決定手続きを用いた
項書き換えシステムの帰納的定理自動証明

山口 諒, 青戸 等人

新潟大学大学院自然科学研究科
{yamaguchi@nue., aoto@}ie.niigata-u.ac.jp

概 要　等式論理において，自然数やリストなどのデータ構造上で成立する等式を帰納的定理という．帰納的定理の自動証明法として，項書き換えシステムを用いた書き換え帰納法が知られている (Reddy, 1990)．また，自然数の乗加算やリスト構造のいくつかの演算を用いた等式については帰納的定理の決定手続きが知られている (Aoto&Stratulat,2014)．本論文では，書き換え帰納法と帰納的定理の決定手続きを融合した帰納的定理自動証明法を提案する．提案手法では，等式に使用されているソート情報を推定し、必要な等式公理を満たす関数記号を書き換え帰納法と推定したソート情報を用いて検出することで，予めどのような関数記号が用いられているかといった情報がない場合や，異なる公理が用いられている場合にも，適用可能な決定手続きを適用する．提案手法を実装するとともに，いくつかの具体例に対して実験を行い，提案手法の有効性を確認した．

1　はじめに

プログラムの性質のうちには，自然数やリストといったデータ構造に関する帰納法で証明されるような性質がある．例えば，リストを連結する append 関数は結合性をもつが，この性質はリストの構造に関する帰納法を用いることで証明することが出来る．等式論理の枠組みでは，このような性質は帰納的定理という概念でとらえられており，書き換え帰納法や潜在帰納法といった，帰納的定理の自動証明法が知られている [3, 5].

一方，Aoto と Stratulat[2] により，いくつかの特定の項書き換えシステムについて，帰納的定理が決定可能であることが示されている．文献 [2] の手法は書き換え帰納法が失敗する場合にも成功するが，適用範囲は限定的である．一方，書き換え帰納法は失敗することもあるが，適用範囲は広い．そこで，本論文では，書き換え帰納法と文献 [2] の決定手続きを組み合わせた，強力な帰納的定理の自動証明システムの実現を提案する．

文献 [2] で提案されている帰納的定理の決定手続きは，関数記号を固定した特定の項書き換えシステムに対して機能する．このため，書き換え帰納法などの一般的な証明法において，この決定手続きを利用するには，その特定の理論に用いる関数記号を前もって知る必要がある．そこで提案手法では，書き換え帰納法を用いることで，どのような関数記号が必要となる公理を満たすかを検証する．次に，関数記号を推定できたら，決定手続きが適用可能な場合には決定手続きを，そうではない場合には書き換え帰納法を適用することで，書き換え帰納法と文献 [2] の手法の組み合わせを実現する．

2　準備

2.1　項書き換えシステム

本論文で用いる定義及び記法を紹介する．

ソートの集合を $\mathcal{S}$，多ソート関数記号集合を $\mathcal{F}$ と表す．関数記号 f のソートを $\alpha_1, \alpha_2, \ldots, \alpha_n \to \beta$ と表す．以下では簡単のため，単一ソートを持つ場合について説明する．変数集合を $\mathcal{V}$ と表わし，項の集合を $T(\mathcal{F}, \mathcal{V})$ と記す．項 t の変数集合を $\mathcal{V}(t)$ と表し，項 t の根記号を $root(t)$ と記す．

関数 $\sigma : \mathcal{V} \to T(\mathcal{F}, \mathcal{V})$ で集合 $\{x \mid \sigma(x) \neq x\}$ が有限であるものを代入とよぶ．また，集合 $\{x \mid \sigma(x) \neq x\}$ を代入 σ の定義域とよび，$dom(\sigma)$ と記す．項 s と t の最汎単一化子を $\mathsf{mgu}(s, t)$ と記す．

文脈とは，ホールとよばれる特別の定数 $\Box$ が一度現れる項のことをいう．また，文脈 $C[\,]$ のホールを項 t に置き換えて得られる項を $C[t]$ と記す．$s = C[t]$ となるとき，t を s の部分項とよぶ．

2 つの項 l, r が，$l \notin \mathcal{V}$，$\mathcal{V}(r) \subseteq \mathcal{V}(l)$ を満たすとき，$l \to r$ を書き換え規則とよぶ．書き換え規則の有限集合を項書き換えシステムとよぶ．書き換え規則 $l \to r \in \mathcal{R}$，代入 σ，文脈 $C[\,]$ が存在して，$s = C[l\sigma]$ かつ $t = C[r\sigma]$ となるとき，$s \to_{\mathcal{R}} t$ と記し，書き換えステップとよぶ．$s \to_{\mathcal{R}} t$ となるような t が存在しないとき，項 s を正規形とよぶ．$\to_{\mathcal{R}}$ の反射推移閉包を $\overset{*}{\to}_{\mathcal{R}}$，$\to_{\mathcal{R}}$ の等価閉包を $\overset{*}{\leftrightarrow}_{\mathcal{R}}$ と記す．

項 t が $\mathcal{V}(t) = \emptyset$ を満たすとき，t を基底項とよぶ．関数記号集合 $\mathcal{F}$ 上の基底項の集合を $T(\mathcal{F})$ と記す．また，任意の $x \in dom(\sigma_g)$ について $\sigma_g(x) \in T(\mathcal{F})$ となっているような代入 σ_g を基底代入とよぶ．$\mathcal{V}(t) \subseteq dom(\sigma_g)$ となっているとき，項 $t\sigma_g$ を基底具体化とよぶ．等式 $s\sigma_g \approx t\sigma_g$ が等式 $s \approx t$ の基底具体化であるとは，$\mathcal{V}(s) \cup \mathcal{V}(t) \subseteq dom(\sigma_g)$ となっているときをいう．

関数記号を定義記号と構成子に分け，$\mathcal{R}$ の定義記号の集合 $\mathcal{D}$ を，$\mathcal{D} = \{root(l) \mid l \to r \in \mathcal{R}\}$，構成子の集合 $\mathcal{C}$ を，$\mathcal{C} = \mathcal{F} \backslash \mathcal{D}$ と定義する．等式集合 E に出現する関数記号を $\mathcal{F}(E)$ と表す．

ある $f \in \mathcal{D}$ と $t_1, \ldots, t_n \in T(\mathcal{C}, \mathcal{V})$ が存在して $t = f(t_1, \ldots, t_n)$ となるとき，項 t が基本項であるといい，基本項の集合を $B(\mathcal{D}, \mathcal{C}, \mathcal{V})$ と記す．また，基底基本項の集合を $B(\mathcal{D}, \mathcal{C})$ と記し，どの基底基本項も正規形ではないとき，$\mathcal{R}$ を擬簡約であるという．項 t の部分項のうち基本項であるものを基本部分項とよぶ．また，$\mathcal{B}(s)$ は s の基本部分項の集合を表わす．

