帰納的定理自動証明のための項書き換えシステム自動変換

佐藤 洸一 [1], 菊池 健太郎 [1], 青戸 等人 [1], 外山 芳人 [1]

[1] 東北大学 電気通信研究所

{koichi,kentaro,aoto,toyama}@nue.riec.tohoku.ac.jp

概 要 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，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．本論文では，文脈移動法および文脈分割法を項書き換えシステムに対して再定式化し，その正当性を証明する．さらに，これらの変換法と書き換え帰納法 (Reddy,1989) を組み合わせた帰納的定理の自動証明システムを構築する．本システムでは，末尾再帰プログラムにおける定理証明が失敗した場合，変換に必要な文脈交換律等の条件を自動証明し，変換して得られる単純再帰プログラムを用いた定理証明を試みる．

1 はじめに

末尾再帰プログラムは，最後の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，そ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ことが知られている [6].

従来のプログラム変換の研究は，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．そのような変換法として，素朴な関数型プログラムに対する文脈移動法と文脈分割法 [6] 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再帰プログラムへと変換する手法である．変換で得られた再帰プログラムは，計算にアキュムレータを使用せず，帰納法による自動証明に適した構造となっている．

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．本論文では，項書き換えシステムに対して文脈移動法と文脈分割法を再定式化するとともに，そのうえで項書き換えシステムの枠組みに基づき厳密な証明を与える．本論文では，提案手法に基づいた項書き換えシステムの自動変換手続きと，書き換え帰納法 [10] を組み合わせた帰納的定理自動証明システムを実現する．なお，帰納的定理自動証明手続きは，自動変換システムの文脈移動法および文脈分割法の正当性を保証する条件の自動証明にも利用される．実験を通じて，書き換え帰納法による帰納的定理自動証明においては，末尾再帰をもつ項書き換えシステムを単純再帰な項書き換えシステムに自動変換してから書き換え帰納法を適用することがしばしば有効であることを明らかにする．

本論文の構成は以下の通りである．2 節では準備として項書き換えシステムの概念と記法について説明する．3 節では文脈移動法，4 節では文脈分割法による項書き換えシステムの変換手法を示す．5 節では提案手法の正当性を保証する条件の自動検証法を説明する．6 節では提案手法による項書き換えシステムの自動変換実験について説明し，7 節で変換を用いた自動証明について説明する．8 節はまとめと今後の課題を述べる．

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 [3] に従って定義する．

関数記号と変数の集合をそれぞれ $\mathcal{F}, \mathcal{V}$ 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．項の列 $\bar{t}$ についても変数および関数記号全体の集合を $\mathcal{V}(\bar{t}), \mathcal{F}(\bar{t})$ として同様に表す．$\mathcal{V}(t) = \emptyset$ である項 t を基底項とよび，その集合を $\mathcal{T}(\mathcal{F})$ で表す．項 t に出現する変数がすべて異なるとき，項 t は線形であるという．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．$t = C[u]$ 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 は書き換え規則の有限集合である．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．

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

このとき，s は t に書き換えられるという．また，項の列について $\bar{t} \to_R \bar{s}$ は $t_1 \to_R s_1, \cdots, t_n \to_R s_n$ を表す．$s \to_R t$ 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 で得られる正規形であり，一意に定まるものを $t{\downarrow_R}$ で表す．項の列に対しては $\bar{t}{\downarrow_R} = t_1{\downarrow_R}, \ldots, t_n{\downarrow_R}$ と定義される．また，代入 θ に対して $\theta{\downarrow_R}$ を $\theta{\downarrow_R}(x) = (\theta(x)){\downarrow_R}$ と定める．$\to_R$ の反射推移閉包を $\overset{*}{\to}_R$ で表す．項書き換えシステム R のすべての書き換え規則 $l \to r \in R$ の左辺が線形ならば，R は左線形であるという．

R の規則 $l_1 \to r_1$，$l_2 \to r_2$ について，l_1 のある部分項 $u \notin \mathcal{V}$ について $u\theta = l_2\theta$ となる θ が存在するとき $l_1 \to r_1$ と $l_2 \to r_2$ は重なるという．ただし $\mathcal{V}(l_1) \cap \mathcal{V}(l_2) = \emptyset$ とし，$l_1 \to r_1$ と $l_2 \to r_2$ が同じ規則のとき部分項 u は $u \neq l_1$ とする．R を項書き換えシステムとし，任意の項 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 $t_2 \overset{*}{\to} s$ であるとき R は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を $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ と表し，R の構成子記号集合を $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ 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．項 $t \in \mathcal{T}(\mathcal{C})$ を基底構成子項という．値域が基底項集合もしくは基底構成子項集合である代入をそれぞれ基底代入，基底構成子代入といい，θ_g, θ_{gc} 等で表す．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 $s \overset{*}{\to}_R t$ が成立するとき，R は十分完全性 [7] をもつという．**本論文では，変換対象として左線形で重なりがなく，十分完全性をもつ項書き換えシステムを考える．**なお，左線形で重なりをもたない項書き換えシステムは合流性をもつことが知られている [3].

3 項書き換えシステムに対する文脈移動法

本節では，文脈移動法に基づく項書き換えシステムの変換を提案し，一定の条件のもとで変換が正当性をもつことを示す.

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 [6] は，非相互再帰かつ線形再帰である末尾再帰関数について，末尾再帰呼び出しにおけるアキュムレータ位置の移動文脈を，再帰呼び出しの外側に移動する．同様に非相互再帰かつ線形再帰の項書き換えシステム上の関数について文脈移動法を考える．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項が右辺に現れる場合に相当する．よって，そのような項におけるアキュムレータ位置の移動文脈を，その項の外側へ移動することにより，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 R の規則 $l \to r$ において，$root(l) = f$ かつ $f \in \mathcal{F}(r)$ 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．

定義 3.2 (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．

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

R_A, R'_A は R, R' それぞれの f, f' の再帰規則，R_B, R'_B は非再帰規則，R_C は補助関数にあたる規則である．変換対象の関数記号を f とし，文脈 $C_i[\,], C_j[\,]$ を移動文脈，変数 z をアキュムレータとよぶ．変換対象の f とアキュムレータ z は上記の変換規則で明示された場所以外には出現しない．すなわち，以下が成立するものとする．

$$\forall i(1 \le i \le p).f \notin \mathcal{F}(\bar{l}_i, \bar{r}_i, C_i, l_i, r_i) \wedge z \notin \mathcal{V}(\bar{l}_i, \bar{r}_i, C_i, l_i, r_i)$$

