書き換え帰納法を利用した帰納的定理証明の補題生成法

加藤 裕人, 青戸 等人

新潟大学大学院自然科学研究科

{kato@nue., aoto@}ie.niitaga-u.ac.jp

概 要　項書き換えシステムの帰納的定理の自動証明法においては，適当な補題の発見が証明成功の鍵となることが多い．そのため，発散鑑定法や健全一般化法といった補題生成法が提案されている．しかし，これらは証明過程で局所的に適切な補題を生成する手法であり，証明過程に依存した限定的な補題しか生成されない．本論文では，特定の証明過程に役立つ補題だけではなく，他の様々な自動証明手法において役立つような補題を発見するための補題生成法を提案する．提案手法のアイデアは，項書き換えシステムの自動証明法の一つである書き換え帰納法を柔軟に実行し，様々な証明発散過程を得ることである．また、その証明発散過程から差分照合を利用することで補題を生成する．これにより，発散鑑定法で発見できないような発散過程に対しても補題生成が可能となる．提案手法の実装および実験について報告する．

1　はじめに

等式論理に基づく計算モデルとして項書き換えシステムがある．項書き換えシステムをプログラムとしてみなすときに成立する等式を帰納的定理とよぶ．帰納的定理の自動証明法として，Reddy により提案された書き換え帰納法が知られている [8]．書き換え帰納法では，等式の集合と書き換え規則の集合の対に関する導出を行いながら，証明を行う．証明過程において，推論規則による導出が無限に繰り返される場合には手続きは停止せず (発散)，証明に失敗する．このような場合に，適切な補題を一緒に証明することで，発散が抑制されて証明が成功することがある．また，書き換え帰納法の拡張として反証機能付き書き換え帰納法が知られている [3][5]．反証機能付き書き換え帰納法では，与えられた等式集合に対して証明と反証を並行して実行することができる．

書き換え帰納法が発散して失敗する場合に，その発散系列から補題を発見する方法として発散鑑定法がある [10]．発散鑑定法は明示的帰納法のヒューリスティックであるリップリング法 [4] を書き換え帰納法の補題生成に応用したものである．しかし，どのような場合でも必ず補題が発見できるわけではない．また，書き換え帰納法や発散鑑定法の手続きは，規則を適用する等式の選び方や，適用する推論規則等に任意性があるため，非決定的である．このため，適切な補題を導出する方法は自明ではない．一方，補題の発見は書き換え帰納法に限らず，様々な自動証明法において，証明成功の鍵となることが多い．このため，補題発見の成功率を高めることができれば，より強力な定理自動証明システムの実現につながると考えられる．

定理自動証明システムに有用な補題を発見するシステムとして HipSpec が提案されている [6]．HipSpec は，Haskell プログラムに関して様々な性質を証明する Hip[9] とよばれる自動ツールと，関数プログラムについての等式を推論する QuickSpec[7] とよばれるツールを組み合わせることで補題を生成する．HipSpec では，より単純な等式から始まってランダムに等式を生成し，その等式の証明に成功するとき，それを補題として出力する．

補題を自動生成する場合，完全にランダムに補題を生成すると，補題の探索に多くの時間がかかってしまう．逆に探索時間を少なくしようとすると，補題の候補を絞り込みすぎて補題の生成に失敗してしまう．そこで，有効な補題の候補をうまく絞り込める発散鑑定法を基に探索時間を増やすことで有効な補題を発見できると考えられる．

本論文では，書き換え帰納法を利用した補題生成法を提案する．書き換え帰納法において，発散系列を抽出するもととなる導出の仮定部の集合が大きく生成されるように，本来，定理証明を目的としている書き換え帰納法のために必要となっていた戦略や制約を緩和する．さらに，生成された仮定部から発散系列を抽出する手法，発散鑑定法で補題を生成できない発散系列からの補題を生成する手法を提案する．また，提案手法の実装および実験結果について報告する．

本論文の構成は次のとおりである．第 2 節では本論文で用いる基本概念や記法について説明する．第 3 節では書き換え帰納法を用いた発散系列の生成法について説明する．第 4 節では発散鑑定法で発見できないような発散過程に対する補題生成手法について説明する．第 5 節では提案手法の実装と実験結果を示す．第 6 節は結論である．

2 準備

本節では，本論文で用いる定義および記法について説明する．

2.1 項書き換えシステム

2 項関係 $\rightarrow$ の反射推移閉包を $\stackrel{*}{\rightarrow}$，等価閉包を $\stackrel{*}{\leftrightarrow}$ と記す．ソート付き関数記号集合およびソート付き変数集合記号をそれぞれ $\mathcal{F},\mathcal{V}$ で表す．関数記号 f の引数の個数を $arity(f)$ で表す．$\mathcal{F}_0 = \{f \in \mathcal{F} \mid arity(f) = 0\}$ とおく．$\mathcal{F},\mathcal{V}$ 上のソート付き項の集合を $\mathcal{T}(\mathcal{F},\mathcal{V})$ で表す．項上の等式を $s \approx t$ と記す．ただし，ここで，s,t は同一のソートを持つ項とする．また，等式 $s \approx t$ と $t \approx s$ は同一視する．項 t の根記号 $root(t)$ とは $t \in \mathcal{V}$ のときは t，$t = f(t_1,\ldots,t_n)(f \in \mathcal{F})$ のときは，f である．t を項とするとき，$\mathcal{F}(t)$ は t に出現する関数全体の集合を，$\mathcal{V}(t)$ は t に出現する変数全体の集合を表す．項 t が $\mathcal{V}(t) = \emptyset$ を満たすとき，t を基底項とよぶ．文脈とは特別な定数 $\Box$(ホール) をちょうど 1 つ持つ項のことである．$C[t]$ は文脈 C のホールを項 t で置き換えて得られる項を表す．項 u が t の部分項であるとは，$t = C[u]$ なる文脈 C があるときをいう．

関数 $\sigma : \mathcal{V} \rightarrow \mathcal{T}(\mathcal{F},\mathcal{V})$ で，x と $\sigma(x)$ が同一のソートをもち，かつ，集合 $\{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ が有限であるものを，代入とよぶ．集合 $\{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ を代入 σ の定義域とよび，$dom(\sigma)$ と記す．代入 σ の定義域を $\mathcal{T}(\mathcal{F},\mathcal{V})$ に拡張した準同型拡張も同じく σ で表し，項 t の代入 $\sigma(t)$ を $t\sigma$ と記す．$t\sigma$ が基底項となるとき，これを t の基底具体化とよぶ．等式 $s \approx t$ の基底具体化も同様に定める．