文脈と代入に関して閉じた関係を書き換え関係とよび，整礎な書き換え関係を簡約関係とよぶ．簡約関係が半順序 (非反射的かつ推移的) のとき，簡約順序とよぶ．

2.2 帰納的定理

等式 $s \approx t$ が項書き換えシステム $\mathcal{R}$ の帰納的定理であるとは，$s \approx t$ の任意の基底具体化 $s\theta_g \approx t\theta_g$ について，$s\theta_g \overset{*}{\leftrightarrow}_{\mathcal{R}} t\theta_g$ が成立するときをいう．等式 $s \approx t$ と $t \approx s$ は区別しない．また，等式集合 E のすべての要素が $\mathcal{R}$ の帰納的定理であるとき，$\mathcal{R} \models_{ind} E$ と表す．

例 2.1 (帰納的定理) 次の項書き換えシステム $\mathcal{R}$ を考える．

$$\mathcal{R} = \left\{ \begin{array}{ll} +(0, y) & \to y \\ +(s(x), y) & \to s(+(x, y)) \end{array} \right\}$$

このとき，$+(+(x, y), z) \approx +(x, +(y, z))$ は，$\mathcal{R}$ の帰納的定理となる．よって，$\mathcal{R} \models_{ind} \{+(+(x, y), z) \approx +(x, +(y, z))\}$ が成立する．

2.3 書き換え帰納法

書き換え帰納法は Reddy[3] により提案された帰納的定理の自動証明法である．書き換え帰納法では，等式集合 E と項書き換え規則集合 H の対 $\langle E, H \rangle$ に関する導出を行いながら，証明を行う．表 1 に書き換え帰納法の入出力を，図 1 に書き換え帰納法の導出規則を示す．ここで，Expand 規則に用いられる関数 *Expd* は以下のように定義される．

$$Expd_u(s, t) = \{C[r]\sigma \approx t\sigma \mid s = C[u], \sigma = \mathsf{mgu}(u, l), l \to r \in \mathcal{R}\}$$

$\uplus$ は直和を表す．等式の向きは区別しない (したがって，Simplify 規則は，等式の左辺も右辺も書き換えを行う)．$\mathcal{R}$ および $>$ は入力として与えられる項書き換えシステムと簡約順序である．

表 1. 書き換え帰納法の入出力

入力	擬簡約な項書き換えシステム $\mathcal{R}$, 等式集合 E, $\mathcal{R} \subseteq >$ なる簡約順序 $>$
出力	“証明成功” または “証明失敗”

Simplify

$$\frac{\langle E \uplus \{s \approx t\}, H\rangle}{\langle E \cup \{s' \approx t\}, H\rangle}\ s \to_{\mathcal{R} \cup H} s'$$

Delete

$$\frac{\langle E \uplus \{s \approx s\}, H\rangle}{\langle E, H\rangle}$$

Expand

$$\frac{\langle E \uplus \{s \approx t\}, H\rangle}{\langle E \cup Expd_u(s,t), H \cup \{s \to t\}\rangle}\ u \in \mathcal{B}(s), s > t$$

図 1. 書き換え帰納法の推論規則

等式集合と項書き換え規則集合の対について，図 1 の推論規則を適用して，$\langle E, H\rangle$ から $\langle E', H'\rangle$ が得られることを $\langle E, H\rangle \rightsquigarrow \langle E', H'\rangle$ と記す． $\rightsquigarrow$ の反射推移閉包を $\rightsquigarrow^*$ と記す． 必要に応じて，$\rightsquigarrow$ に添字をつけ Simplify 規則 (Delete 規則， Expand 規則) による導出を $\rightsquigarrow^s$ ($\rightsquigarrow^d, \rightsquigarrow^e$) と表す．

直観的には，E はこれから証明しようとする等式を，H は帰納法の仮定および証明済みの定理を表す．等式集合 E が $\mathcal{R}$ の帰納的定理であることを証明するには，$\langle E, \emptyset\rangle$ から導出を始める．最終的に $\langle \emptyset, H\rangle$ が導出されるとき，証明成功となる．

書き換え帰納法は半アルゴリズムである．すなわち，等式集合が空にならないまま無限に導出を繰り返して手続きが停止しないことがあり得る．この場合を手続きが**発散する**という．また，規則を適用する等式の選び方や適用する推論規則に任意性があるので上記の手続きは非決定的である．

例 2.2 (書き換え帰納法の実行例) 項書き換えシステム $\mathcal{R}$ と証明する等式集合 E として，以下が与えられたとする．

$$\mathcal{R} = \left\{ \begin{array}{lcl} +(0,y) & \to & y \\ +(s(x),y) & \to & s(+(x,y)) \end{array} \right\}$$

$$E = \left\{ +(+(x,y),z) \approx +(x,+(y,z)) \right\}$$

$> を優位順序 + > s > 0$ に基づく辞書式経路順序としたときの書き換え帰納法の実行過程の例を図 2 に示す．

次の定理により，書き換え帰納法の手続きが “証明成功” を返した場合，入力等式集合 E が入力項書き換えシステム $\mathcal{R}$ の帰納的定理であることが保証される．

命題 2.3 (書き換え帰納法の正当性 [3]) $\mathcal{R}$ を擬簡約な項書き換えシステム， E を等式集合，$>$ を $\mathcal{R} \subseteq >$ なる簡約順序とする．ある書き換え規則集合 H が存在して，$\langle E, \emptyset\rangle \rightsquigarrow^* \langle \emptyset, H\rangle$ となるならば，$\mathcal{R} \models_{ind} E$ である．