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．R' の規則の左辺は R の規則の左辺に出現する f を f' に置き換えたものである．よって，R と同様 R' も左線形かつ重なりがないため，R' は合流性をもつ．

例 3.3 (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

ここでは，文脈移動法による変換の正当性，つまり，R を変換して得られる R' は変換対象 f を f' に置き換えた項を R と同じ結果に書き換えられることを示す．

変換の正当性を保証するため，以下の条件を考える．この条件は [6] で提案された条件をもとに項書き換えシステムの枠組みで再定式化した条件となっている．

定義 3.4 (文脈交換律)**.** $C_1[\ ], \ldots, C_n[\ ]$ を文脈移動法の規則における移動文脈とする．このとき，以下の条件を文脈交換律とよぶ.

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

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 3.5. 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，任意の基底構成子代入 θ_{gc} と基底構成子項 v に対して，

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

が成立する.

証明. $f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v$ 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.

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

項 $f(\bar{t}, u)$ に適用できる R_A の規則は $\bar{t}$ によってのみ定まり，u 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.

$$\begin{aligned}
f(\bar{x}, z)\theta_{gc} &= f(\bar{l}_{i_1}\theta_1, u_1) \\
&\rightarrow_{R_A} f(\bar{r}_{i_1}\theta_1, C_{i_1}\theta_1[u_1]) \\
&\xrightarrow{*}_{R_C} f(\bar{l}_{i_2}\theta_2, C_{i_1}\theta_1[u_1]) \\
&\rightarrow_{R_A} f(\bar{r}_{i_2}\theta_2, C_{i_2}\theta_2[C_{i_1}\theta_1[u_1]]) \\
&\ \ \vdots \\
&\xrightarrow{*}_{R_C} f(\bar{l}_{i_n}\theta_n, C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u_1]\cdots]) \\
&\rightarrow_{R_B} C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u_1]\cdots]]
\end{aligned}$$

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.

$$\begin{aligned}
f'(\bar{x}, z)\theta_{gc} &= f'(\bar{l}_{i_1}\theta_1, u_1) \\
&\rightarrow_{R'_A} C_{i_1}\theta_1[f'(\bar{r}_{i_1}\theta_1, u_1)] \\
&\xrightarrow{*}_{R'_C} C_{i_1}\theta_1[f'(\bar{l}_{i_2}\theta_2, u_1)] \\
&\rightarrow_{R'_A} C_{i_1}\theta_1[C_{i_2}\theta_2[f'(\bar{r}_{i_2}\theta_2, u_1)]] \\
&\ \ \vdots \\
&\xrightarrow{*}_{R'_C} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[f'(\bar{l}_{i_n}\theta_n, u_1)]\cdots] \\
&\rightarrow_{R'_B} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u_1]]\cdots]
\end{aligned}$$

ここで，$u \in \mathcal{T}(\mathcal{C})$，$\forall k.f, f' \notin \mathcal{F}(C_{i_k}\theta_k[\ ])$，$\theta_k{\downarrow}_R = \theta_k{\downarrow}_{R_C} = \theta_k{\downarrow}_{R'}$ となり，R の十分完全性より $\theta_k{\downarrow}_{R_C}$ は基底構成子代入となるので，文脈交換律と合流性より以下が成り立つ．

$$\begin{aligned} v &= C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u_1]\cdots]]{\downarrow}_R \\ &= C_{i_n}\theta_n[C_{i_{n-1}}\theta_{n-1}[\cdots C_{i_1}\theta_1[u_1]\cdots]]{\downarrow}_{R_C} \\ &= C_{i_n}(\theta_n{\downarrow}_{R_C})[\cdots C_{i_1}(\theta_1{\downarrow}_{R_C})[u_1]\cdots]{\downarrow}_{R_C} \\ &= C_{i_1}(\theta_1{\downarrow}_{R_C})[\cdots C_{i_n}(\theta_n{\downarrow}_{R_C})[u_1]\cdots]{\downarrow}_{R_C} \\ &= C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u_1]]\cdots]{\downarrow}_{R_C} \\ &= C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u_1]]\cdots]{\downarrow}_{R'} \end{aligned}$$

よって，以下が成立する．

$$\begin{aligned} f'(\bar{x}, z)\theta_{gc} &\overset{*}{\to}_{R'} C_{i_1}\theta_1[\cdots C_{i_{n-1}}\theta_{n-1}[C_{i_n}\theta_n[u_1]]\cdots] \\ &\overset{*}{\to}_{R'} v \end{aligned}$$

□

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，

$$s \overset{*}{\to}_R v \ \Rightarrow\ s' \overset{*}{\to}_{R'} v$$

が成立する．ここで，s' は s に出現するすべての f を f' で置き換えた項である．

証明． s に関する帰納法によって補題を示す．$g \neq f$ として $s = g(t_1, \ldots, t_m)$ の場合，R の十分完全性より各 i に対して $t_i \overset{*}{\to}_R v_i$ となる基底構成子項 v_i が存在するため，s は $s \overset{*}{\to}_R g(v_1, \ldots, v_m)$ と書き換えられる．ここで仮定と R の合流性から $g(v_1, \ldots, v_m) \overset{*}{\to}_R v$ である．帰納法の仮定からそれぞれの i について ${t_i}' \overset{*}{\to}_{R'} v_i$ が成り立つため $s' = g({t_1}', \ldots {t_m}') \overset{*}{\to}_{R'} g(v_1, \ldots, v_m)$ であり，$g(v_1, \ldots, v_m) \overset{*}{\to}_{R_C} v$ であるため $g(v_1, \ldots, v_m) \overset{*}{\to}_{R'} v$ が成立．$s = f(t_1, \ldots, t_m)$ の場合も同様に各 i に対して $t_i \overset{*}{\to}_R v_i$ となる基底構成子項 v_i が存在し $s \overset{*}{\to}_R f(v_1, \ldots, v_m)$．仮定 $s \overset{*}{\to}_R v$ と R の合流性より，$f(v_1, \ldots, v_m) \overset{*}{\to}_R v$．ここで，帰納法の仮定より各 i について $t_i \overset{*}{\to}_{R'} v_i$ となるので，$s' = f'({t_1}', \ldots, {t_m}') \overset{*}{\to}_{R'} f'(v_1, \ldots, v_m)$．さらに，補題 3.5 より $f'(v_1, \ldots, v_m) \overset{*}{\to}_{R'} v$． □