同一ソートを持つ 2 つの項 l,r が，$l \notin \mathcal{V}, \mathcal{V}(r) \subseteq \mathcal{V}(l)$ を満たすとき，$l \rightarrow r$ 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 (TRS) とよぶ．項書き換えシステムに含まれる書き換え規則の左辺の根記号となる関数記号を定義記号という．定義記号集合を $\mathcal{D}$ と記す．構成子記号集合を $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ と定義する．定義記号を含まない項を構成子項とよぶ．構成子項の集合を $\mathcal{T}(\mathcal{C},\mathcal{V})$ と記す．定義記号 f と構成子項 $c_1,\ldots,c_n$ からなる項 $f(c_1,\ldots,c_n)$ を基本項とよぶ．基本項の集合を $\mathcal{B}(\mathcal{D},\mathcal{C},\mathcal{V})$ と記す．t を項とするとき，t の基本部分項の集合を $\mathcal{B}(t)$ と記す．任意の $l \rightarrow r \in \mathcal{R}$ について，$l \in \mathcal{B}(\mathcal{D},\mathcal{C},\mathcal{V})$ であるとき，$\mathcal{R}$ を構成子システムとよぶ．

$\mathcal{R}$ を項書き換えシステムとする．書き換え規則 $l \rightarrow r \in \mathcal{R}$，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，$s \rightarrow_{\mathcal{R}} t$ と記す．代入および文脈に閉じている項上の関係 $>$ を書き換え関係とよぶ．整礎な書き換え関係を簡約関係という．特に，簡約関係が半順序 (非反射的かつ推移的) となるとき，簡約順序とよぶ．どの基本基底項も正規形でないとき，擬簡約であるという．任意の基底項 t に対してある基底構成子項 s が存在して $t \stackrel{*}{\rightarrow}_{\mathcal{R}} s$ となるとき，$\mathcal{R}$ は十分完全性をもつという．

2.2 帰納的定理と書き換え帰納法

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，$s\theta_g \stackrel{*}{\leftrightarrow}_{\mathcal{R}} t\theta_g$ が成立するときをいう．また，等式集合 E のすべての等式が帰納的定理で

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

Decompose

$$\frac{\langle E \uplus \{f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)\}, H\rangle}{\langle E \cup \{s_1 \approx t_1, \ldots, s_n \approx t_n\}, H\rangle} f \in \mathcal{C}$$

Disproof

$$\frac{\langle E \uplus \{s \approx x\}, H\rangle}{\text{Disproof}} x \in \mathcal{V}\backslash\mathcal{V}(s)$$

$$\frac{\langle E \uplus \{f(s_1,\ldots,s_n) \approx x\}, H\rangle}{\text{Disproof}} f \in \mathcal{C}, x \in \mathcal{V}$$

$$\frac{\langle E \uplus \{f(s_1,\ldots,s_m) \approx g(t_1,\ldots,t_n)\}, H\rangle}{\text{Disproof}} f \neq g,\ f,g \in \mathcal{C}$$

図 2. Decompose 規則と Disproof 規則

あるとき，E は $\mathcal{R}$ の帰納的定理であるという．

例 2.1 (帰納的定理) 自然数上の加算を項書き換えシステム $\mathcal{R}$ で与える．ここで，自然数 $0,1,2,\ldots$ は $\mathsf{0}, \mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{s}(\mathsf{0})), \ldots$ と表現する．

$$\mathcal{R} = \begin{cases} +(\mathsf{0}, x) \to x \\ +(\mathsf{s}(x), y) \to \mathsf{s}(+(x,y)) \end{cases}$$

このとき，$+(+(x,y),z) \approx +(x,+(y,z))$ は，任意の基底項 s_g, t_g, u_g について $+(+(s_g,t_g),u_g) \overset{*}{\leftrightarrow}_{\mathcal{R}} +(s_g,+(t_g,u_g))$ が成立する．よって，$\mathcal{R}$ の帰納的定理である．

帰納的定理の自動証明法として書き換え帰納法が知られている [8]．書き換え帰納法では，等式集合 E(等式部とよぶ) と書き換え規則集合 H(仮定部とよぶ) の対 $\langle E, H\rangle$ に関する導出を行いながら証明を行う．図 1 に書き換え帰納法の推論規則を示す．ここで，$\mathcal{R}$ は項書き換えシステム，$\uplus$ は直和，$>$ は簡約順序，$\mathsf{mgu}(s,t)$ は s と t の最汎単一化子を表す．$Expd$ は以下のように定義される演算である．

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

なお，適宜，$(\mathcal{V}(s) \cup \mathcal{V}(t)) \cap \mathcal{V}(l) = \emptyset$ となるように，書き換え規則の変数は名前変えする．$\langle E, H\rangle$ から推論規則を適用して $\langle E', H'\rangle$ 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 $\overset{*}{\rightsquigarrow}$ 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．書き換え帰納法の原理は簡約順序 $>$ によるネーター帰納法に基づく．*Expand* 規則により，より順序の小さい等式へ問題を帰着させている．

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，$\mathcal{R} \subseteq >$ なる簡約順序 $>$ を与え，$\langle E, \emptyset \rangle$ から導出を始め，推論規則の適用を繰り返す．最終的に $\langle \emptyset, H \rangle$ が導出されるとき，証明成功となる．また，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る．この場合を手続きが発散するという．

例 2.2 (書き換え帰納法の証明) 例 2.1 の $\mathcal{R}$ を与え，次の等式集合 E に書き換え帰納法を適用して E が $\mathcal{R}$ の帰納的定理であることを示す．

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

以下で，$>$ を $+ > \mathsf{s} > \mathsf{0}$ 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証明過程を示す．

$$\left\langle \left\{ +(+(x,y),z) \approx +(x,+(y,z)) \right\}, \{\} \right\rangle$$

$$\rightsquigarrow^{e} \quad \left\langle \left\{ \begin{array}{l} +(x,z) \approx +(x,+(\mathsf{0},z)) \\ +(\mathsf{s}(+(x,y)),z) \approx +(\mathsf{s}(x),+(y,z)) \end{array} \right\}, \left\{ +(+(x,y),z) \to +(x,+(y,z)) \right\} \right\rangle$$

$$\rightsquigarrow^{s}\rightsquigarrow^{s}\rightsquigarrow^{s}\rightsquigarrow^{s} \quad \left\langle \left\{ \begin{array}{l} +(x,z) \approx +(x,z) \\ \mathsf{s}(+(x,+(y,z))) \approx \mathsf{s}(+(x,+(y,z))) \end{array} \right\}, \left\{ +(+(x,y),z) \to +(x,+(y,z)) \right\} \right\rangle$$

$$\rightsquigarrow^{d}\rightsquigarrow^{d} \quad \left\langle \{\}, \left\{ +(+(x,y),z) \to +(x,+(y,z)) \right\} \right\rangle$$