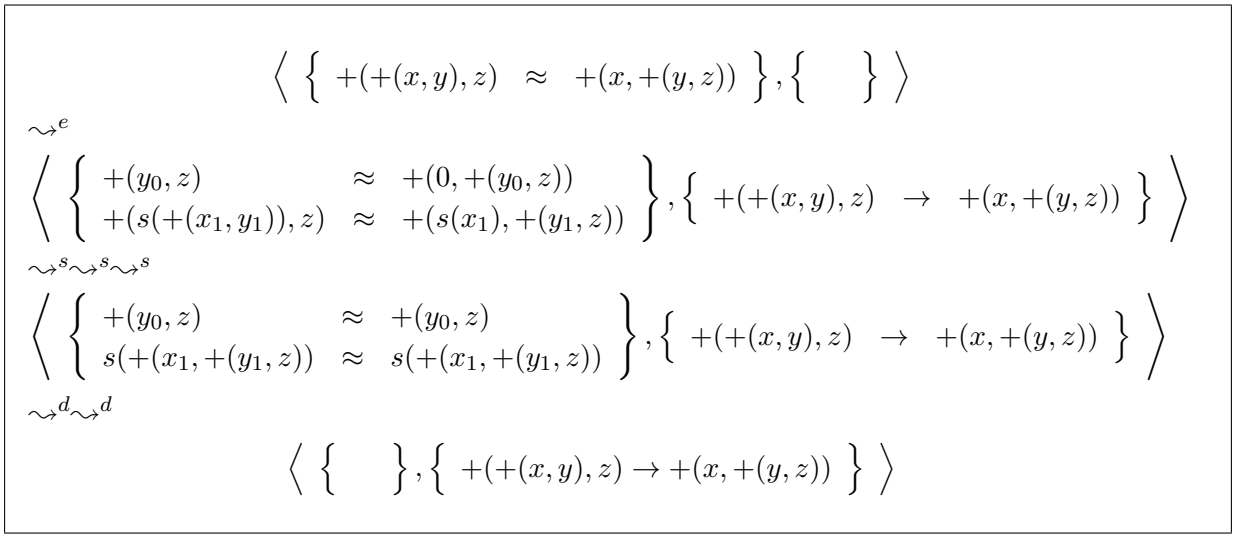

図 2. 書き換え帰納法の実行例

3 自然数乗加算の TRS に関する決定手続きを用いた帰納的定理証明手続き

Aoto と Stratulat[2] によって，特定の項書き換えシステム $\mathcal{R}$ に対する帰納的定理の決定手続きが与えられている．自然数の乗加算の項書き換えシステムに関する決定手続き $DecProc^{N\{\times,+\}}$ の入出力を表 2 に，自然数の加算の項書き換えシステムに関する決定手続き $DecProc^{N\{+\}}$ の入出力を表 3 に示す．

表 2. $DecProc^{N\{\times,+\}}$ の入出力

入力	等式集合 $E \subseteq T(\{f_\times, f_+, f_s, f_0\}, \mathcal{V})^2$
	$f_\times : N, N \to N$
	$f_+ : N, N \to N$
	$f_s : N \to N$
	$f_0 : N$
出力	True if $\mathcal{R}^N_{(f_\times,f_+,f_s,f_0)} \models_{ind} E$
	False if $\mathcal{R}^N_{(f_\times,f_+,f_s,f_0)} \not\models_{ind} E$

表 3. $DecProc^{N\{+\}}$ の入出力

入力	等式集合 $E \subseteq T(\{f_+, f_s, f_0\}, \mathcal{V})^2$
	$f_+ : N, N \to N$
	$f_s : N \to N$
	$f_0 : N$
出力	True if $\mathcal{R}^N_{(f_+,f_s,f_0)} \models_{ind} E$
	False if $\mathcal{R}^N_{(f_+,f_s,f_0)} \not\models_{ind} E$

ここで，$\mathcal{R}^N_{(f_\times,f_+,f_s,f_0)}$ は以下の項書き換えシステムを表す．また，$\mathcal{R}^N_{(f_+,f_s,f_0)}$ は以下の項書き換えシステムの最初の 2 つの書き換え規則からなる．

$$\left\{ \begin{array}{lcl} f_+(f_0,y) & \to & y \\ f_+(f_s(x),y) & \to & f_s(f_+(x,y)) \\ f_\times(f_0,z) & \to & f_0 \\ f_\times(f_s(x),y) & \to & f_+(y,f_\times(x,y)) \end{array} \right\}$$

ただし，$(f_\times, f_+, f_s, f_0)$ は入力に応じて具体的な関数記号におきかえられる．

しかし，この手続きは，関数記号をパラメータとして必要とする．そのため，書き換え帰納法と同じ枠組みで用いることができない．そこで，この節では，項書き換えシステムと等式集合と簡約順序のみの入力から，決定手続きを用いる手法 (ProcNatInd) を提案する．提案手法では，まず，決定手続きに用いる関数記号の候補を推定し，その後，その候補による決定手続きの利用が正しいかを判定する．まず，関数記号の情報を用いて関数記号の候補を絞り込む．

表 **4.** ProcNatInd の入出力

入力	擬簡約な項書き換えシステム $\mathcal{R}$, 等式集合 E, $\mathcal{R} \subseteq >$ なる簡約順序 $>$
出力	True または “証明失敗”

定義 3.1 (乗加算・加算記号候補) (f_1, f_2, f_3, f_4) が *($\alpha \in \mathcal{S}$ に対する)* **乗加算記号候補**であるとは，$f_1 : \alpha, \alpha \to \alpha$, $f_2 : \alpha, \alpha \to \alpha$, $f_3 : \alpha \to \alpha$, $f_4 : \alpha$, $f_1, f_2 \in \mathcal{D}$, $f_3, f_4 \in \mathcal{C}$ となるときをいう．同様にして，**加算記号候補** (f_2, f_3, f_4) であるとは，乗加算記号候補から f_1 に該当する関数記号を取り除いたものをいう．

これらの関数記号候補を発見する手続きについては 5 節で説明する．得られた関数記号候補で決定手続きを利用できるためには，$\mathcal{R} \models_{ind} \mathcal{R}^N_{(f_1,f_2,f_3,f_4)}$ が成立することが必要である．そこで，この判定を書き換え帰納法を用いて行う．

定義 3.2 (ProcNatInd) ProcNatInd の入出力を表 4 に，また，手続きを以下に示す．

Step 1. $\mathcal{R}$ から定義記号の集合 $\mathcal{D}$ と構成子の集合 $\mathcal{C}$ を構成する．

Step 2. $\mathcal{D}$,$\mathcal{C}$ の情報とソート情報で，$\mathcal{F}(E) \subseteq \{f_1, f_2, f_3, f_4\}$ となるような乗加算記号候補 (f_1, f_2, f_3, f_4) を見つける．候補がなければ Step 5 へ．

Step 3. $\mathcal{R} \models_{\mathsf{ind}} \mathcal{R}^N_{(f_1,f_2,f_3,f_4)}$ を入力された簡約順序 $>$ を用いて書き換え帰納法で証明．
証明成功なら Step 4 へ，失敗なら Step 2 へ戻って異なる候補を選択．

Step 4. $DecProc^{N\{\times,+\}}(E, f_1, f_2, f_3, f_4)$ が True ならば True を，それ以外は証明失敗を返す．

Step 5. $\mathcal{D}$,$\mathcal{C}$ の情報とソート情報で，$\mathcal{F}(E) \subseteq \{f_1, f_2, f_3\}$ となるような加算記号候補 (f_1, f_2, f_3) を見つける．候補がなければ失敗を返す．

Step 6. $\mathcal{R} \models_{\mathsf{ind}} \mathcal{R}^N_{(f_1,f_2,f_3)}$ を入力された簡約順序 $>$ を用いて書き換え帰納法で証明．
証明成功なら Step 7 へ，失敗ならそのまま失敗を返す．

Step 7. $DecProc^{N\{+\}}(E, f_1, f_2, f_3)$ が True ならば True を，それ以外は証明失敗を返す．

定理 3.3 (ProcNatInd の正当性) $\mathrm{ProcNatInd}(\mathcal{R}, E, >) = True$ のとき，$\mathcal{R} \models_{ind} E$ である．