系 3.7. R' は十分完全性をもつ．

証明． 補題 3.6 と R の十分完全性より成立する． □

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． R と R' の十分完全性と合流性より，$s{\downarrow}_R$ と $s'{\downarrow}_{R'}$ はそれぞれ一意に定まる基底構成子項である．このとき補題 3.6 から $s' \overset{*}{\to}_{R'} s{\downarrow}_R$ であるため $s{\downarrow}_R = s'{\downarrow}_{R'}$ が成り立つ． □

4 項書き換えシステムに対する文脈分割法

本節では，文脈分割法に基づく項書き換えシステムの変換を提案し，一定の条件のもとで変換が正当性をもつことを示す．

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 [6] は，末尾再帰呼び出しにおけるアキュムレータ引数位置の文脈を，各分岐の共通文脈と各分岐固有の部分に分け，共通文脈を再帰呼び出しの外側へ，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位置に出現する文脈を共通文脈と固有の部分項に分け，共通文脈を再帰呼び出しの外側に，固有部分項をアキュムレータ位置に移動する．

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．

$$
\begin{aligned}
R &= R_{A0} \cup R_{A1} \cup R_{B0} \cup R_{B1} \cup R_C \\
R_{A0} &= \{f(\bar{l}_i, z) \to f(\bar{q}_i, C[r_i, z]) && (1 \le i \le m) \\
R_{A1} &= \{f(\bar{l}_{i'}, z) \to f(\bar{q}_{i'}, z) && (m+1 \le i' \le m') \\
R_{B0} &= \{f(\bar{l}_j, z) \to C[r_j, z] && (m'+1 \le j \le n) \\
R_{B1} &= \{f(\bar{l}_{j'}, z) \to z && (n+1 \le j' \le n') \\
R_C &= \{l_k \to r_k && (n'+1 \le k \le p) \\
R &= R'_{A0} \cup R'_{A1} \cup R'_{B0} \cup R'_{B1} \cup R'_C \\
R'_{A0} &= \{f'(\bar{l}_i) \to C[f'(\bar{q}_i), r_i] && (1 \le i \le m) \\
R'_{A1} &= \{f'(\bar{l}_{i'}) \to f'(\bar{q}_{i'}) && (m+1 \le i' \le m') \\
R'_{B0} &= \{f'(\bar{l}_j) \to r_j && (m'+1 \le j \le n) \\
R'_{B1} &= \{f'(\bar{l}_{j'}) \to e && (n+1 \le j' \le n') \\
R'_C &= R_C
\end{aligned}
$$

さらに R_A, R_B, R'_A, R'_B を以下のように定義する．

$$
\begin{aligned}
R_A &= R_{A0} \cup R_{A1} & \qquad R'_A &= R'_{A0} \cup R'_{A1} \\
R_B &= R_{B0} \cup R_{B1} & \qquad R'_B &= R'_{B0} \cup R'_{B1}
\end{aligned}
$$

$R_{A0}, R_{A1}, R'_{A0}, R'_{A1}$ 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} のように表記する．

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 を共通文脈とよび，$\mathcal{V}(C) = \emptyset$ とする．このとき，変換対象の f とアキュムレータ z は上記の変換規則で明示された場所以外には出現しない．すなわち，以下が成立するものとする．

$$\forall i (1 \le i \le p). f \notin \mathcal{F}(\bar{l}_i, \bar{q}_i, l_i, r_i, C) \wedge z \notin \mathcal{V}(\bar{l}_i, \bar{q}_i, r_i, C)$$

また，項 e は共通文脈 C の単位元となる基底構成子項であり，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．R' の規則の左辺は R の規則の左辺に出現する f を f' に置き換え，変換対象の規則で左辺の最後の変数 z を取り除いたものである．よって，R と同様 R' も左線形かつ重なりがないため，R' は合流性をもつ．

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．

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

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．共通文脈 C は $App(\Box_2, \Box_1)$ で，この文脈の単位元は Nil となる．

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

□

4.2 文脈分割法による変換の正当性

ここでは，文脈分割法による変換の正当性，つまり，R を変換して得られる R' は R で書き換える項と対応する項を R と同じ結果に書き換えられることを示す．

変換の正当性を保証するため文脈分割法の変換規則における共通文脈 C と文脈単位元 e について以下の 2 つの条件を考える．これらの条件も文脈交換律と同様に [6] で提案された条件をもとに項書き換えシステムの枠組みで再定式化したものとなっている．

定義 4.3 (文脈結合律)**.** C を文脈分割法の規則における共通文脈とする．このとき，以下の条件を文脈結合律とよぶ．

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

定義 4.4 (文脈単位元)**.** 文脈分割法の規則における共通文脈 C と文脈単位元 e について以下の条件を文脈単位元の条件とよぶ．

$$\forall \theta_{gc}.C[x, e]\theta_{gc}{\downarrow_{R_C}} = C[e, x]\theta_{gc}{\downarrow_{R_C}} = \theta_{gc}(x)$$

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．ここで，C は文脈分割法の変換規則の共通文脈である．このとき，任意の基底構成子代入 θ_{gc} と基底構成子項 v に対して，

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

が成立する．

証明. $f(\bar{x}, z)\theta_{gc} \xrightarrow{*}_R v$ 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．

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

ここで，$u'_k = C[r_{i_k}\theta_k, u_k]$ または $u'_k = u_k$ である．項 $f(\bar{t}, u)$ に適用できる R_A の規則は u 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_k もしくは u'_k 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 $f(\bar{t},u)$ 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．

$$\begin{aligned}
C[f'(\bar{x}),z]\theta_{gc} = \ & C[f'(\bar{l}_{i_1}\theta_1),u_1] \\
& \to_{R'_A} C[D_{i_1}[f'(\bar{q}_{i_1}\theta_1)],u_1] \\
& \overset{*}{\to}_{R'_C} C[D_{i_1}[f'(\bar{l}_{i_2}\theta_2)],u_1] \\
& \to_{R'_A} C[D_{i_1}[D_{i_2}[f'(\bar{q}_{i_2}\theta_2)]],u_1] \\
& \ \ \vdots \\
& \overset{*}{\to}_{R'_C} C[D_{i_1}[\cdots D_{i_{n-1}}[f'(\bar{l}_{i_n}\theta_n)]\cdots],u_1] \\
& \to_{R'_B} C[D_{i_1}[\cdots D_{i_{n-1}}[r'_{i_n}]\cdots],u_1]
\end{aligned}$$