$\langle \emptyset, H \rangle$ が導出されたため，証明成功となる．

例 2.3 (書き換え帰納法の発散) リスト結合，リスト反転を項書き換えシステム $\mathcal{R}$ で与える ($x :: [\,]$ を $[x]$ で表す)．

$$\mathcal{R} = \left\{ \begin{array}{l} @([\,], ys) \to ys \\ @(x :: xs, ys) \to x :: @(xs, ys) \\ \mathsf{rev}([\,]) \to [\,] \\ \mathsf{rev}(x :: xs) \to @(\mathsf{rev}(xs), [x]) \end{array} \right.$$

$$E = \left\{ \mathsf{rev}(\mathsf{rev}(@(xs, ys))) \approx @(xs, ys) \right.$$

書き換え帰納法の実行過程を以下に示す．

$$\left\langle \left\{ \mathsf{rev}(\mathsf{rev}(@(xs,ys))) \approx @(xs,ys) \right\}, \{\} \right\rangle$$

$$\overset{*}{\rightsquigarrow} \left\langle \left\{ \begin{array}{l} \mathsf{rev}(@(\mathsf{rev}(@(xs,ys)),[x])) \approx x :: @(xs,ys) \\ \mathsf{rev}(\mathsf{rev}(@(xs,ys))) \to @(xs,ys) \end{array} \right\}, \right\rangle$$

$$\overset{*}{\rightsquigarrow} \left\langle \left\{ \begin{array}{l} \mathsf{rev}(@(@(\mathsf{rev}(@(xs,ys)),[x]),[y])) \approx y :: (x :: @(xs,ys)) \\ \mathsf{rev}(@(\mathsf{rev}(@(xs,ys)),[x])) \to x :: @(xs,ys) \\ \mathsf{rev}(\mathsf{rev}(@(xs,ys))) \to @(xs,ys) \end{array} \right\}, \right\rangle$$

$$\overset{*}{\rightsquigarrow} \cdots$$

このように書き換え帰納法の手続きを繰り返し適用しても等式部が空にならず，発散する．発散した場合，適当な等式を E に追加して書き換え帰納法を実行することで発散を抑制し，証明に成功する場合がある．このように，E の証明を成功させるために，E に追加して一緒に証明を行う等式を補題とよぶ．

また，図 1 の推論規則に図 2 の推論規則を追加することで帰納的定理でないことを証明することができる．反証を行うことができる書き換え帰納法は反証機能付き書き換え帰納法とよばれている [3][5].

命題 2.4 (反証機能付き書き換え帰納法の正当性 [3][5][11]) どのソートについても 2 つ以上の基底構成子項が存在するものとする．また，$\mathcal{R}$ が基底合流性をもつ擬簡約な構成子システムとし，$>$ は $\mathcal{R} \subseteq >$ なる簡約順序とする．このとき，(1)$\langle E, \emptyset\rangle \rightsquigarrow^* \langle \emptyset, H\rangle$ ならば E は $\mathcal{R}$ の帰納的定理である．(2)$\langle E, \emptyset\rangle \rightsquigarrow^*$ Disproof ならば E は $\mathcal{R}$ の帰納的定理でない．

2.3 発散鑑定法による補題生成

書き換え帰納法が発散する場合，同じパターンにより拡大する系列 (発散系列) が観測されることがある．発散鑑定法 [10] は，書き換え帰納法が発散した場合に，その発散系列を収束させるような補題を構成する．以下で，発散鑑定法による補題生成を簡単な例で説明する．

例 2.5 (発散鑑定法による補題生成例) 例 2.3 の $\mathcal{R}$ と E について，発散鑑定法で補題を生成することを考える．例 2.3 に書き換え帰納法を適用した場合，以下のような発散系列が出現する．

$$\begin{aligned}
\mathsf{rev}(\mathsf{rev}(@(xs, ys))) &\to @(xs, ys) && (1)\\
\mathsf{rev}(@(\mathsf{rev}(@(xs, ys)), [x])) &\to x :: @(xs, ys) && (2)\\
\mathsf{rev}(@(@(\mathsf{rev}(@(xs, ys)), [x]), [y])) &\to y :: (x :: @(xs, ys)) && (3)\\
&\vdots
\end{aligned}$$

(1) と (2) の書き換え規則の差分は以下のように，枠と下線 (注釈とよぶ) を用いて表せる．

$$\mathsf{rev}(\boxed{@(\underline{\mathsf{rev}(@(xs, ys))}, [x])}) \to \boxed{x :: \underline{@(xs, ys)}} \qquad (4)$$

ここで，注釈付きの書き換え規則 (4) の枠から下線部を取り除いた部分が書き換え規則 (1) と (2) の差分となっており，差分を取り除くと変数の違いを除いて書き換え規則 (1) が得られることに注意する．このような差分を求める手続きとして差分照合が知られている [2]．差分照合によって複数の差分が得られることがある．そのような場合には，発散鑑定法の補題生成に失敗する差分の数を減らすために，可能な限り外側に差分があるものを選ぶ (極大差分照合)．

この差分 (4) から補題を次のように生成する．差分 (4) の右辺の下線部 $@(xs, ys)$ は書き換え規則 (1) の右辺 $@(xs, ys)$ と照合する．よって，書き換え規則 (1) を右辺から左辺へ逆向きに使って，書き換え規則 (2) の右辺を書き換えると次の書き換え規則を得る．

$$\mathsf{rev}(\boxed{@(\underline{\mathsf{rev}(@(xs, ys))}, [x])}) \to \boxed{x :: \underline{\mathsf{rev}(\mathsf{rev}(@(xs, ys)))}} \qquad (5)$$

最後に両辺に共通する部分項 $\mathsf{rev}(@(xs, ys))$ を新しい変数 zs を用いて，(5) を一般化すると，次の補題が生成される．

$$\mathsf{rev}(@(zs, [x])) \approx x :: (\mathsf{rev}(zs))$$

これを等式として，等式集合 E に追加して，書き換え帰納法を実行すると証明に成功する．

以上を手続きとしてまとめる．発散系列を $\{s_i \to t_i\}_i$，文脈を G, H, C，極大差分照合を $s_i = G[U_i], s_{i+1} = G[\boxed{H[\underline{U_i}]}], t_i = V_i, t_{i+1} = \boxed{C[\underline{V_i}]}$，とする．このとき，書き換え規則 $G[\boxed{H[\underline{U_i}]}] \to \boxed{C[\underline{G[U_i]}]}$ を生成する．その後，両辺の共通な部分項出現 $\underline{U_i}$ を新しい変数で一般化することで補題を得る．

発散鑑定法による補題生成は健全ではない．つまり、以下のように正しくない補題 (非定理) を生成することがある．

例 2.6 (発散鑑定法による非健全な補題生成例) 反復的なリスト反転を項書き換えシステム $\mathcal{R}$ で与え，等式集合 E に書き換え帰納法を適用する．

$$\begin{aligned}\mathcal{R} &= \left\{ \begin{array}{l} \mathsf{qrev}([\,], ys) \to ys \\ \mathsf{qrev}(x :: xs, ys) \to \mathsf{qrev}(xs, x :: ys) \end{array} \right. \\ E &= \left\{ \mathsf{qrev}(\mathsf{qrev}(xs, [\,]), [\,]) \approx xs \right.\end{aligned}$$