(証明) 以下では $\mathcal{R}^N_{(f_1,f_2,f_3)}$ の場合は同様であるため，$\mathcal{R}^N_{(f_1,f_2,f_3,f_4)}$ の場合のみ考える．$\mathcal{R}^N_{(f_1,f_2,f_3,f_4)}$ を $\mathcal{R}^N$ と省略する．$\mathcal{R} \models_{ind} \mathcal{R}^N$ より，$\forall l \to r \in \mathcal{R}^N$, $\forall \theta_g.\ l\theta_g \overset{*}{\leftrightarrow}_{\mathcal{R}} r\theta_g$ である．$\mathcal{R}^N \models_{ind} E$ と仮定すると，$\forall u \approx v \in E$, $\forall \theta_g.\ u\theta_g \overset{*}{\leftrightarrow}_{\mathcal{R}^N} v\theta_g$ であるから，$\forall u \approx v \in E$, $\forall \theta_g.\ u\theta_g \overset{*}{\leftrightarrow}_{\mathcal{R}} v\theta_g$．よって，$\mathcal{R} \models_{ind} E$ が成立． □

例 3.4 入力として次の項書き換えシステム $\mathcal{R}$，等式集合 E が与えられたとする．

$$\mathcal{R} = \left\{ \begin{array}{lcl} plus(zero, x) & \to & x \\ plus(succ(x), z) & \to & succ(plus(x, z)) \\ times(zero, z) & \to & zero \\ times(succ(x), y) & \to & plus(y, times(x, y)) \end{array} \right\}$$

$$E = \left\{ \ times(x, plus(y,z)) \approx plus(times(x,y), times(x,z)) \ \right\}$$

関数記号を定義記号 $\mathcal{D} = \{times, plus\}$，構成子 $\mathcal{C} = \{succ, zero\}$ に分割する (Step 1)．この情報とソート情報を用いて，乗加算記号候補 $(plus, times, succ, zero)$ が選ばれたとする (Step 2)．$\mathcal{R} \models_{\mathsf{ind}} \mathcal{R}^N_{(plus,times,succ,zero)}$ を書き換え帰納法を用いて検証し，この場合は証明が失敗するため，Step 2 に戻る (Step 3)．前とは別の乗加算記号候補 $(times, plus, succ, zero)$ を選択する (Step 2)．$\mathcal{R} \models_{\mathsf{ind}} \mathcal{R}^N_{(times,plus,succ,zero)}$ を書き換え帰納法を用いて検証し，“証明成功” が返ってきて Step 4 へ移る (Step 3)．$\mathcal{R}^N_{(times,plus,succ,zero)} \models_{\mathsf{ind}} E$ を決定手続きで判定し，“True” が返ってくる (Step 4)．したがって，$\mathcal{R} \models_{ind} E$ の証明に成功する．

例 3.5 (公理の表現が異なる例) 入力 $\mathcal{R}$，E を以下のとおり与える．

$$\mathcal{R} = \left\{ \begin{array}{lcl} +(y,0) & \to & y \\ +(y,s(x)) & \to & s(+(x,y)) \end{array} \right\}$$

$$E = \left\{ \ +(+(x,y),z) \ \approx \ +(y,+(z,x)) \ \right\}$$

ここで ProcNatInd を用いると，関数記号を定義記号 $\mathcal{D} = \{+\}$，構成子 $\mathcal{C} = \{s, 0\}$ に分割する (Step 1)．次に，$\mathcal{R}$ に出現する関数記号が 3 つであることから，乗加算記号候補が見つからず，Step 5 へ移る (Step 2)．$\mathcal{D}, \mathcal{C}$ の情報とソート情報を用いて，加算記号候補 $(+, s, 0)$ が選ばれたとする (Step 5)．$\mathcal{R} \models_{\mathsf{ind}} \mathcal{R}^N_{(+,s,0)}$ を書き換え帰納法を用いて検証し，“証明成功” が返ってきて Step 7 へ移る (Step 6)．$\mathcal{R}^N_{(+,s,0)} \models_{\mathsf{ind}} E$ を決定手続きで判定し，“True” が返ってくる (Step 4)．したがって，$\mathcal{R} \models_{ind} E$ の証明に成功する．

$\mathcal{R}$ において加算が第 2 引数に関する場合分けで定義されていることに注意する．このように，公理の形が異なる場合にも，決定手続きをうまく適用することができる．

4 自然数リストの TRS に関する決定手続きを用いた帰納的定理証明手続き

文献 [2] では，自然数リストに関する項書き換えシステムに対する帰納的定理の決定手続きも与えられている．以下に，決定手続き $DecProc^{NL}$ の入出力を表 6 に示す．ここで，$\mathcal{R}^{NL}_{f_{rev},f_@,f_{cons},f_{nil},f_s,f_0}$ は以下の項書き換えシステムを表わす．また，$\mathcal{R}^{NL}_{f_@,f_{cons},f_{nil},f_s,f_0}$ は，以下の項書き換えシステムの最初の 2 つの書き換え規則からなる．以下では，$F_@ = \{f_@, f_{cons}, f_{nil}, f_s, f_0\}$ とおき，列 $(f_@, f_{cons}, f_{nil}, f_s, f_0)$ を $\vec{F}_@$ と記す．また，$F_{rev} = \{f_{rev}\} \cup F_@$ とおき，列 $(f_{rev}, f_@, f_{cons}, f_{nil}, f_s, f_0)$ を $\vec{F}_{rev}$ と記す．

表 5. $DecProc^{NL\{@,rev\}}$ の入出力

入力	等式集合 $E \subseteq T(F_@, \mathcal{V})^2$
	$f_{rev} : NL \to NL$
	$f_@ : NL, NL \to NL$
	$f_{cons} : N, NL \to NL$
	$f_{nil} : NL$
	$f_s : N \to N$
	$f_0 : N$
出力	True　if $\mathcal{R}^{NL}_{\vec{F}_{rev}} \models_{ind} E$
	False　if $\mathcal{R}^{NL}_{\vec{F}_{rev}} \not\models_{ind} E$

表 6. $DecProc^{NL\{@\}}$ の入出力

入力	等式集合 $E \subseteq T(F_{rev}, \mathcal{V})^2$
	$f_@ : NL, NL \to NL$
	$f_{cons} : N, NL \to NL$
	$f_{nil} : NL$
	$f_s : N \to N$
	$f_0 : N$
出力	True　if $\mathcal{R}^{NL}_{\vec{F}_@} \models_{ind} E$
	False　if $\mathcal{R}^{NL}_{\vec{F}_@} \not\models_{ind} E$

$$\mathcal{R}^{NL} = \left\{ \begin{array}{lcl} f_@(f_{nil}, ys) & \to & ys \\ f_@(f_{cons}(x, xs), ys) & \to & f_{cons}(x, f_@(xs, ys)) \\ f_{rev}(f_{nil}) & \to & f_{nil} \\ f_{rev}(f_{cons}(x, xs)) & \to & f_@(f_{rev}(xs), f_{cons}(x, f_{nil})) \end{array} \right\}$$

