反証機能付き書き換え帰納法のための補題自動生成法

嶌津 聡志　　青戸 等人　　外山 芳人

書き換え帰納法 (Reddy, 1989) は，項書き換えシステムにもとづく帰納的定理の自動証明法である．一般に，補題自動生成法において生成される補題はいつも正しい (定理である) とは限らないが，正しい補題のみを生成する補題自動生成法を健全であるという．発散鑑定法 (Walsh, 1996) は書き換え帰納法のために提案された補題自動生成法であるが，健全ではない．本論文では，発散鑑定法の手続きの一部を健全一般化法 (Urso と Kounalis, 2004) に置き換えることにより，単相項書き換えシステムに対して適用可能な，健全な発散鑑定法を提案する．また，これらの補題自動生成法を反証機能付き書き換え帰納法上に実装し，補題生成能力の比較実験を行う．これにより，(健全) 発散鑑定法と健全一般化法が異なる有効範囲を持つことを明らかにし，従来から知られていた健全一般化法と我々の提案する健全発散鑑定法を組み合わせることにより，書き換え帰納法のためのより強力な健全補題自動生成法が実現できることを示す．

Rewriting induction (Reddy, 1989) is an automated inductive theorem proving method for term rewriting systems. An automated lemma generation method for automated inductive theorem proving is said to be sound if it does not produce incorrect lemmas. Divergence critic (Walsh, 1996) is a well-known automated lemma generation method for the rewriting induction, but it is unsound. In this paper, we propose a sound variant of the divergence critic applicable for monomorphic term rewriting systems by incorporating sound generalization (Urso and Kounalis, 2004) in a part of its procedure. We implement these three automated lemma generation methods on a rewriting induction system with disproof to evaluate effectiveness of these methods. Our experiment reveals that the (sound) divergence critic and the sound generalization are often effective for different kinds of conjectures. Thus, the sound divergence critic can be combined with the sound generalization to obtain a more powerful automated sound lemma generation method for rewriting induction.

1　はじめに

関数型言語や代数的仕様記述法などの等式論理を基礎とする系では，さまざまな性質を等式論理の帰納的定理として取り扱うことができる．それゆえ，等式論理における帰納的定理の自動証明手法 [1] [2] [3] [6] [7] [9] [12] [13] [14] [15] は，代数的仕様やプログラムの検証，また仕様とプログラムの等価性判定などを自動的に行うために重要である．書き換え帰納法 (rewriting induction) は，Reddy [13] により提案された帰納的定理の自動証明法であり，書き換え帰納法にもとづく定理自動証明システムとしては SPIKE [6] [7] がよく知られている．

書き換え帰納法では，証明すべき等式の集合と仮定の集合の対に対して推論規則を繰り返し適用する．このとき，推論規則の適用に失敗する場合や，推論規則による導出が無限に繰り返される (発散する) 場合には，証明すべき等式が帰納的定理かどうかを判定できない．後者の場合には，推論の過程において，証明すべき等式集合へ適切な等式を補題として追加することによって，証明が成功する場合がある．

書き換え帰納法のための補題自動生成法として，Walsh の提案した発散鑑定法 (divergence critic) [17]

Automated Lemma Generation for Rewriting Induction with Disproof.
Satoshi Shimazu, NTT アクセスサービスシステム研究所, NTT Access Network Service Systems Laboratories.
Takahito Aoto, Yoshihito Toyama, 東北大学電気通信研究所, RIEC, Tohoku University.
コンピュータソフトウェア, Vol.16, No.5 (1999), pp.78–83.
[小論文] xxxx 年 yy 月 zz 日受付.

が知られている．発散鑑定法は明示的帰納法のヒューリスティクスであるリップリング法[9]を書き換え帰納法の補題生成に応用したものである．書き換え帰納法の証明過程から得られる等式系列の拡大パターンを差分照合[5]によって特定し，差分照合の差分情報から発散を収束させるのに適切な補題を生成する方法である．

補題の自動生成機能を組み込むことにより，帰納的定理自動証明システムの証明能力の向上を図ることができる．しかしながら，一般に，補題自動生成法において生成される補題はいつも正しい(帰納的定理である)とは限らない．このため，誤った補題が導入されてしまうと，本来成功するはずの証明が失敗してしまう．同様に，反証機能付き書き換え帰納法 (rewriting induction with disproof) [6] [7] 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ため，証明可能な等式集合に対して誤った補題を追加すると反証が導かれてしまう場合がある．このため，バックトラック機能を用いたり補題の正しさを別に証明するなどして，誤った補題の導入や誤った反証を防ぐ必要があるが，これはしばしば書き換え帰納法の見通しの良い実装や効率的な証明を困難にする．実際，反証機能付き書き換え帰納法を実装している SPIKE では，発散鑑定法による補題生成機能を利用者に提供しているが，生成された補題の追加はユーザーに任されており，得られた補題が自動的に追加されるわけではない．