ここで，$r'_{i_n} = r_{i_n}\theta_n$ または $r'_{i_n} = e$ であり，$D_{i_k} = C[\Box, r_{i_k}\theta_k]$ または $D_{i_k} = \Box$ である．また，文脈単位元の条件より $C[e, r_{j_m}\theta'_m]{\downarrow_{R_C}} = r_{j_m}\theta'_m{\downarrow_{R_C}}$ であることから，この書き換え列の最終項は R_C によって以下の項に書き換えられる．

$$C[C[\cdots C[r_{j_m}\theta'_m{\downarrow_{R_C}}, r_{j_{m-1}}\theta'_{m-1}],\cdots,r_{j_1}\theta'_1],u_1]$$

ここで，$u_1 \in \mathcal{T}(\mathcal{C})$，$\forall k.f, f' \notin \mathcal{F}(C, r_{j_k}\theta'_k)$，文脈結合律より

$$\begin{aligned}
v &= C[r_{j_m}\theta'_m, C[r_{j_{m-1}}\theta'_{m-1},\cdots,C[r_{j_1}\theta'_1,u_1]\cdots]]{\downarrow_R} \\
&= C[r_{j_m}\theta'_m, C[r_{j_{m-1}}\theta'_{m-1},\cdots,C[r_{j_1}\theta'_1,u_1]\cdots]]{\downarrow_{R_C}} \\
&= C[r_{j_m}(\theta'_m{\downarrow_{R_C}}),\cdots,C[r_{j_1}(\theta'_1{\downarrow_{R_C}}),u_1]\cdots]{\downarrow_{R_C}} \\
&= C[\cdots C[r_{j_m}(\theta'_m{\downarrow_{R_C}}), r_{j_{m-1}}(\theta'_{m-1}{\downarrow_{R_C}})],\cdots,u_1]{\downarrow_{R_C}} \\
&= C[C[\cdots C[r_{j_m}\theta'_m{\downarrow_{R_C}}, r_{j_{m-1}}\theta'_{m-1}],\cdots,r_{j_1}\theta'_1],u_1]{\downarrow_{R_C}} \\
&= C[C[\cdots C[r_{j_m}\theta'_m{\downarrow_{R_C}}, r_{j_{m-1}}\theta'_{m-1}],\cdots,r_{j_1}\theta'_1],u_1]{\downarrow_{R'}}
\end{aligned}$$

よって，以下が成立する．

$$\begin{aligned}
& C[f'(\bar{x}),z]\theta_{gc} \\
& \overset{*}{\to}_{R'} C[C[\cdots C[r_{j_m}\theta'_m{\downarrow_{R_C}}, r_{j_{m-1}}\theta'_{m-1}],\cdots,r_{j_1}\theta'_1],u_1] \\
& \overset{*}{\to}_{R'} v
\end{aligned}$$

□

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，

$$s \overset{*}{\to}_R v \ \Rightarrow\ s' \overset{*}{\to}_{R'} v$$

が成立する．ここで，s' は s に出現するすべての $f(\bar{t},u)$ を $C[f'(\bar{t}),u]$ で置き換えた項である．

証明. s に関する帰納法によって補題を示す．$g \neq f$ として $s = g(t_1,\ldots,t_m)$ の場合，R の十分完全性より各 i に対して $t_i \overset{*}{\to}_R v_i$ となる基底構成子項 v_i が存在するため，s は $s \overset{*}{\to}_R g(v_1,\ldots,v_m)$ と書き換えられる．ここで仮定と R の合流性から $g(v_1,\ldots,v_m) \overset{*}{\to}_R v$ である．帰納法の仮定からそれぞれの i について ${t_i}' \overset{*}{\to}_{R'} v_i$ が成り立つため $s' = g({t_1}',\ldots {t_m}') \overset{*}{\to}_{R'} g(v_1,\ldots,v_m)$ であり，$g(v_1,\ldots,v_m) \overset{*}{\to}_{R_C} v$ であるため $s' \overset{*}{\to}_{R'} v$ が成立．$s = f(t_1,\ldots,t_m)$ の場合も同様に各 i に対して $t_i \overset{*}{\to}_R v_i$ となる基底構成子項 v_i が存在し，$s \overset{*}{\to}_R f(v_1,\ldots,v_m)$．仮定 $s \overset{*}{\to}_R v$ と R の合流性より，$f(v_1,\ldots,v_m) \overset{*}{\to}_R v$．ここで，補題 4.5 より $C[f'(v_1,\ldots,v_{m-1}),v_m] \overset{*}{\to}_{R'} v$．したがって，帰納法の仮定より $s' = C[f'({t_1}',\ldots,{t_{m-1}}'),{t_m}'] \overset{*}{\to}_{R'} C[f'(v_1,\ldots,v_{m-1}),v_m] \overset{*}{\to}_{R'} v$ ． □

系 4.7. R' は十分完全性をもつ．

証明． s' を R' の基底項とし，s' に出現するすべての $f'(\bar{t})$ を $f(\bar{t}, e)$ で置き換えた項を s とする．R の十分完全性より $s \xrightarrow{*}_R u$ となる基底構成子項 u が存在する．補題 4.5 の証明と同様にして，$\theta_{gc}(z) = e$ となる任意の基底構成子代入 θ_{gc} に対して，

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

を示すことができる．よって，$s' \xrightarrow{*}_{R'} u$ が成り立つ． □

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 4.8 (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' が得られたとし，C を変換規則における共通文脈，e を共通文脈の単位元とする．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 $f'(\bar{t})$，$u \neq e$ である u についてすべての $f(\bar{t}, u)$ を $C[f'(\bar{t}), u]$ に置き換えた項である．