ただし，$f_{rev}, f_@, f_{cons}, f_{nil}$ は入力に応じて具体的な関数記号におきかえられる．これらの決定手続きに対しても，3 節に与えたときと同様に，項書き換えシステムと等式集合と簡約順序のみの入力から，関数記号を推定して，(可能な場合に) 決定手続きを用いる帰納的定理証明手続きを与えることは容易である．決定手続き $DecProc^{NL}$ を用いる手続きを ProcNatListInd とおく．手続きの詳細は略す．また，定義 3.1 と同様に，リスト演算に対する関数記号候補を以下のように定義する．

定義 4.1 (リスト演算に対する関数記号候補) $(f_1, \ldots, f_6)$ が**連結反転リスト候補**であるとは，ソート α, β $(\alpha \neq \beta)$ が存在し，$f_1 : \alpha \to \alpha$，$f_2 : \alpha, \alpha \to \alpha$，$f_3 : \beta, \alpha \to \alpha$，$f_4 : \alpha$，$f_5 : \beta \to \beta$，$f_6 : \beta$，$f_1, f_2 \in \mathcal{D}$，$f_2, \ldots, f_6 \in \mathcal{C}$ であるときを言う．$(f_2, \ldots, f_6)$ が**連結リスト候補**であるとは，連結反転リスト候補から f_1 に該当する関数記号を取り除いたものをいう．

5　ソートおよび関数記号の推定を利用した帰納的定理証明手続き

3,4 節では，決定手続きを利用した帰納的定理証明手続き ProcNatInd, ProcNatListInd を与えた．しかし，一般的には，入力の $\mathcal{R}$ と E が与えられたときに，これらのどの手続きを利用すればよいかは前もってはわかっているわけではない．本節では，ProcNatInd, ProcNatListInd に共通に用いることのできるソートおよび関数記号の推定手続きを与える．これにより，推定されたソートおよび関数記号に応じて，適切な帰納的定理証明手続きをよび出すことが可能になる．

以下では，関数記号 f のソート情報を $f : \alpha_1, \ldots, \alpha_n \to \alpha_0$ とするとき，$Sort_i(f)$ $(0 \leq i \leq n)$ を $Sort_i(f) = \alpha_i$ と定義する．

定義 5.1 (ソート情報の推定 (estimateSort))

Nat に対応するソート型を推定する手続き estimateNat，*NatList* に対応するソート型を推定する手続き estimateList を順に実行する．

Nat の推定 (estimateNat)

入力：項書き換えシステム $\mathcal{R}$

出力：ソート集合 $NAT \subseteq \mathcal{S}$

Step 1. 定義記号 $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$，構成子 $\mathcal{C} = \mathcal{F} \backslash \mathcal{D}$，$NAT := \{\ \}$，
$maybeSucc = \{f \in \mathcal{C} \mid arity(f) = 1,\ Sort_0(f) = Sort_1(f)\}$，
$maybeZero = \{f \in \mathcal{C} \mid arity(f) = 0\}$ とおく．
$maybeSucc \neq \emptyset$ かつ，$maybeZero \neq \emptyset$ ならば Step2 へ，そうでなければ終了．

Step 2.
$$NotNAT := \left\{ \alpha \in \mathcal{S} \;\middle|\; \begin{array}{l} \exists f, g \in maybeSucc \\ f \neq g, Sort_0(f) = Sort_0(g) = \alpha \end{array} \right\}$$
$maybeSucc := \{f \in maybeSucc \mid Sort_0(f) \notin NotNAT\}$

Step 3. $maybeSucc$ の要素 $c \in maybeSucc$ それぞれについて，$Sort_0(c') = Sort_0(c)$ となる $c' \in maybeZero$ なる c' が唯一つ存在するとき，$NAT := NAT \cup \{(Sort_0(c), c, c')\}$

List の推定 (estimateList)
入力：項書き換えシステム $\mathcal{R}$
出力：$NAT\_LIST \subseteq \mathcal{S}$

Step 1. estimateNat を実行しておく（出力：NAT）

Step 2. 定義記号 $\mathcal{D} = \{l(\epsilon) \mid l \to r \in \mathcal{R}\}$，構成子 $\mathcal{C} = \mathcal{F} \backslash \mathcal{D}$，$NAT\_LIST = \{\ \}$，
$maybeCons = \{f \in \mathcal{C} \mid arity(f) = 2,\ Sort_0(f) = Sort_2(f),\ Sort_0(f) \neq Sort_1(f)\}$，
$maybeNil = \{f \in \mathcal{C} \mid arity(f) = 0\}$ とおく．
$maybeCons \neq \emptyset$ かつ，$maybeNil \neq \emptyset$ ならば Step2 へ，それ以外は終了．

Step 3.

$$NotNATLIST := \left\{ \alpha \in \mathcal{S} \;\middle|\; \begin{array}{l} \exists f, g \in maybeCons \\ f \neq g, Sort_0(f) = Sort_0(g) = \alpha \end{array} \right\}$$

$maybeCons := \{f \in maybeCons \mid Sort_0(f) \notin NotNATLIST\}$

Step 4. $maybeCons$ の要素 $c \in maybeCons$ それぞれについて，$Sort_0(c') = Sort_0(c)$ となる $c' \in maybeNil$ なる c' が唯一つ存在するとき，Step5. へ

Step 5. $s = Sort_1(c)$ なる $(s, d, d') \in NAT$ が存在するとき，$NAT\_LIST := NAT\_LIST \cup \{Sort_0(c), Sort_1(c), c, c', d, d')\}$

例 5.2 (estimateSort の実行例) $S = \{N, NL\}$ とし，関数記号集合のソート情報を，以下のとおり与える．

$$\left\{ \begin{array}{lcl} times & : & N, N \to N \\ plus & : & N, N \to N \\ succ & : & N \to N \\ zero & : & N \end{array} \right\} \cup \left\{ \begin{array}{lcl} @ & : & NL, NL \to NL \\ cons & : & N, NL \to NL \\ rev & : & NL \to NL \\ nil & : & NL \end{array} \right\}$$