このとき，書き換え帰納法は発散し，以下のような発散系列が出現する．

$$\begin{aligned} \mathsf{qrev}(\mathsf{qrev}(xs, [\,]), [\,]) &\to xs && (6)\\ \mathsf{qrev}(\mathsf{qrev}(xs, x :: [\,]), [\,]) &\to x :: xs && (7)\\ \mathsf{qrev}(\mathsf{qrev}(xs, x :: (y :: [\,])), [\,]) &\to y :: (x :: xs) && (8)\\ &\vdots \end{aligned}$$

(6) と (7) の書き換え規則の差分照合は

$$\mathsf{qrev}(\mathsf{qrev}(xs, \boxed{x :: \underline{[\,]}}), [\,]) \to \boxed{x :: \underline{xs}} \tag{9}$$

となる．差分照合 (9) の右辺の下線部 xs は書き換え規則 (6) の右辺 xs と照合する．よって，書き換え規則 (6) を右辺から左辺へ逆向きに使って，書き換え規則 (7) の右辺を書き換えると次の書き換え規則を得る．

$$\mathsf{qrev}(\mathsf{qrev}(xs, \boxed{x :: \underline{[\,]}}), [\,]) \to \boxed{x :: \underline{\mathsf{qrev}(\mathsf{qrev}(xs, [\,]), [\,])}} \tag{10}$$

最後に両辺に共通する部分項 $[\,]$ を新しい変数 ys を用いて，(10) を一般化すると，次の補題が生成される．

$$\mathsf{qrev}(\mathsf{qrev}(xs, x :: ys), [\,]) \approx x :: \mathsf{qrev}(\mathsf{qrev}(xs, ys), [\,]) \tag{11}$$

しかし，この補題は非定理である．このように，発散鑑定法は正しくない補題を生成する可能性がある．

3 書き換え帰納法を用いた発散系列の生成法

3.1 補題生成のための書き換え帰納法

従来提案されている健全発散鑑定法付き書き換え帰納法 [11] では，書き換え帰納法の証明，発散系列の観測，補題の生成と追加を逐次的に繰り返す．このため，その証明過程に特有な補題しか生成されない．

一方、補題を発見するためには多くの発散系列の観測が有用であり，様々な発散パターンが含まれる仮定部を得るのが望ましい．そこで，この目的に合うように *Expand* 規則を変更する．

書き換え帰納法の *Expand* 規則には以下の制約がある．

- 等式部に含まれる 1 つの等式 $s \approx t$ に対して実行する．
- 簡約順序 $s > t$ が成立する等式 $s \approx t$ に対してだけ実行する．
- 一つの基本部分項 $u \in \mathcal{B}(s)$ に対して実行する．

G-Expand

$$\frac{\langle E \uplus \{s_i \approx t_i\}_i, H\rangle}{\langle E \cup \bigcup_i \bigcup_{u\in\mathcal{B}(s_i)} Expd_u(s_i,t_i), H \cup \{s_i \to t_i\}_i\rangle}$$

Decompose

$$\frac{\langle E \uplus \{f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)\}, H\rangle}{\langle E \cup \{s_1 \approx t_1,\ldots,s_n \approx t_n\}, H\rangle} f \in \mathcal{C}$$

図 3. 補題生成のための書き換え帰納法

これらの制約を緩和することによって様々な発散パターンが含まれる仮定部を生成する．変更した *Expand* 規則を図 3 に示す．

G-Expand 規則では簡約順序 $s > t$ が成立するしないにかかわらず，等式集合に含まれる複数の等式 $s \approx t$ に対して，複数の基本部分項 u に対して並列に $Expd_u(s,t)$ による展開を行う．

さらに，反証機能付き書き換え帰納法 [3][5] の推論規則の 1 つである *Decompose* 規則を組み込む (図 3)．*Decompose* 規則は必ずしも健全であるとは限らないが，*Decompose* 規則を書き換え帰納法の中に組み込むことで，補題生成に適した、より単純な等式を含む仮定部を生成できる．

また，*G-Expand* 規則において順序付けできない等式も扱うことができるので，*Simplify*, *Delete* 規則についても順序付けできない書き換え規則を扱えるように，書き換え帰納法に基づいた基底合流性証明 [1] で提案されている規則を用いる．

3.2 発散系列の抽出

補題生成のための書き換え帰納法を用いて，多様な仮定部を生成した場合，様々な複合的な発散系列を観測することがある．そのような場合，一般的な補題を見つけるためにはその発散系列から単純な系列を発見することが望ましい．そこで以下では発散系列抽出手続きを与える．まず手続きに用いる階層項の定義を与える．

定義 3.1 階層項を以下のように帰納的に定義する．

1. 基本項は階層項
2. $f \in \mathcal{D}$ かつ $t_1,\ldots,t_n$ が階層項のとき，$f(t_1,\ldots,t_n)$ は階層項

階層項の集合を $\mathcal{H}(\mathcal{D},\mathcal{C},\mathcal{V})$ と記す．一般に，書き換え帰納法では内側にある構成子を外側に出せないため発散してしまう．そのため階層項を考えることで，内側にある構成子を外側に出せる補題を生成するのに必要な発散系列の発見につながる．

Step1 仮定部から $r_0 \in \mathcal{H}(\mathcal{D},\mathcal{C},\mathcal{V})$ となる書き換え規則 $l_0 \to r_0$ を選ぶ．

Step2 $l_0 \to r_0$ との差分が存在する書き換え規則 $l_1 \to r_1$ を仮定部から選び，その差分照合を $G[\boxed{H[\underline{l_0}]}] \to D[\boxed{C[\underline{r_0}]}]$ とする．

Step3 残りの仮定部から以下のように差分が増えていく書き換え規則を発散系列として抽出．

$$G[H[H[l_0]]] \to D[C[C[r_0]]]$$
$$G[H[H[H[l_0]]]] \to D[C[C[C[r_0]]]]$$
$$\vdots$$

なお，**Step1** で書き換え規則を選ぶ際は，$r_0 \in \mathcal{B}(\mathcal{D}, \mathcal{C}, \mathcal{V})$ を満たすものから優先する．**Step1** で右辺が階層項である書き換え規則を選ぶことで，**Step3** において内側にある構成子を外側に出せる補題を生成するのに必要な発散系列が発見できる．また，**Step2** で選ばれた書き換え規則は **Step1** では考えない．この手続きは，**Step1** の条件を満たす書き換え規則がなくなるまで繰り返す．