証明． R と R' の十分完全性と合流性より，$s{\downarrow_R}$ と $s'{\downarrow_{R'}}$ はそれぞれ一意に定まる基底構成子項である．このとき補題 4.6 より s に出現するすべての $f(\bar{t}, u)$ を $C[f'(\bar{t}), u]$ で置き換えた項 s'' について $s'' \xrightarrow{*}_{R'} s{\downarrow_R}$ である．さらに，s に出現するすべての $f(\bar{t}, u)$ を $u = e$ ならば $f'(\bar{t}){\downarrow_{R'}}$，$u \neq e$ ならば $C[f'(\bar{t}), u]$ に置き換えた項 s''' を考えると，文脈単位元の条件，系 4.7 より書き換え $s'' \xrightarrow{*}_{R'} s'''$ が得られる．一方，R' の定義より $s' \xrightarrow{*}_{R'} s'''$ であるため，R' の合流性より $s' \xrightarrow{*}_{R'} s{\downarrow_R}$．よって，$s{\downarrow_R} = s'{\downarrow_{R'}}$ が成立する． □

5 変換の正当性を保証する条件の自動検証

本節では文脈移動法，文脈分割法の正当性を保証する条件である文脈交換律，文脈結合律，文脈単位元の条件の自動証明に用いる書き換え帰納法について説明する．

定義 5.1 (帰納的定理 [10])**.** 任意の基底代入 σ_g について $s\sigma_g \overset{*}{\leftrightarrow}_R t\sigma_g$ が成り立つとき，等式 $s \doteq t$ は帰納的定理であるといい，$R \models_{\mathrm{ind}} s \doteq t$ と記す．また，等式集合 E の全ての等式がすべて R の帰納的定理であるとき E を R の帰納的定理であるといい，$R \models_{\mathrm{ind}} E$ と記す．

帰納的定理の定義より，以下の性質がいえる．

補題 5.2. 十分完全性，合流性をもつ項書き換えシステム R について，$R \models_{\mathrm{ind}} s \doteq t$ が成立するならば，任意の基底構成子代入 σ_{gc} について $s\sigma_{gc}{\downarrow_R} = t\sigma_{gc}{\downarrow_R}$ が成り立つ．

本論文では，変換を適用する項書き換えシステムに十分完全性を仮定しており，左線形で重なりがないことから合流性が言える．よって，文脈移動法において任意の $i, j (1 \leq i \leq m, 1 \leq j \leq n)$ について $R_C \models_{\mathrm{ind}} C_i[C_j[z]] \doteq C_j[C_i[z]$ が成立するならば，補題 5.2 により任意の θ_{gc} について文脈交換律 $C_i[C_j[z]]\theta_{gc}{\downarrow_{R_C}} = C_j[C_i[z]]\theta_{gc}{\downarrow_{R_C}}$ が成立する．文脈分割法においても同様に，対応する帰納的定理が成立するならば，補題 5.2 より文脈結合律および文脈単位元の条件が成立する．このことから，文脈交換律，文脈結合律，文脈単位元の条件を示すには，対応する等式が帰納的定理であることを証明すればよいことがわかる．

帰納的定理の自動証明法として書き換え帰納法 [10] が提案されている．本研究では，青戸 [2] によって拡張された書き換え帰納法を，変換の正当性を保証する条件の自動証明に用いる．書き換え帰納法は，等式集合 E，書き換え規則集合 H，等式集合 K の組 $\langle E, H, K\rangle$ に対する推論規則として与えられる [2]．E は証明すべき等式，H は帰納法の仮定または証明された定理，K は簡約化順序がつけられない等式集合となる．$\langle E, \emptyset, \emptyset\rangle$ に書き換え帰納法の推論規則を適用し，$\langle \emptyset, H, K\rangle$ が得られたならば $R \models_{\mathrm{ind}} E$，推論規則 *Disproof* が適用されたならば反証成功であり $R \not\models_{\mathrm{ind}} E$ である [2]．

本研究で実装を行う書き換え帰納法の手続きを以下に示す.

書き換え帰納法の手続き

入力：証明する等式 $s \doteq t$，十分完全性をもつ項書き換えシステム R，関数記号上の擬順序関係 $\gtrsim_{\mathcal{F}}$

出力：$s \doteq t$ が帰納的定理であることを証明できた場合 $true$，帰納的定理でない場合 $false$

1. 擬順序関係 $\gtrsim_{\mathcal{F}}$ から生成される辞書式経路擬順序より R の停止性を判定する．失敗した場合は終了する．
2. 入力された R の合流性を危険対解析に基づき判定する．失敗した場合は終了する．
3. 書き換え帰納法の規則に対応する組を $\langle E, H, K\rangle = \langle\{s \doteq t\}, \emptyset, \emptyset\rangle$ として初期化する．
4. $\langle E, H, K\rangle$ に書き換え帰納法の推論規則を適用する．E が空になった場合，証明は成功して $true$ を返す．反証に成功した場合 $false$ を返す．
5. 4 を繰り返す．ただし，一定の回数 n 以上繰り返された場合はその時点で打ち切り，失敗したことを表示したのち終了する．

6　項書き換えシステムの自動変換

本節では，これまでで説明した項書き換えシステムの変換手法に基づき実装した自動変換システムについて述べる．また，自動変換実験により本システムが項書き換えシステムを正しく変換できることを確かめる．

6.1　項書き換えシステムの自動変換手続き

実装した文脈移動法および文脈分割法による変換手続きを以下に示す.

文脈移動法・文脈分割法による変換手続き

入力：十分完全性をもつ項書き換えシステム R，変換対象の関数記号 f，関数記号上の擬順序関係 $\gtrsim_{\mathcal{F}}$

出力：R を変換して得られる R'

1. 擬順序関係 $\gtrsim_{\mathcal{F}}$ から生成される辞書式経路擬順序より R の停止性を判定する．失敗した場合は終了する．
2. 入力された R の合流性を危険対解析に基づき判定する．失敗した場合は終了する．
3. R が変換適用可能な形か否かの判定を行う．適用可能な形でない場合は終了する．このとき，文脈移動法であれば移動文脈，文脈分割法であれば共通文脈を抽出する．文脈分割法で共通文脈となりうる文脈が複数存在する場合，そのうち最大の文脈を共通文脈とする．
4. 得られた文脈から変換の正当性を保証する条件を調べる．いずれの場合も証明に失敗した場合は終了する．

   文脈移動法の場合 すべての $i, j (1 \leq i \leq m, 1 \leq j \leq n)$ について R_C，$>_{\mathcal{F}}$ を用いた書き換え帰納法により $R_C \models_{\mathrm{ind}} C_i[C_j[z]] \doteq C_j[C_i[z]]$ を証明する．

   文脈分割法の場合 R_C，$>_{\mathcal{F}}$ を用いた書き換え帰納法により $R_C \models_{\mathrm{ind}} C[C[x, y], z] \doteq C[x, C[y, z]]$ を証明する．さらに，$R_C \models_{\mathrm{ind}} C[x, e] \doteq x, C[e, x] \doteq x$ をみたす文脈単位元 e の候補を生成し，書き換え帰納法により証明する．予め定められたサイズ以下の e の候補すべてについて証明に失敗した場合終了する．

5. 適用する変換手法の規則に従い R より R' を生成し出力する.