また，入力 $\mathcal{R}$ は，例 3.4 の $\mathcal{R}$ に以下の規則を追加した項書き換えシステムとし，等式集合 E を以下のとおりに与える．

$$\left\{ \begin{array}{lcl} @(nil, ys) & \to & ys \\ @(cons(x, xs), ys) & \to & cons(x, @(xs, ys)) \\ rev(nil) & \to & nil \\ rev(cons(x, xs)) & \to & @(rev(xs), cons(x, nil)) \end{array} \right\}$$

$$E = \left\{ \begin{array}{l} cons(times(zero, x), zs) \approx \\ cons(times(times(plus(z, z), zero), succ(times(zero, zero))), zs) \end{array} \right\}$$

estimateNat の実行例
関数記号集合を定義記号集合 $\mathcal{D} = \{times, plus, len, rev, @\}$，構成子集合 $\mathcal{C} = \{succ, zero, cons, nil\}$ に分割し，$maybeSucc = \{succ\}$，$maybeZero = \{zero, nil\}$ ととる (Step1)．$maybeSucc$ の要素がただ 1 つであるため，次のステップへ (Step2)．$c = succ$ について，(i)$c' = zero$ のとき，$Sort_0(c') = N = Sort_0(c)$，(ii)$c' = nil$ のとき，$Sort_0(c') = NL \neq N = Sort_0(c) = N$．(i),(ii) より，$NAT = \{(N, succ, zero)\}$ (Step3).

estimateList の実行例
estimateNat を実行し，$NAT = \{(N, succ, zero)\}$ を得る (Step1)．関数記号集合を定義記号集合 $\mathcal{D} = \{times, plus, len, rev, @\}$，構成子集合 $\mathcal{C} = \{succ, zero, cons, nil\}$ に分割し，$maybeCons = \{cons\}$，$maybeNil = \{zero, nil\}$ ととる (Step2)．$maybeCons$ の要素がただ 1 つであるため，次のステップへ (Step3)．$c = cons$ について，(i)$c' = zero$ のとき，$Sort_0(c') = N \neq NL = Sort_0(c)$，(ii)$c' = nil$

のとき，$Sort_0(c') = NL = Sort_0(c)$．(i),(ii) より，次のステップへ (Step4)．$Sort_1(c) = N$ であるから，$NAT\_LIST = \{(NL, N, cons, nil, succ, zero)\}$(Step5)．

定義 5.3 (決定手続きと関数記号の推定を利用した帰納的定理証明手続き (ProcInd))

Step 1. estimateSort を実行し，ソート情報を推定する．(出力：NAT, NAT_LIST)

Step 2. それぞれの $(\alpha, c, c') \in NAT$ に対して，ソート α に対する乗加算記号候補 (f_1, f_2, c, c') または加算記号候補 (f_2, c, c') をすべて構成し，これらの候補の集合を $NatFunc$ とおく．同様にして，NAT_LIST の要素それぞれに対する連結反転リスト候補，または連結リスト候補をすべて構成し，これらの候補の集合を $NatListFunc$ とおく．

Step 3. $(f_1, f_2, f_3, f_4) \in NatFunc$ が存在し，$\mathcal{F}(E) \subseteq \{f_1, f_2, f_3, f_4\}$ となるとき，候補 (f_1, f_2, f_3, f_4) を用いて，ProcNatInd の Step2 以下を実行する．同様にして，$\mathcal{F}(E)$ を含む連結反転リスト候補，または連結リスト候補があるとき，ProcNatListInd を実行する．

6 決定手続きを利用した書き換え帰納法

書き換え帰納法を用いて証明をする際，等式をそれ以上 Expand できずに停止してしまう場合 (証明失敗) や，等式集合が空にならないまま無限に導出を繰り返して手続きが停止しないことがあり得る (発散)．しかし，決定手続きは，等式集合が自然数やリストの公理を満たす場合に限定されるものの，発散することなく証明が終了する．そこで，書き換え帰納法と決定手続きの手法を組み合わせることで，より強力な証明システムを実現する．

例 6.1 入力として以下の項書き換えシステム $\mathcal{R}$ と等式集合 E を与える．(pow は自然数を 2 乗する関数である．)

$$E = \left\{ \begin{array}{lcl} pow(x) & \approx & T(x, x) \end{array} \right\}$$

$$\mathcal{R} = \left\{ \begin{array}{lcl} T(Z, y) & \to & Z \\ T(S(x), y) & \to & P(y, T(x, y)) \\ P(Z, y) & \to & y \\ P(S(x), y) & \to & S(P(x, y)) \\ pow(Z) & \to & Z \\ pow(S(x)) & \to & P(pow(x), S(T(x, S(S(Z))))) \end{array} \right\}$$

この場合，導出過程は図 3 のとおりになり，証明に失敗する．しかしながら，最後の等式集合に着目してみると，$\mathcal{R}^N_{(T,P,S,Z)}$ の帰納的定理である．よって，前節の特定の関数記号に依存しない決定手続きを用いることで，証明に成功する．

そこで，次のように書き換え帰納法を拡張する．

定義 6.2 (書き換え帰納法の拡張) 書き換え帰納法に以下の規則を追加する．

E-Delete

$$\frac{\langle E \uplus \{s \approx s''\}, H \rangle}{\langle E, H \rangle} \ \mathrm{ProcInd}(\mathcal{R}, \{s \approx s''\}, >) = \mathrm{True}$$