Urso と Kounalis (2004) は，単相項書き換えシステムに対して適用可能な健全な補題自動生成法である健全一般化法 (sound generalization) を提案している[16]．健全一般化法では，項書き換えシステムの書き換え規則の構造を注意深く解析することにより，帰納的定理である等式の両辺に出現する共通部分項のうち，新しい変数に置き換えて得られた等式も帰納的定理となる共通部分項を特定する．そして，証明すべき等式から一般化して得られた等式を補題として生成する．このため，証明すべき等式が正しい場合には，得られた補題も必ず正しいものとなり，補題自動生成法の健全性が保証される．Urso と Kounalis は書き換え帰納法にもとづく定理自動証明システム NICE [15] [16] 上に健全一般化法を実装し，その有効性を示している．

一方，発散鑑定法は健全性をもたない．このため，証明する等式が正しい場合でも，誤まった補題を生成してしまうことがある．そこで，一般的には，同じ拡大パターンが 2 回続いているかなどをチェックすることにより，誤まった補題の生成が起こりにくいようにする．しかしながら，このヒューリスティクスによって健全性が理論的に保証されるわけではない．また，著者らの知る限り，発散鑑定法が健全となるための十分条件等も知られていない．

本論文では，発散鑑定法の手続きの一部を健全一般化法に置き換えることにより，単相項書き換えシステムに対して適用可能な，健全発散鑑定法を提案する．また，これらの補題自動生成法を反証機能付き書き換え帰納法上に実装し，補題生成能力の比較実験を行う．この実験により，(健全) 発散鑑定法と健全一般化法が異なる有効範囲を持つことを明らかにし，従来から知られていた健全一般化法と我々の提案する健全発散鑑定法を組み合わせることにより，書き換え帰納法のためのより強力な健全補題自動生成法が実現できることを示す．

本論文の構成は次のとおりである．2 節では必要となる定義を与え，書き換え帰納法について説明する．3 節では発散鑑定法と健全一般化法による補題生成を説明する．4 節では，健全な新しい補題生成法として健全発散鑑定法を提案し，5 節で反証機能付き書き換え帰納法上での証明実験をとおして補題生成能力の比較を行う．

2 準備

2.1 項書き換えシステム

本論文でもちいる定義および記法を紹介する．詳細は文献[4] [10] 等を参照されたい．

ソート付き関数記号集合およびソート付き変数集合を $\mathcal{F}, V$ で表す．$\mathcal{F}, V$ 上のソート付き項の集合を $T(\mathcal{F}, V)$ で表す．項 t の**根記号**とは $t = x \in V$ のときは x，$t = f(t_1, \ldots, t_n)$ $(f \in \mathcal{F})$ のときは f を指す．$C[t]$ は文脈 C のホールを項 t で置き換えて得られる項を表わす．項 u が t の**部分項**である $(u \trianglelefteq t)$ と

は，$t = C[u]$ なる文脈 C があるときをいう．

項 t に現れる変数がすべて異なるとき，t は**線形**であるという．t に現れる変数の集合を $V(t)$ と記す．$V(t) = \emptyset$ となる t を**基底項**とよぶ．定義域上の任意の変数 x について $\sigma_g(x)$ が基底項となる代入 σ_g を**基底代入**とよぶ．本論文では，項 t に対する基底代入 σ_g を考えるときには，$V(t\sigma_g) = \emptyset$ をみたすものと仮定する．

同一ソートの項 s, t の対を**等式**とよび，$s \doteq t$ と記す．ここで，等式 $s \doteq t$ と $t \doteq s$ は同一視する．$l \notin V$ かつ $V(r) \subseteq V(l)$ なる同一ソートの項 l, r の対を**書き換え規則**とよび，$l \to r$ と記す．書き換え規則の集合を**項書き換えシステム**という．以下では，項書き換えシステムからソートが自明な場合には，ソートを省略する．

項書き換えシステムが**左線形**であるとは，すべての書き換え規則の左辺が線形であるときをいう．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**とよぶ．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．項 $t \in T(\mathcal{C}, V)$ を**構成子項**とよぶ．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．$\mathcal{B}(t) = \{u \trianglelefteq t \mid u \text{ は基本項 }\}$ は項 t の**基本部分項**集合を表す．項書き換えシステムが**構成子システム**であるとは，すべての書き換え規則の左辺が基本項であるときをいう．