以上の変換手続きを用いて，項書き換えシステムの自動変換実験，書き換え帰納法による帰納的定理の自動証明実験を行う.

6.2 自動変換システムの実装

文脈移動法・文脈分割法による項書き換えシステム変換手続きと，書き換え帰納法に基づく帰納的定理自動証明手続きを，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，変換適用可能な形か否かの判定手続きと実際の変換手続き，文脈交換律の書き換え帰納法による判定手続きを実装した．文脈分割法による項書き換えシステムの変換手続きについても同様に変換適用可能な形か否かの判定手続き，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書き換え帰納法による判定手続きを実装した．実装した行数は文脈移動法が約 80 行，文脈分割法が約 120 行，書き換え帰納法が約 250 行となっている．以下に自動変換システムの動作例として，自然数の乗算を行う項書き換えシステムを変換した例を示す.

```
- printrules mult;
[ Mult(0, y, z) -> z,
  Mult(S(x), y, z) -> Mult(x, y, Add(y, z)),
  Add(0, y) -> y,
  Add(S(x), y) -> S(Add(x, y)) ]

- val mult' = CM.context_moving mult "Mult" order;
transformation finished : 4[ms]

- printrules mult';
[ Mult'(S(x), y, z) -> Add(y, Mult'(x, y, z)),
  Mult'(0, y, z) -> z,
  Add(0, y) -> y,
  Add(S(x), y) -> S(Add(x, y)) ]
```

6.3 自動変換実験

本システムによる項書き換えシステムの変換実験結果と，その実行時間 (単位ミリ秒) を表 1 に示す．実験に用いた例は，総減算の機能をもつ項書き換えシステム以外は文献 [5] で与えられたプログラムを項書き換えシステムに置き換えたものである．[5] のプログラムのうち，項書き換えシステムで自然に表現することが可能である例を実験例として選んだ．変換結果は，○は変換成功，× は変換失敗，$\times^*$ は正当性を保証する条件をみたすが書き換え帰納法による証明が失敗をそれぞれ表す．また，実行時間が 10 秒を超えた場合はタイムアウトとして実験を打ち切った.

全 26 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 18 例，失敗例 8 例となり，文脈分割法の成功例が 12 例，失敗例 14 例となった．両方で成功した例は 7 例であった．また，書き換え帰納法による条件の証明が停止しない例が文脈移動法において 2 例，文脈分割法において 4 例見られた．本実験では，7 割以上の例についていずれかの変換法で変換に成功した．よって，書き換え帰納法に基づく帰納的定理証明と組み合わせることで全自動変換は可能であるといえる．また，実験では文脈移動法か文脈分割法どちらかしか成功しない例が多く見られた．このことから，変換を用いた自動証明においても，両者の併用が有効であると考えられる．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，条件を満たさず失敗する例，条件は正しいが証明に失敗する例からそれぞれの代表的な例を示す.

表 1. 項書き換えシステム変換の実験結果

	文脈移動法		文脈分割法	
R	変換結果	実行時間 [ms]	変換結果	実行時間 [ms]
加算	○	0	×	0
乗算	○	2	○	0
乗算 (2)	○	0	×	0
2 倍	○	0	×	0
1/2	○	0	×	0
二進対数	○	0	×	0
2 の冪	○	0	×	0
除算	○	0	×	0
リスト長	○	0	×	0
総減算	○	12	×	0
三角数	○	3	○	0
リスト和	○	2	○	0
重み付き総和	○	106	○	0
階乗	×*	タイムアウト	×*	2
冪乗	×*	タイムアウト	×*	2
最大要素	○	0	×*	10
最小要素	○	0	×*	3
Member	○	25	○	6
Subset	○	5	○	6
スカラー積	○	236	○	0
リスト結合	×	2	○	0
リスト反転	×	0	○	0
Filter	×	1	○	0
First	×	0	○	0
Increment	×	1	○	0
減算	×	0	×	0

例 6.1 (両方成功)**.** 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 でリスト和，Member 関数などである．これらはいずれも加算，論理演算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 6.2 (文脈移動法のみ成功)**.**

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

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．

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

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する．実際の変換では文脈結合律を表す帰納的定理 $R_C \models_{\mathrm{ind}} Minus(Minus(x, y), z) \doteq Minus(x, Minus(y, z))$ の書き換え帰納法による証明において自動変換システムが $false$ を返し終了した． □

例 6.3 (文脈分割法のみ成功)**.**

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

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．

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

文脈分割法のみ成功する例において共通文脈はどの例でも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．変換手続きでは手続きの 4 で $R_C \models_{\mathrm{ind}} Append(Append(x, y), z) \doteq Append(Append(x, z), y)$ の証明の際自動変換システムが $false$ を返し終了していた． □

例 6.4 (両方失敗)**.**

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

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 R の 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．変換手続きにおいては 3 の段階で変換適用可能な形でないと判定され手続きが終了した．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． □

例 6.5 (条件は成立するが証明に失敗)**.**

$$R = \begin{cases} Maxlist(Nil, z) \to z \\ Maxlist(Cons(x, xs), z) \to Maxlist(xs, Max(x, z)) \end{cases}$$

最大要素 $Maxlist(l, 0)$ は自然数のリスト l から最大の要素を取り出す．文脈移動法において文脈交換律を示すため書き換え帰納法を適用する等式は $Max(x, Max(y, z)) = Max(y, Max(x, z))$，文脈分割法における文脈結合律は $Max(Max(x, y), z) = Max(x, Max(y, z))$ となるが，文脈結合律の証明において書き換え帰納法の手続きが一定回数以上繰り返されたため変換に失敗した． □