例 3.2 (発散系列の抽出例) 自然数上の減算を項書き換えシステム $\mathcal{R}$ で与え，等式集合 E を書き換え帰納法で証明すると，発散して失敗する．

$$\mathcal{R} = \begin{cases} \mathsf{minus}(\mathsf{0}, y) \to \mathsf{0} \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{0}) \to \mathsf{s}(x) \\ \mathsf{minus}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{minus}(x, y) \end{cases}$$

$$E = \left\{ \mathsf{minus}(\mathsf{minus}(x, y), z) \approx \mathsf{minus}(\mathsf{minus}(x, z), y) \right.$$

このとき，補題生成のための書き換え帰納法を適用すると，以下の仮定部が観測される．

$$\begin{aligned} \mathsf{minus}(\mathsf{minus}(x, y), \mathsf{0}) &\to \mathsf{minus}(x, y) \\ \mathsf{minus}(\mathsf{minus}(x, y), z) &\to \mathsf{minus}(\mathsf{minus}(x, z), y) \\ \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{0}) &\to \mathsf{minus}(\mathsf{s}(x), y) \\ \mathsf{minus}(\mathsf{minus}(x, y), \mathsf{s}(\mathsf{0})) &\to \mathsf{minus}(x, \mathsf{s}(y)) \\ \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{0})) &\to \mathsf{minus}(x, y) \\ &\vdots \end{aligned}$$

発散系列抽出手続きに基づいて発散系列を構成する．

Step1 $l_0 \to r_0$ として $\mathsf{minus}(\mathsf{minus}(x, y), \mathsf{0}) \to \mathsf{minus}(x, y)$ を選ぶ．

Step2 $l_1 \to r_1$ として $\mathsf{minus}(\mathsf{minus}(x, y), \mathsf{s}(\mathsf{0})) \to \mathsf{minus}(x, \mathsf{s}(y))$ を選び，
その差分照合を $\mathsf{minus}(\mathsf{minus}(x, y), \boxed{\mathsf{s}(\underline{\mathsf{0}})}) \to \mathsf{minus}(x, \boxed{\mathsf{s}(\underline{y})})$ とする．

Step3 以下の発散系列が仮定部から抽出される．

$$\begin{aligned} \mathsf{minus}(\mathsf{minus}(x, y), \mathsf{0}) &\to \mathsf{minus}(x, y) \\ \mathsf{minus}(\mathsf{minus}(x, y), \mathsf{s}(\mathsf{0})) &\to \mathsf{minus}(x, \mathsf{s}(y)) \\ \mathsf{minus}(\mathsf{minus}(x, y), \mathsf{s}(\mathsf{s}(\mathsf{0}))) &\to \mathsf{minus}(x, \mathsf{s}(\mathsf{s}(y))) \\ &\vdots \end{aligned}$$

なお **Step2** において，$l_1 \to r_1$ として $\mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{0})) \to \mathsf{minus}(x, y)$ を選ぶと，**Step3** において発散系列が見つからない．

次に，$l_0 \to r_0$ として $\mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{0})) \to \mathsf{minus}(x, y)$ を選ぶと，以下の発散系列が得られる．

$$\begin{aligned} \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{0})) &\to \mathsf{minus}(x, y) \\ \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{s}(\mathsf{0}))) &\to \mathsf{minus}(x, \mathsf{s}(y)) \\ \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x), y), \mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0})))) &\to \mathsf{minus}(x, \mathsf{s}(\mathsf{s}(y))) \\ &\vdots \end{aligned}$$

次に，$l_0 \to r_0$ として $\mathsf{minus}(\mathsf{minus}(x,y),z) \to \mathsf{minus}(\mathsf{minus}(x,z),y)$ を選ぶと，以下の発散系列が得られる．

$$\begin{array}{rcl}
\mathsf{minus}(\mathsf{minus}(x,y),z) & \to & \mathsf{minus}(\mathsf{minus}(x,z),y) \\
\mathsf{minus}(\mathsf{minus}(x,y),\mathsf{s}(z)) & \to & \mathsf{minus}(\mathsf{minus}(x,z),\mathsf{s}(y)) \\
\mathsf{minus}(\mathsf{minus}(x,y),\mathsf{s}(\mathsf{s}(z))) & \to & \mathsf{minus}(\mathsf{minus}(x,z),\mathsf{s}(\mathsf{s}(y))) \\
& \vdots &
\end{array}$$

最後に，$l_0 \to r_0$ として $\mathsf{minus}(\mathsf{minus}(x,y),y) \to \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x),y),\mathsf{s}(y))$ を選ぶと，以下の発散系列が得られる．

$$\begin{array}{rcl}
\mathsf{minus}(\mathsf{minus}(x,y),y) & \to & \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x),y),\mathsf{s}(y)) \\
\mathsf{minus}(\mathsf{minus}(x,y),\mathsf{s}(y)) & \to & \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x),y),\mathsf{s}(\mathsf{s}(y))) \\
\mathsf{minus}(\mathsf{minus}(x,y),\mathsf{s}(\mathsf{s}(y))) & \to & \mathsf{minus}(\mathsf{minus}(\mathsf{s}(x),y),\mathsf{s}(\mathsf{s}(\mathsf{s}(y)))) \\
& \vdots &
\end{array}$$

この発散系列を抽出すると，$l_0 \to r_0$ の候補がなくなるため手続きが終了する．

4 補題生成手法

発散鑑定法 [10] は，書き換え帰納法が発散した場合に，その発散系列を収束させるような書き換え規則を構成するのに有効である．しかしながら，発散鑑定法はどのような発散系列に対しても，必ず補題が生成できるわけではない．以下で，発散鑑定法での補題生成の失敗例を示す．

例 4.1 (発散鑑定法の補題生成失敗例) 以下の発散系列に対して，発散鑑定法を適用することを考える．

$$\begin{array}{rcll}
\mathsf{qrev}(@(xs,[x]),[\,]) & \to & x :: \mathsf{qrev}(xs,[\,]) & (12) \\
\mathsf{qrev}(@(xs,[x]),y :: [\,]) & \to & x :: \mathsf{qrev}(xs,y :: [\,]) & (13) \\
\mathsf{qrev}(@(xs,[x]),y :: (z :: [\,])) & \to & x :: \mathsf{qrev}(xs,y :: (z :: [\,])) & (14) \\
& \vdots & &
\end{array}$$

(12) と (13) の書き換え規則の差分照合は以下のように表せる．

$$\mathsf{qrev}(@(xs,[x]),\boxed{y :: \underline{[\,]}}) \to x :: \mathsf{qrev}(xs,\boxed{y :: \underline{[\,]}}) \tag{15}$$