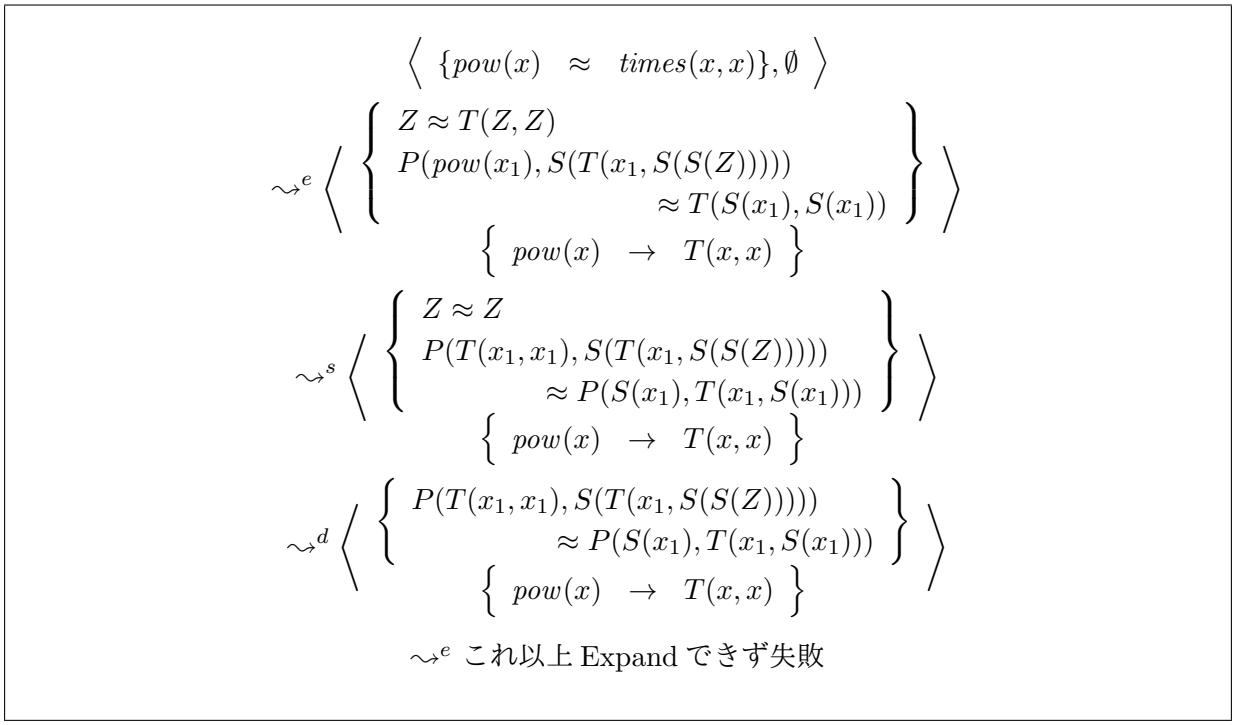

図 3. pow 関数を用いた等式の書き換え帰納法での導出例

表 7. 融合証明法の入出力

入力	擬簡約な項書き換えシステム $\mathcal{R}$, 等式集合 E, $\mathcal{R} \subseteq >$ なる簡約順序 $>$
出力	“証明成功” または “証明失敗”

この拡張した書き換え帰納法を融合証明法とよび，その入出力を表 7 に示す．

この規則は，導出過程の等式集合 E に含まれる等式 $s \approx s''$ について，特定の関数記号に依存しない決定手続き (ProcInd) を用いて帰納的定理が証明成功する場合，該当する等式 $s \approx s''$ を等式集合 E から取り除く．

先の pow 関数を用いた例 6.1 が E-Delete 規則を適用した拡張書き換え帰納法を用いることで証明に成功する．

7 実装と実験

書き換え帰納法 (RI)，本論文で提案した融合証明法 (FP) の 2 つの証明法を実装した．実装には関数型言語である SML/NJ を用いた．プログラムコードは約 3000 行である．書き換え帰納法の入力に必要な簡約順序は辞書式経路順序を用いた．また，擬簡約性はユーザーが保証するものとして，判定していない．書き換え帰納法 (RI) や融合証明法 (FP) 内の拡張書き換え帰納法において，一連の (Expand,Simplify,Delete,E-delete) 規則の適用を 1 回としたとき，20 回行っても手続きが停止しない場合に，手続きが発散した (∞) とした．

17 個の例で実験を行った結果を表 8 に示す．書き換え帰納法 (RI) では，28 個中 4 つに成功した．失敗例は，E が空でないにも拘らず，どの推論規則も適用できずに停止した．融合証明法 (FP) に

ついては，今回実験した 17 個中 14 個に成功した．失敗例は，導出の途中で等式集合 E 中の等式が，決定手続きを適用できなかった場合である．これらの実験結果から，決定手続きや書き換え帰納法で証明困難な例についても，提案手法が有効であることが確認された．

表 8. 提案手法を用いた帰納的定理証明の実験結果 (○:定理，×:失敗，∞:発散)

	等式集合 E	RI	FP
1	$succ(plus(x, plus(y, z))) \approx plus(y, plus(succ(z), x))$	×	○
2	$plus(plus(y, plus(z, x)), z) \approx plus(y, x)$	○	○
3	$succ(succ(plus(plus(succ(x), times(x, x)), x)))$ $\approx plus(succ(x), succ(succ(plus(x, times(x, x)))))$	∞	○
4	$times(x, plus(zero, zero)) \approx times(plus(x, x), zero)$	×	○
5	$pow(x) \approx T(x, x)$	×	○
6	$r(a(r(xs), r(ys))) \approx a(ys, xs)$	∞	○
7	$r(a(xs, xs)) \approx a(r(xs), r(xs))$	∞	○
8	$a(r(r(xs)), n) \approx xs$	∞	○
9	$r(a(xs, ys)) \approx a(r(ys), r(xs))$	×	○
10	$r(a(r(xs), ys)) \approx a(r(ys), xs)$	×	○
11	$qrev(xs, ys) \approx a(r(xs), ys)$	×	○
12	$revapp(xs, revapp(ys, zs)) \approx revapp(a(ys, xs), zs)$	×	○
13	$qrev(xs, ys) \approx A(R(xs), ys), \quad rev2(xs) \approx R(xs)$	×	○
14	$ite(plus(x, y), xs) \approx app(ite(x, xs), ite(y, xs))$, $app(app(xs, ys), zs) \approx app(xs, app(ys, zs))$	○	○
15	$last(rev(cons(x, ys))) \approx cons(x, nil)$	∞	∞
16	$mult(app(xs, ys)) \approx times(mult(xs), mult(ys))$	×	×
17	$foldrH2(xs, y) \approx foldlH2(rev(xs), y)$	×	×
成功数		2/17	14/17

8 おわりに

文献 [2] の自然数の乗加算やリスト構造の演算を用いた決定手続き ($DecProc$) の手法と書き換え帰納法を融合することで，帰納的定理証明の能力を高める手法を提案した．文献 [2] では，乗加算とリスト演算の両方が同時に含まれた等式やリストの長さを返す演算が含まれた項書き換えシステムに対する決定手続きも与えられている．提案手法にこれらの決定手続きを組み込むことは容易であると思われるが，関数記号が多く含まれる場合に対応できるように，より効率的な関数記号推定の手法を用いる必要があると考えられる．今回の実験例では書き換え帰納法が失敗する原因の多くは，それ以上 Expand 規則を適用できないためであった．用いる簡約順序をより強力にしたり，あるいは等式の向き付けが必要ない書き換え帰納法の拡張を利用することで，Expand 規則が適用できるようになることもある．そのような書き換え帰納法においても発散する例は多いため，本手法はやはり有効であると考えられるが，そのようなより強力な書き換え帰納法に決定手続きを融合して本手法の有効性を確認することも今後の課題の一つである．また，書き換え帰納法中の関数記号推定の実装について，現在の実装では，一度推定した関数記号であっても再度公理を証明して関数記号を推定し直しているため，一度推定した情報を保存しておくなど改良することで，より効率化を図る必要がある．また，文献 [2] の決定手続きにおいて，リストを自然数リストに限定する必要

があるのは，反証に関する健全性を保証するためである．提案手法では反証を利用しないので，要素を自然数としないリストに対しても本手法は拡張できると考えられるが，そのような拡張も今後の課題である．最後に，文献 [2] 以外の決定手続きを調査して応用することや，決定手続きを利用して健全一般化法 [1, 6] といった補題生成法をより強力なものに拡張すること，項分割法 [4] と決定手続きを融合したより強力な帰納的定理証明法の考案なども検討していきたい．

謝辞

本論文を改善するための大変貴重なコメントをいただ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 18K11158 の補助を受けて行われた．

参考文献

[1] Aoto, T.: Sound lemma generation for proving inductive validity of equations, *Proc. of 28th FSTTCS*, LIPIcs, Vol. 2, Schloss Dagstuhl, 2008, pp. 13–24.

