 例に対して文脈移動法と文脈分割法両方の変換実験を行い，文脈移動法の成功例が 19 例，失敗例 6 例となり，文脈分割法の成功例が 15 例，失敗例 10 例となった．文脈分割法による失敗例 10 例のうち，7 例については実験例に変換を施すことで成功する例であった．両方で成功した例は 10 例であった．以下では，両方で変換に成功する例，どちらか一方で変換に成功する例それぞれの代表的な例，失敗例や特殊な事例を示す．

例 5.1 (両方とも成功)**.** 以下の乗算の項書き換えシステム R では両方の変換が成功した．なお，以下では補助関数の規則については省略する．

$$R = \begin{cases} Mult(0, y, z) \to z \\ Mult(S(x), y, z) \to Mult(x, y, Add(y, z)) \end{cases}$$

文脈移動法の変換システムによって得られた項書き換えシステム R' を以下に示す．

$$R' = \begin{cases} Mult'(0, y, z) \to z \\ Mult'(S(x), y, z) \to Add(y, Mult'(x, y, z)) \end{cases}$$

また，文脈分割法でも変換に成功し，R とは異なる以下の R'' が得られた．

$$R'' = \begin{cases} Mult'(0, y) \to 0 \\ Mult'(S(x), y) \to Add(Mult'(x, y), y) \end{cases}$$

このようなどちらでも変換可能な項書き換えシステムは表 1 で総和，階乗，最大要素などである．これらはいずれも加算，乗算，最大値 (Max 関数) といった交換律と結合律どちらもみたす演算をアキュムレータに対して行なう．これらの演算では両方の変換が成功する．□

例 5.2 (文脈移動法のみ成功)**.**

$$R = \begin{cases} Suball(Nil, z) \to z \\ Suball(Cons(x, xs), z) \to \\ \quad Suball(xs, Minus(z, x)) \end{cases}$$

総減算 $Suball(l, x)$ は x からリスト l の要素すべてを取り除く．減算 $Minus$ は文脈交換律が成り立つため文脈移動法で変換が可能である．自動変換システムによって以下の項書き換えシステム R' が得られた．

$$R' = \begin{cases} Suball'(Nil, z) \to z \\ Suball'(Cons(x, xs), z) \to \\ \quad Minus(Suball'(xs, z), x) \end{cases}$$

文脈分割法においては，文脈 $Minus(\Box_2, \Box_1)$ にお

いて結合律が成立しないため変換は失敗した． □

例 5.3 (文脈分割法のみ成功)**.**

$$R = \begin{cases} Cat(Nil, z) \to z \\ Cat(Cons(l, ls), z) \to Cat(ls, Append(z, l)) \end{cases}$$

リストの結合を行う項書き換えシステムは文脈分割法のみで成功した．自動変換システムによって以下の R' が得られた．

$$R' = \begin{cases} Cat'(Nil) \to Nil \\ Cat'(Cons(l, ls)) \to Append(l, Cat'(ls)) \end{cases}$$

文脈分割法のみ成功する例において文脈は交換律が成り立たたず結合律が成り立つリストの結合 $Append$ であった． □

例 5.4 (両方とも失敗)**.**

$$R = \begin{cases} Minus(0, z) \to z \\ Minus(S(x), S(z)) \to Minus(x, z) \end{cases}$$

減算を行う項書き換えシステムは文脈移動法，文脈分割法どちらでも失敗した．この原因は 2 番目の規則左辺のアキュムレータの位置が $S(z)$ となっているため変換が適用できる形とはなっていないことである．しかし，自然数の S を一つ消す前者関数 P を考え，以下のように変形すると変換が可能となる．この例ではプログラムを項書き換えシステムとして自然な形に直す作業に問題があったと言える．

$$R = \begin{cases} Minus(0, z) \to z \\ Minus(S(x), z) \to Minus(x, P(z)) \end{cases}$$

□

例 5.5 (変形後に文脈分割法が変換成功)**.** 表 1 で $\times^*$ で示した，文脈分割法の直接の適用は失敗するが適切な変形により文脈分割法が適用できる例について説明する．

$$R = \begin{cases} Double(0, z) \to z \\ Double(S(x), z) \to Double(x, S(S(z))) \end{cases}$$

R は自然数を 2 倍する項書き換えシステムである．この項書き換えシステムは文脈移動法で変換が成功，文脈分割法で変換に失敗した．文脈は $S(S(\Box))$ となり，加算であるため文脈交換律，結合律はともに成立するが，ホールが 2 つないため文脈分割法の共通文脈として用いることはできない．しかし，$S(S(\Box))$ が $Add(S(S(0)), \Box)$ と等価であること，Add は交換律，結合律が成立することを考えると以下の R' のように加算関数 Add を導入することで変換が可能となる．

$$R' = \begin{cases} Double(0, z) \to z \\ Double(S(x), z) \to \\ \quad Double(x, Add(S(S(0)), z)) \end{cases}$$

以下に変形した後に文脈分割法を適用した項書き換えシステム R'' を示す．

$$R'' = \begin{cases} Double'(0) \to 0 \\ Double'(S(x)) \to \\ \quad Add(Double'(x), S(S(0))) \end{cases}$$

乗算 (2) や二進対数など 8 例が二項演算の形ではない文脈のため変換に失敗したが，このように適切な変換を施すことにより変換が可能となる．ただし，加算については，$S(z)$ で表されるアキュムレータに 1 加算する項を $Add(S(0), z)$ とおくことで文脈分割法を適用できたが，実質的には文脈に変換対象の関数が入っている場合と考えられるので除外した． □

6 まとめと今後の課題

本報告では，関数型末尾再帰プログラムについて提案されていたプログラム変換法である文脈移動法および文脈分割法を，項書き換えシステムへ適用するための変換規則を提案した．また，それぞれの変換法の正当性を示すとともに，変換は項書き換えシステムの基底合流性と十分完全性を保存していることを明らかにした．また，提案手法に基づき自動変換システムを実装し，正当性を保証する条件のもとで正しく変換が行われることを確認した．

現時点では，変換の正当性を保証するための一部の条件の自動判定が未実装なので，本研究室の自動証明システムを利用した自動判定手続きの実装が今後の課題である．また，基底合流性と十分完全性以外の性質についても変換で保存される条件を明らかにすることも今後の課題である．

参 考 文 献

[1] Standard ML of New Jersey. `http://www.smlnj.org/`.

[2] Baader, F. and Nipkow, T.: *Term Rewriting and*

All That, Cambridge University Press, 1998.

[3] Giesl, J.: Context-moving transformations for function verification, Technical Report IBN 99/51, Darmstadt University of Technology, 1999.

[4] Giesl, J.: Context-moving transformations for function verification, *Proc. LOPSTR 1999*, LNCS 1817, 2000, pp. 293–312.

[5] Klop, J.: *Handbook of Logic in Computer Science*, Vol. 2, Oxford University Press, 1992, pp. 1–112.