この時，(15) の右辺は発散鑑定法の手続きにおける差分照合の右辺の形と異なるため，補題の生成に失敗する．

発散鑑定法で成功するのは，C, C', D を文脈，$s, t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ としたとき，

$$D[\boxed{C[\underline{s}]}] \to \boxed{C'[\underline{t}]}$$

のような差分が生成された場合のみである．しかし，一般的に $D[\boxed{C[\underline{s}]}] \to D'[\boxed{C'[\underline{t}]}]$, $D' \neq \Box$ のときに発散鑑定法は失敗する．以下では，このような差分が得られた場合でも補題生成に成功する手法を提案する．

4.1 差分照合と一般化に基づく補題生成法

本論文では，まず 1 つ目の補題生成手法として，差分照合と一般化に基づく補題生成法を新たに提案する．この補題生成法では発散系列から得られた差分照合を利用して，発散している部分項を一般化することで補題を生成する．具体的には，次のような場合に，発散系列から補題を生成する．

発散系列から得られる差分を $D[\boxed{C[\underline{s}]}] \to D'[\boxed{C[\underline{s}]}]$ とする．$C[s] \in \mathcal{T}(\mathcal{C},\mathcal{V}), s \in \mathcal{F}_0$ のとき，新しい変数 x を用いて一般化することで，次の補題を生成する．

$$D[x] \approx D'[x]$$

この手法は，書き換え規則の両辺に共通して帰納的に発散していく構成子項を一般化することで補題を生成している．

例 4.2 (差分照合と一般化に基づく補題生成法の補題生成例) 例 4.1 の発散系列を考えると，差分は以下のように表せる．

$$\mathsf{qrev}(@(xs,[x]),\boxed{y :: \underline{[\,]}}) \to x :: \mathsf{qrev}(xs,\boxed{y :: \underline{[\,]}})$$

ここで $y :: [\,] \in \mathcal{T}(\mathcal{C},\mathcal{V}), [\,] \in \mathcal{F}_0$ なので，差分照合と一般化に基づく補題生成法が適用でき，新しい変数 ys を用いて，以下のような補題が生成される．

$$\mathsf{qrev}(@(xs,[x]),ys) \approx x :: \mathsf{qrev}(xs,ys)$$

4.2 右辺のランダム生成に基づく補題生成法

次に 2 つ目の補題生成手法として，右辺のランダム生成に基づく補題生成法を新たに提案する．この補題生成法では，左辺を発散系列から得られた差分照合を利用することで生成し，右辺をある深さまでくまなく生成することで補題を生成する．以下では，補題の左辺の生成法と補題の右辺の生成法について，それぞれ説明する．

左辺の生成: C, C', D, D' を文脈，$s, t \in \mathcal{T}(\mathcal{F},\mathcal{V})$，差分照合を $D[\boxed{C[\underline{s}]}] \to D'[\boxed{C'[\underline{t}]}]$，$x$ を新しい変数とする．このとき，以下のように左辺を生成する．

- $root(D[\Box]) \in \mathcal{D}$，$\mathcal{F}(D[\Box]) \cap \mathcal{C} \neq \emptyset$ の場合，左辺は $D[x]$
- $root(D[C[\Box]]) \in \mathcal{D}$，$\mathcal{F}(D[C[\Box]]) \cap \mathcal{C} \neq \emptyset$ の場合，左辺は $D[C[x]]$
- それ以外の場合，失敗

補題の左辺は定義記号を根とし，少なくとも一つの構成子記号を含むように生成する．これは構成子を外側に出すような補題を生成したいという意図に基づいている．また，補題は一般的な形の方が有効に適用できる場合が多い．そこでこのように部分構成子項を一般化することで，より一般的で発散を抑制することができるような補題の左辺を生成できる．なお，左辺の生成法の 1 つ目と 2 つ目の両方が適用できるときは，1 つ目の方がより一般的な形をしているため，1 つ目の場合を優先して適用する．

右辺の生成: l を補題の左辺とするとき，深さ n 以下の補題の右辺集合 T_n を以下のように帰納的に生成する．

- $T_1 = \{t \mid t \in \mathcal{V}(l) \vee t \in \mathcal{F}_0\}$
- $T_n = \{f(t_1,\ldots,t_m) \mid f \in \mathcal{F}(l),\ t_1,\ldots,t_m \in T_{n-1}\} \cup T_{n-1}$

このように左辺と右辺を生成することによって，補題を生成する．しかし，正しくない補題が多数生成されるため，発散鑑定法等に比べて正しい補題を得るまで計算時間がかかる場合が多い．また，右辺の生成における T_n はある n まで生成する．

例 4.3 (右辺のランダム生成に基づく 補題生成法の補題生成例) 次の発散系列について，補題を 生成することを 考える．

$$\begin{aligned}
@(@(\mathsf{rev}(xs), x :: [\,]), [y]) &\to @(\mathsf{rev}(xs), x :: (y :: [\,])) && (16)\\
@(@(\mathsf{rev}(xs), x :: (y :: [\,])), [z]) &\to @(\mathsf{rev}(xs), x :: (y :: (z :: [\,]))) && (17)\\
@(@(\mathsf{rev}(xs), x :: (y :: (z :: [\,]))), [v]) &\to @(\mathsf{rev}(xs), x :: (y :: (z :: (v :: [\,])))) && (18)\\
&\vdots
\end{aligned}$$

(16) と (17) の書き換え規則から 差分照合は以下のように表せる．

$$@(@(\mathsf{rev}(xs), x :: \boxed{(y :: \underline{[\,]})}), [z]) \to @(\mathsf{rev}(xs), x :: (y :: \boxed{(z :: \underline{[\,]})}))$$

ここで $root(@(@(\mathsf{rev}(xs), x :: \Box), [z])) \in \mathcal{D}$，$::\in \mathcal{F}(@(@(\mathsf{rev}(xs), x :: \Box), [z]))$，$::\in \mathcal{C}$ より，新しい変数 ys を用いて次のような左辺を持つ補題が生成される．

$$@(@(\mathsf{rev}(xs), x :: ys), [z])$$

次に T_4 を用いて補題生成を試みると，以下のような補題が生成される．

$$\begin{aligned}
@(@(\mathsf{rev}(xs), x :: ys), [z]) &\approx xs\\
@(@(\mathsf{rev}(xs), x :: ys), [z]) &\approx @(xs, ys)\\
@(@(\mathsf{rev}(xs), x :: ys), [z]) &\approx @(\mathsf{rev}(xs), x :: (z :: ys))\\
&\vdots
\end{aligned}$$

生成した補題の中で帰納的定理であるものは次の 2 つである．

$$\begin{aligned}
@(@(\mathsf{rev}(xs), x :: ys), [z]) &\approx @(@(\mathsf{rev}(xs), [\,]), x :: @(ys, [z]))\\
@(@(\mathsf{rev}(xs), x :: ys), [z]) &\approx @(\mathsf{rev}(xs), x :: @(ys, [z]))
\end{aligned}$$