[2] Aoto, T. and Stratulat, S.: Decision procedures for proving inductive theorems without induction, *Proc. of the 16th PPDP*, ACM Press, 2014, pp. 237–248.

[3] Reddy, U. S.: Term rewriting induction, *Proc. of the 10th CADE*, LNAI, Vol. 449, Springer-Verlag, 1990, pp. 162–177.

[4] Urso, P. and Kounalis, E.: Term partition for mathematical induction, *Proc. of 14th RTA*, LNCS, Vol. 2706, Springer-Verlag, 2003, pp. 352–366.

[5] 小池広高, 外山芳人: 潜在帰納法と書き換え帰納法の比較, コンピュータソフトウェア, Vol. 17, No. 6(2000), pp. 1–12.

[6] 嶌津聡志, 青戸等人, 外山芳人: 反証機能付き書き換え帰納法のための補題自動生成法, コンピュータソフトウェア, Vol. 26, No. 2(2009), pp. 41–55.

A 実験の詳細

以下に，実験例として用いた書き換え規則，辞書式経路順序を構成するのに用いた優位順序 $>$ を記載する．（それぞれの関数は，適宜置き換えられているものとする．）

- 実験例 1,2

$$\mathcal{R} = \left\{ \begin{array}{lcl} plus(z, y) & \to & y \\ plus(succ(x), y) & \to & succ(plus(x, y)) \end{array} \right\}$$

$$plus > succ > z$$

- 実験例 3,4

$$\mathcal{R} = \left\{ \begin{array}{lcl} plus(zero, y) & \to & y \\ plus(succ(x), y) & \to & succ(plus(x, y)) \\ times(zero, y) & \to & zero \\ times(succ(x), y) & \to & plus(y, times(x, y)) \end{array} \right\}$$

$$times > plus > succ > zero$$

- 実験例 5

$$\mathcal{R} = \left\{ \begin{array}{lcl} P(Z,y) & \to & y \\ P(S(x),y) & \to & S(P(x,y)) \\ T(Z,y) & \to & Z \\ T(S(x),y) & \to & P(y,T(x,y)) \\ pow(Z) & \to & Z \\ pow(S(x)) & \to & P(pow(x),S(T(x,S(S(Z))))) \end{array} \right\}$$

$$pow > T > P > S > Z$$

- 実験例 6〜10

$$\mathcal{R} = \left\{ \begin{array}{lcl} a(n,ys) & \to & ys \\ a(c(x,xs),ys) & \to & c(x,a(xs,ys)) \\ r(n) & \to & n \\ r(c(x,xs)) & \to & a(r(xs),c(x,n)) \end{array} \right\}$$

$$r > a > c > n > s > 0$$

- 実験例 11

$$\mathcal{R} = \left\{ \begin{array}{lcl} a(n,ys) & \to & ys \\ a(c(x,xs),ys) & \to & c(x,a(xs,ys)) \\ r(n) & \to & n \\ r(c(x,xs)) & \to & a(r(xs),c(x,n)) \\ qrev(n,ys) & \to & ys \\ qrev(c(x,xs),ys) & \to & qrev(xs,c(x,ys)) \end{array} \right\}$$

$$qrev > r > a > c > n > s > 0$$

- 実験例 12

$$\mathcal{R} = \left\{ \begin{array}{lcl} a(n,ys) & \to & ys \\ a(c(x,xs),ys) & \to & c(x,a(xs,ys)) \\ r(n) & \to & n \\ r(c(x,xs)) & \to & a(r(xs),c(x,n)) \\ revapp(xs,ys) & \to & a(r(xs),ys) \end{array} \right\}$$

$$revapp > r > a > c > n > s > 0$$

- 実験例 13

$$\mathcal{R} = \left\{ \begin{array}{lcl} A(N,ys) & \to & ys \\ A(C(x,xs),ys) & \to & C(x,A(xs,ys)) \\ R(N) & \to & N \\ R(C(x,xs)) & \to & A(R(xs),C(x,N)) \\ qrev(N,ys) & \to & ys \\ qrev(C(x,xs),ys) & \to & qrev(xs,C(x,ys)) \\ rev2(xs) & \to & qrev(xs,N) \end{array} \right\}$$

$$qrev > rev2 > R > A > C > N > s > 0$$

- 実験例 14

$$\mathcal{R} = \left\{ \begin{array}{lcl} app(nil,ys) & \to & ys \\ app(cons(x,xs),ys) & \to & cons(x,app(xs,ys)) \\ plus(zero,y) & \to & y \\ plus(succ(x),y) & \to & succ(plus(x,y)) \\ ite(zero,xs) & \to & nil \\ ite(succ(x),xs) & \to & app(xs,ite(x,xs)) \end{array} \right\}$$

$$ite > app > cons > nil > plus > succ > zero$$

- 実験例 15

$$\mathcal{R} = \left\{ \begin{array}{lcl} app(nil, ys) & \to & ys \\ app(cons(x, xs), ys) & \to & cons(x, app(xs, ys)) \\ last(nil) & \to & nil \\ last(cons(x, nil)) & \to & cons(x, nil) \\ last(cons(x, cons(y, zs))) & \to & last(cons(y, zs)) \\ rev(nil) & \to & nil \\ rev(cons(x, xs)) & \to & app(rev(xs), cons(x, nil)) \end{array} \right\}$$

$$rev > last > app > cons > nil > s > 0$$

- 実験例 16

$$\mathcal{R} = \left\{ \begin{array}{lcl} app(cons(x, xs), ys) & \to & cons(x, app(xs, ys)) \\ app(nil, ys) & \to & ys \\ plus(0, y) & \to & y \\ plus(s(x), y) & \to & s(plus(x, y)) \\ times(0, y) & \to & 0 \\ times(s(x), y) & \to & plus(y, times(x, y)) \\ mult(nil) & \to & s(0) \\ mult(cons(x, ys)) & \to & times(x, mult(ys)) \end{array} \right\}$$

$$mult > times > plus > app > cons > nil > s > 0$$

- 実験例 17

$$\mathcal{R} = \left\{ \begin{array}{lcl} app(nil, ys) & \to & ys \\ app(cons(x, xs), ys) & \to & cons(x, app(xs, ys)) \\ plus(0, y) & \to & y \\ plus(s(x), y) & \to & s(plus(x, y)) \\ h2(x, y) & \to & plus(x, y) \\ foldlH2(nil, y) & \to & y \\ foldlH2(cons(x, xs), y) & \to & foldlH2(xs, h2(y, x)) \\ foldrH2(nil, y) & \to & y \\ foldrH2(cons(x, xs), y) & \to & h2(x, foldrH2(xs, y)) \\ rev(nil) & \to & nil \\ rev(cons(x, ys)) & \to & app(rev(ys), cons(x, nil)) \end{array} \right\}$$

$$rev > app > foldlH2 > foldrH2 > h2 > plus > cons > nil > s > 0$$