7　変換を利用した帰納的定理の自動証明

本節では，末尾再帰規則を持つ項書き換えシステム R に基づく書き換え帰納法では証明が困難な帰納的定理が，文脈移動法および文脈分割法を用いて R を R' に変換し，R' に基づく書き換え帰納法を用いることで証明できる場合があることを明らかにする．

本研究で実装した変換を組み合わせた書き換え帰納法による証明手続きを以下に示す．

変換を組み合わせた書き換え帰納法の手続き

入力：証明する等式 $s \doteq t$，十分完全性をもつ項書き換えシステム R，関数記号上の擬順序関係 $\gtrsim_{\mathcal{F}}$

出力：$s \doteq t$ が帰納的定理であることを証明できた場合 $true$，帰納的定理でない場合 $false$

1. 書き換え帰納法による $R \models_{\mathrm{ind}} s \doteq t$ の証明を 5 節の手続きを用いて試みる．帰納的定理かそうでないかが定まった場合その結果を返して終了する．
2. R そのままでの証明に失敗した場合，R に含まれる規則の形から末尾再帰で定義された関数の規則集合を探す．発見できなかった場合失敗したことを出力し終了する．
3. 2 で発見した末尾再帰規則を文脈移動法が適用可能であれば文脈移動法，そうでない場合文脈分割法が適用可能であれば文脈分割法によって変換を行い新たな項書き換えシステム R' を得る．変換が全くできない場合は失敗したことを出力し終了する．
4. 3 で得られた R' によって $R' \models_{\mathrm{ind}} s' \doteq t'$ の証明を試みる．ここで s', t' はそれぞれ s, t に出現する項 $f(\bar{t}, s)$ をすべて 3 で適用した変換に応じて対応する項 (文脈移動法を適用した場合 $f'(\bar{t}, s)$，文脈分割法の場合共通文脈を C，単位元を e として $s = e$ ならば $f'(\bar{t})$，$s \neq e$ ならば $C[f'(\bar{t}), s]$) に置き換えた項である．帰納的定理であるかそうでないかが定まれば $true$ か $false$ を返し，失敗した場合は失敗したことを出力して終了する．

この手続きにより自動定理証明の実験を行った結果を表 2 に示す．表 2 は実験を行った 34 例を示し，変換前と変換後ともに証明に成功した例が 8 例，変換後に証明が成功した例が 14 例，変換後も証明に失敗した例が 12 例となっている．各実験例については証明の対象となる帰納的定理，末尾再帰規則で規則を定義した関数記号，末尾再帰規則を単純再帰規則に変換する前後の証明結果を示している．帰納的定理はいずれも予め正しいことが確かめられたものであるため，成功例では関数が $true$ を返し失敗例では証明が打ち切られるという結果となっている．

表 2 の実験結果より，証明に失敗する自明でない帰納的定理が項書き換えシステムの変換によって証明可能となる例を多数確認できた．したがって，書き換え帰納法に基づく帰納的定理の自動証明において，項書き換えシステムの変換がしばしば有効となる場合があるといえる．

以下では変換前と変換後の証明結果ごとに代表的な例を示す．

例 7.1 (変換前・変換後成功)**.** 変換前・変換後ともに書き換え帰納法による証明が成功する帰納的定理 $Mult(x, 0, 0) \doteq 0$ は以下の項書き換えシステム R(Add の規則省略) に基づく証明を行った．

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

末尾再帰関数の帰納法による証明の問題点は，アキュムレータ引数が再帰呼び出しの度に変化するため，帰納法の仮定を用いることができないことである．上記の例では，$Mult$ の第三引数 z が $Add(y, z)$ に変化している．今回証明すべき帰納的定理では $y = 0$ であるため，$Add(0, z) \to_R z$ なので実質的に z は変化しない．よって，帰納法の仮定を適用できるため証明が成功した．変換前と変換後でともに証明が成功する例では，同様に実質的にアキュムレータが変化しない例や，通常の書き換えのみで証明できる例がほとんどであった．　□

表 2. 証明実験の結果

帰納的定理	末尾再帰関数	変換前	変換法	変換後
$Plus(0,x) \doteq x$	$Plus$	成功	移動	成功
$Plus(x,y) \doteq Plus(y,x)$	$Plus$	成功	移動	成功
$Plus(Plus(x,y),z) \doteq Plus(x,Plus(y,z))$	$Plus$	成功	移動	成功
$Mult(S(0),x,0) \doteq x$	$Mult$	成功	移動	成功
$Mult(x,0,0) \doteq 0$	$Mult$	成功	移動	成功
$Mult(x,0,0) \doteq Mult(0,x,0)$	$Mult$	成功	移動	成功
$Sum(Cons(0,xs),0) \doteq Add(0,Sum(xs,0))$	Sum	成功	移動	成功
$Length(App(Nil,x),0) \doteq Length(x,0)$	$Length$	成功	移動	成功
$Add(x,0) \doteq x$	Add	失敗	移動	成功
$Add(x,Minus(x,x)) \doteq x$	Add	失敗	移動	成功
$Mult(x,S(0),0) \doteq x$	$Mult$	失敗	移動	成功
$Double(S(x),0) \doteq Add(Double(x,0),S(S(0)))$	$Add, Double$	失敗	移動	成功
$Double(x,0) \doteq Mult(x,S(S(0)),0)$	$Mult, Double$	失敗	移動	成功
$Half(Double(x,0),0) \doteq x$	$Double, Half$	失敗	移動	成功
$S(S(Mult(x,S(S(0)),0))) \doteq Double(S(x),0)$	$Mult, Double$	失敗	移動	成功
$Quot(x,S(0),S(0),0) \doteq x$	$Quot$	失敗	移動	成功
$Length(App(x,Nil),0) \doteq Length(x,0)$	$Length$	失敗	移動	成功
$Length(App(x,y),0) \doteq Add(Length(x,0),Length(y,0))$	$Length$	失敗	移動	成功
$Sum(Cons(x,y),0) \doteq Add(x,Sum(y,0))$	Sum	失敗	移動	成功
$Sum(App(xs,ys),0) \doteq Add(Sum(xs,0),Sum(ys,0))$	Sum	失敗	移動	成功
$Rev(App(x,y),Nil) \doteq App(Rev(y,Nil),Rev(x,Nil))$	Rev	失敗	分割	成功
$Length(xs) \doteq Length(Rev(xs,Nil))$	Rev	失敗	分割	成功
$Minus(Plus(x,x),x) \doteq x$	$Plus$	失敗	移動	失敗
$Mult(x,Add(y,z)) \doteq Add(Mult(x,y),Mult(x,z))$	Add	失敗	移動	失敗
$Add(x,x) \doteq Mult(x,S(S(0)),0)$	$Mult$	失敗	移動	失敗
$Quot(Mult(x,S(y),0),S(y),S(y),0) \doteq x$	$Quot, Mult$	失敗	移動	失敗
$Double(x,0) \doteq Add(x,x)$	$Double$	失敗	移動	失敗
$Double(x,0) \doteq Mult(S(S(0)),x)$	$Double$	失敗	移動	失敗
$Log(Exp(S(S(0)),x,S(0)),0) \doteq x$	Log, Exp	失敗	移動	失敗
$Length(App(xs,xs),Nil) \doteq Double(Length(xs),Nil)$	$Length$	失敗	移動	失敗
$Suball(x,Sum(x)) \doteq 0$	$Suball$	失敗	移動	失敗
$Rev(Rev(xs,Nil),Nil) \doteq xs$	Rev	失敗	分割	失敗
$Tri(S(x),0) \doteq Quot(Mult(S(x),S(S(x))),S(S(0)),S(S(0)),0)$	Tri	失敗	移動	失敗
$Half(Length(Cons(0,xs))) \doteq Length(Fir(xs,Nil))$	Fir	失敗	移動	失敗