5 実装と実験

反証機能付き書き換え帰納法，発散鑑定法，本論文で提案した補題生成法を実装し，帰納的定理自動証明システムを構築した．

5.1 実装

実装には関数型言語 SML/NJ を用いた．実装した自動証明システムは，構成子システム $\mathcal{R}$ および等式集合 E を与え，E が $\mathcal{R}$ の帰納的定理であるかを判定する．入力は MSTRS 形式[1]のファイルに (CONJECTURES *equations*) を追加したファイルで指定する．なお，構成子システム $\mathcal{R}$ の十分完全性についてはユーザーが保証するものとして，判定していない．

自動証明システムの手続きについて説明する．帰納的定理自動証明システムの手続きを図 4 に示す．簡約順序には (半順序 $>$ に基づく) 辞書式経路順序を用い，構成子システム $\mathcal{R}$ から SMT ソルバを用いることで自動で構成する．まず，反証機能付き書き換え帰納法で証明を行う．本論文の自動証明システムの反証機能付き書き換え帰納法は，*Expand*, *Simplify*, *Delete*, *Decompose*, *Disproof*

[1] http://project-coco.uibk.ac.at/problems/mstrs.php

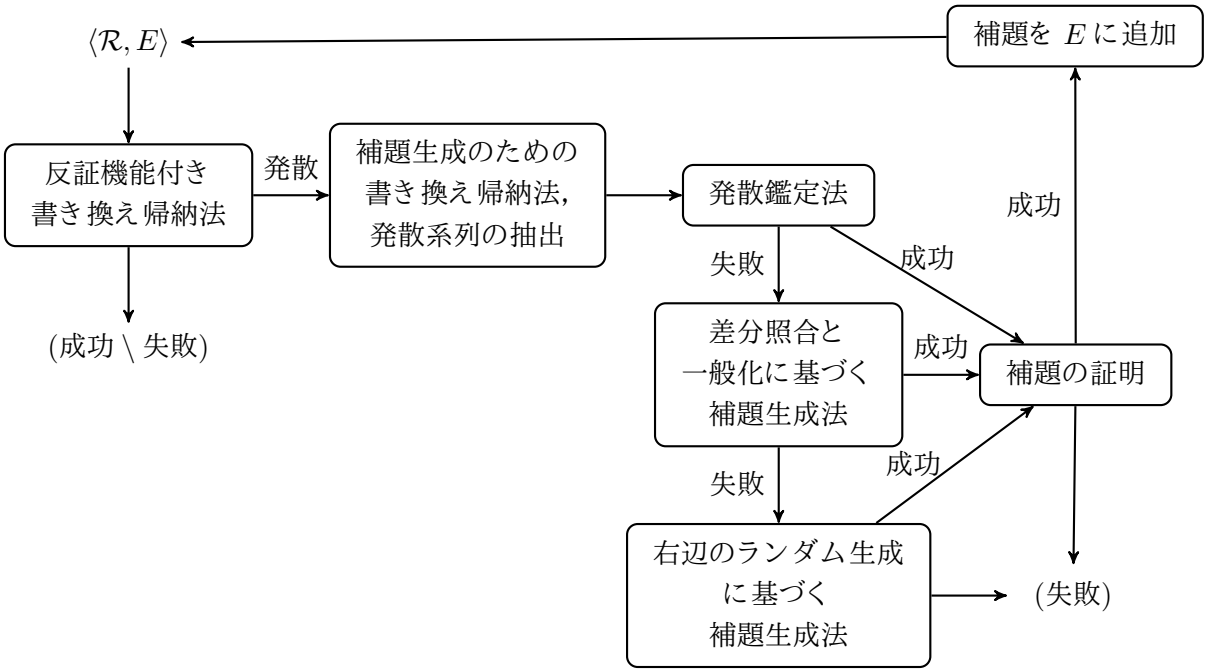

図 4. 帰納的定理自動証明システムの手続き

規則を 1 サイクルとして手続きを繰り返す．この手続きを 6 回繰り返しても終わらない場合に，発散しているとみなす．なお，反証機能付き書き換え帰納法における *Expand*, *Simplify*, *Delete* 規則については，順序付けできない等式を扱えるように書き換え帰納法に基づいた基底合流性証明 [1] で提案されている規則に拡張している．反証機能付き書き換え帰納法で帰納的定理の証明 (もしくは反証) に成功した場合，成功を返して終了する．なお，反証の健全性には，基底合流性，基底構成子項の存在条件が必要である [11] がユーザーが保証するものとして判定していない．

反証機能付き書き換え帰納法が発散した場合，提案した補題生成のための書き換え帰納法を用いて発散系列を抽出する．補題生成のための書き換え帰納法では *G-Expand, Simplify, Decompose, Delete* 規則を 1 サイクルとして，6 回繰り返したところで仮定部から発散系列の抽出を行う．なお，*G-Expand* 規則の $\{s_i \approx t_i\}_i$ の選択は，等式集合 E に含まれるすべての等式について行う．次に抽出したすべての発散系列に対して，補題の生成を行う．

まず，抽出した発散系列に対して発散鑑定法での補題生成を試みる．発散鑑定法で補題が生成できた場合，その補題が帰納的定理であれば，等式集合 E に補題を追加して，その等式集合 E を反証機能付き書き換え帰納法で証明を行う．発散鑑定法での補題の生成に失敗，または生成した補題が帰納的定理であることが証明できなかった場合は，提案した補題生成手法での補題の発見を試みる．

まず，差分照合と一般化に基づく補題生成法を実行する．補題が生成できた場合，その補題が帰納的定理であれば，等式集合 E に補題を追加して，その等式集合 E を反証機能付き書き換え帰納法で証明する．補題の生成に失敗，または生成した補題が帰納的定理であることが証明できなかった場合は，右辺のランダム生成に基づく補題生成法を実行する．右辺のランダム生成に基づく補題生成法では，補題の左辺の深さ以下で右辺を生成する．

以上のように補題の生成は，発散鑑定法，差分照合と一般化に基づく補題生成法，右辺のランダム生成に基づく補題生成法の順に補題の生成を行う．これは，右辺のランダム生成における計算量が多いためである．なお，生成された補題の証明については，反証機能付き書き換え帰納法，または提案した帰納的定理自動証明システムでの証明のどちらを用いるかをオプションで指定する．また，補題のみを出力するオプションもある．

実装したシステムは `http://www.nue.ie.niigata-u.ac.jp/tools/lgrip` からダウンロードで

表 1. 補題自動生成法の実験結果

	補題生成数	証明結果 (成功/失敗/発散)	時間 (min:sec)
実装ツール	129	52/25/12	12:10
(補題生成法なし)	0	3/82/ 4	4:23
(発散鑑定法)	55	28/55/ 6	6:46
(差分照合と一般化)	13	3/81/ 5	5:12
(右辺のランダム生成)	110	31/42/16	18:35
(右辺のランダム生成, 深さ 3)	116	32/40/17	18:27
(右辺のランダム生成, 深さ 4)	29	17/35/37	38:56
(右辺のランダム生成, 深さ 5)	13	10/29/50	51:19
(noIteration)	69	39/36/14	14:08
MSG+SG[11]	-	36/13/40	–
HipSpec[6]	377	73/10/ 6	12:28

きる．

5.2 実験

本実験では，健全発散鑑定法· 健全一般化法付き書き換え帰納法 (MSG+SG)[11]，HipSpec[6]，提案手法の証明能力の比較を行った．文献 [2][11] の例を用いて，全部で 89 例に対して，実験を行った．タイムアウトを 60 秒に設定し，これを超えた時は発散としている．また，証明が 6 サイクル中に終了しなかった場合に補題生成を行うが，補題が生成出来なかったときに失敗としている．補題の証明は本論文の自動証明システムで行う．本実験では CPU：Intel Core i7-3540M，メモリ： 7.7GiB のコンピュータを用いて行った．実験結果を表 1 に示す．表 1 において，実装した自動証明ツール，発散鑑定法，差分照合と一般化に基づく補題生成法，右辺のランダム生成に基づく補題生成法のそれぞれ単体の補題生成能力を比較している．また，noIteration は実装した自動証明ツールにおいて，補題を証明する際に反証機能付き書き換え帰納法で証明している．なお，MSG+SG の実行数値は論文に記されている数値を記載している．

提案手法は，89 例について帰納的定理の証明に成功した例が 52 例，帰納的定理の証明に失敗した例が 37 例である．提案手法の実験結果から，健全発散鑑定法· 健全一般化法付き書き換え帰納法より強力な証明システムであるとわかる．また，発散鑑定法よりも提案した補題生成法を用いることで多くの補題を生成することができた．右辺のランダム生成に基づく補題生成法において深さを増やしていくと，成功例が少なくなっているのはタイムアウトを設定しているためである．健全発散鑑定法· 健全一般化法付き書き換え帰納法で証明できなかった例のうち，22 例が提案手法で証明に成功している．また，健全発散鑑定法· 健全一般化法付き書き換え帰納法で証明ができ，提案手法で証明できなかった例は，7 例ある．これは発散系列が必要ない健全一般化法で証明出来ていると考えられる．また，HipSpec と比べて帰納的定理の証明に成功した例は少なかった．しかし，HipSpec で証明できなかった例のうち，5 例が提案手法で証明に成功している．HipSpec と実装ツールの実行時間を比較すると，HipSpec では証明に成功する例でも時間がかかる例が存在した．しかし，実装ツールでは証明に成功する例はほとんど時間がかからずに証明が成功している．HipSpec はランダムに等式を生成しているため，全体的に実装ツールよりも時間がかかっていると考える．

提案手法で帰納的定理の証明に失敗または発散した例のうち，右辺のランダム生成に基づく補題生成法において右辺を生成する深さを変更することで証明に成功した例が 3 例存在する．また，証明に失敗または発散した原因としては，(i) 簡約順序に用いる順序，(ii) 差分照合の適用ができなかっ

た，(iii) 発散系列が存在しなかった，(iv) 補題の生成に失敗した，の 4 つが考えられる．(i) 簡約順序に用いる順序については，辞書式経路順序以外の順序を導入したり，順序を変更することで改善できると考えられる．(ii) 差分照合の適用ができなかった例について，例えば以下のような発散系列から差分照合を見つけることができなかった．

$$\begin{aligned} +(+(x,x),x) &\rightarrow +(x,+(x,x)) \\ +(+(x,\mathsf{s}(x)),\mathsf{s}(x)) &\rightarrow +(x,\mathsf{s}(+(x,\mathsf{s}(x)))) \\ &\vdots \end{aligned}$$

この発散系列は，複数の部分項が発散していることが観測できる．そのため，従来の差分照合をより一般化することで，発散系列の発見・補題生成の確率を向上できると考えられる．(iii) 発散系列が存在しない・(iv) 補題の生成に失敗した例については，発散系列の抽出法やより強力な補題生成法を考察する必要があると考える．

6 まとめ

本論文では，書き換え帰納法を利用した帰納的定理証明の補題生成法を提案した．書き換え帰納法を柔軟に実行して得られる様々な発散パターンが含まれる仮定部から複数の発散系列を抽出することで，より多くの発散系列の観測が可能となった．そして，発散鑑定法では生成に失敗する発散系列に対して補題の生成に成功する，2 つの補題生成手法を提案した．

また，提案手法に基づく帰納的定理自動証明システムを実装するとともに，実装システムを用いて実験を行った．そして，従来の健全発散鑑定法付き書き換え帰納法と比べて，有効な補題が発見できたとともに，証明能力が向上したことを示した．今後の課題としては，右辺のランダム生成に基づく補題生成法における右辺の有効な形を考察すること，より複雑な差分をもつ発散系列からの補題生成法を考案することがある．

謝辞

本論文を改善するための貴重なコメントを頂きました査読者に感謝致します．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] T. Aoto and Y. Toyama. Ground confluence prover based on rewriting induction. In *Proc. of 1st FACD*, volume 52 of *LIPIcs*, pages 33:1–33:12. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[2] T. Basin and T. Walsh. Difference matching. In *Proc. of 11th CADE*, volume 607 of *LNCS*, pages 295–309. Springer-Verlag, 1992.

[3] A. Bouhoula. Automated theorem proving by test set induction. *Journal of Symbolic Computation*, 23:47–77, 1997.

[4] A. Bouhoula, D. Basin, and A. Ireland. *Rippling: Meta-Level Fuidance for Mathematical Reasonig*. Cambridge University Press, Cambridge, 2005.

[5] A. Bouhoula, E. Kounalis, and M. Rusinowitch. Automated mathematical induction. *Journal of Logic and Computation*, 5:631–668, 1995.

[6] K. Claessen, M. Johansson, D. Rosn, and N. Smallbone. Automating inductive proofs using theory exploration. In *Proc. of 24th CADE*, volume 7898 of *LNCS*, pages 392–406. Springer-Verlag, 2013.

[7] Koen Claessen, Nicholas Smallbone, and John Hughes. Quickspec: Guessing formal specification using testing. In *Proc. of TAP*, pages 6–21, 2010.

[8] U. S. Reddy. Term rewriting induction. In *Proc. of 14th RTA*, volume 2706 of *LNCS*, pages 352–366. Springer-Verlag, 2003.

[9] Dan Rosen. Proving equational haskell properties using automated theorem provers. Master's thesis, University of Gothenburg, 2012.

[10] T. Walsh. A divergence critic for inductive proof. *Journal of Artificial Intelligence Research*, 4:209–235, 1996.

[11] 嶌津聡志, 青戸等人, and 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, 26(2):41–55, 2009.