例 7.2 (変換前失敗・変換後成功)**.** 帰納的定理 $Half(Double(x,0),0) \doteq x$ は以下の項書き換えシステム R に基づく証明を行う.

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

変換後に証明が成功する例では, 変換後アキュムレータが変化しない形となり, 帰納法の仮定が適用可能となる. 書き換え帰納法において帰納法の仮定に相当する規則は, この場合 $Half(Double(x,0),0) \to x$ である. 帰納ステップで証明すべき等式は $Half(Double(S(x),0),0) \doteq S(x)$ となる. このとき, R による書き換えでは $Double$ の第二引数が変化するため, 帰納法の仮定の規則を適用することができない. 文脈移動法による変換を $Double, Half$ それぞれに適用して得られる R' においては書き

換え $Half(Double(S(x),0),0) \xrightarrow{*}_{R'} S(Half(Double(x,0),0))$ により，帰納法の仮定が適用できる形となる．変換後に成功する例は，いずれもこのようなアキュムレータが変化しない形の規則となるため帰納法の仮定が適用できる例となった． □

例 7.3 (変換前・変換後失敗)**.** 変換前と変換後ともに失敗する例では，多くの例が同じ変数を複数もつ項を左辺か右辺にもっている．たとえば，帰納的定理 $Double(x,0) \doteq Add(x,x)$ の証明では，帰納法の仮定を $Double(x,0) \to Add(x,x)$ とすると帰納ステップに相当する等式 $Double(S(x),0) \doteq Add(S(x),S(x))$ を示すこととなる．帰納法の仮定を左辺に一度用いることで証明は成功するが，R と帰納法の仮定の仮定を用いた書き換えの際適切な書き換えを行わなければ両辺は等しくならない．これは変換によって解決する問題ではなく，現在のシステムにはそのような機能を実装していないため同じ変数を複数もつ例の多くで変換後も失敗という結果となった． □

8 まとめと今後の課題

本研究では，関数型プログラムについて提案されていた自動証明のための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を対象に再定式化し，その正当性を証明した．また，変換対象に仮定した十分完全性は変換後も保存されることを示した．文献 [6] で与えられた変換では，変換条件の自動証明についての検討は十分なされていなかった．本論文では，提案手法に基づいた自動変換手続きと，書き換え帰納法に基づく帰納的定理自動証明手続きを実装し，これらを組み合わせることで項書き換えシステムの自動変換を実現した．また，書き換え帰納法による自動証明において，本手法の項書き換えシステムの自動変換が証明能力の向上に有効であることを明らかにした．

本論文では，変換の対象を左線形で重なりがなく，十分完全性をもつ項書き換えシステムに制限しているが，その制限を緩めることは今後の課題である．また，文献 [4] で提案されているパターンに基づく項書き換えシステム変換は文脈移動法・文脈分割法とよく似た変換が可能であるため，本研究との関係を考察することも課題として挙げられる．効率化のためのさまざまなプログラム変換法 [8, 9] を参考に，より強力な自動証明のための項書き換えシステム変換法を開発することも興味深い問題である．

謝辞

本論文に大変貴重なコメントを頂きました査読者に感謝いたします．なお，本研究は一部日本学術振興会科学研究費 25330004，25280025，23500002 の補助を受けて行われた．

参考文献

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

[2] T. Aoto. Designing a rewriting induction prover with an increased capability of non-orientable theorems. In *Proc. of SCSS*, RISC Technical Report, No. 08-08, pp. 1–15, 2008.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[4] Y. Chiba, T. Aoto and Y. Toyama. Program transformation by templates based on term rewriting. In *Proc. of 7th PPDP*, pp. 59–69, 2005.

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

[6] J. Giesl. Context-moving transformations for function verification. In *Proc. of 9th LOPSTR*, LNCS, Vol. 1817, pp. 293–312, 2000.

[7] P. Narendran, D. Kapur and H. Zhang. On sufficient-completeness and related properties of term rewriting systems. *Acta Inf.*, Vol. 24, No. 4, pp. 395–415, 1987.

[8] S. Katsumata and S. Nishimura. Algebraic fusion of functions with an accumulating parameter and its improvement. *J. Funct. Program.*, Vol. 18, No. 5–6, pp. 781–819, 2008.

[9] A. Morihata. Manipulating accumulative functions by swapping call-time and return-time computations. *J. Funct. Program.*, Vol. 22, No. 3, pp. 275–299, 2012.

[10] U.S. Reddy. Term rewriting induction. In *Proc. of 10th CADE*, LNAI, Vol. 449, pp. 162–177, 1990.