$\mathcal{R}$ を項書き換えシステムとする．文脈 C，代入 σ，書き換え規則 $l \to r \in \mathcal{R}$ が存在して，$s = C[l\sigma]$ かつ $t = C[r\sigma]$ となるとき，$s \to_{\mathcal{R}} t$ と記す．$\to_{\mathcal{R}}$ の反射推移閉包を $\xrightarrow{*}_{\mathcal{R}}$，反射対称推移閉包を $\xleftrightarrow{*}_{\mathcal{R}}$ と記す．$t \to_{\mathcal{R}} u$ となる項 u が存在しないとき，項 t を**正規形**とよび，$s \xrightarrow{*}_{\mathcal{R}} t$ なる正規形 t を s の正規形とよぶ．

任意の (基底) 項 t, t_1, t_2 について $t_1 \xleftarrow{*}_{\mathcal{R}} t \xrightarrow{*}_{\mathcal{R}} t_2$ ならば，ある (基底) 項 s が存在して $t_1 \xrightarrow{*}_{\mathcal{R}} s \xleftarrow{*}_{\mathcal{R}} t_2$ となるとき，$\mathcal{R}$ は **(基底) 合流性**をもつという．$\to_{\mathcal{R}}$ が整礎であるとき，$\mathcal{R}$ は**停止性**をもつという．代入および文脈に閉じている整礎な半順序を**簡約順序**とよぶ．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 $t \xrightarrow{*}_{\mathcal{R}} s$ となるとき，$\mathcal{R}$ は**十分完全性** [11] をもつという．

本論文では，項書き換えシステム $\mathcal{R}$ は停止性と十分完全をもつ構成子システムとする．

2.2 帰納的定理と書き換え帰納法

等式 $s \doteq t$ が項書き換えシステム $\mathcal{R}$ の**帰納的定理**であるとは，任意の基底代入 σ_g に対して $s\sigma_g \xleftrightarrow{*}_{\mathcal{R}} t\sigma_g$ が成り立つことをいい，$\mathcal{R} \models_{\mathrm{ind}} s \doteq t$ と記す．等式集合 E に含まれるすべての等式が $\mathcal{R}$ の帰納的定理であるとき，E を $\mathcal{R}$ の帰納的定理とよび，$\mathcal{R} \models_{\mathrm{ind}} E$ と記す．なお，本論文では，任意の等式が帰納的定理となるような自明な理論をさけるために，どのソートについても 2 個以上の基底構成子項が存在するものと仮定する．

例 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．

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

このとき等式 $x + 0 \doteq x$ を考えると，$x + 0 \xleftrightarrow{*}_{\mathcal{R}} x$ は成立しない．しかし，任意の基底代入 σ_g に対して $(x + 0)\sigma_g \xleftrightarrow{*}_{\mathcal{R}} x\sigma_g$ が成立するので，$x + 0 \doteq x$ は $\mathcal{R}$ の帰納的定理である． □

帰納的定理の自動証明法として**反証機能付き書き換え帰納法** (rewriting induction with disproof) [6] [7] が知られている．反証機能付き書き換え帰納法は等式集合 E と書き換え規則集合 H (以下，仮定集合とよぶ) の対 $\langle E, H \rangle$ に関する図 1 の推論規則として与えられる．ここで，$\uplus$ は直和，$>$ は $\mathcal{R} \subseteq >$ なる簡約順序，$\mathrm{mgu}(s, t)$ 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操作である．

$$\mathrm{Expd}_u(s, t) = \{C[r]\sigma \doteq t\sigma \mid s = C[u], \\ \sigma = \mathrm{mgu}(u, l), l \to r \in \mathcal{R}\}$$

$\langle E, H \rangle$ に推論規則を適用して $\langle E', H' \rangle$ が得られたときの**導出**を $\langle E, H \rangle \vdash \langle E', H' \rangle$ と記す．$\vdash$ の反射推移閉包を $\vdash^*$ と記す．このとき以下の命題が成り立つ．

命題 1 (反証機能付き書き換え帰納法の正当性 [6] [7])．$\mathcal{R}$ は基底合流性，停止性と十分完全性をもつ構成子システムとする．このとき，*(1)* $\langle E, \emptyset \rangle \vdash^* \langle \emptyset, H \rangle$ ならば $\mathcal{R} \models_{\mathrm{ind}} E$．*(2)* $\langle E, \emptyset \rangle \vdash^* \mathrm{Disproof}$ ならば $\mathcal{R} \not\models_{\mathrm{ind}} E$．

Simplify

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

Delete

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

Expand

$$\frac{\langle E \uplus \{s \doteq t\},\ H\rangle}{\langle E \cup \mathrm{Expd}_u(s,t),\ H \cup \{s \to t\}\rangle}\ u \in \mathcal{B}(s),\ s > t$$

Decompose

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

Disproof

$$\frac{\langle E \uplus \{s \doteq x\}, H\rangle}{\text{Disproof}}\ x \in V \setminus V(s)$$

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

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

図 1 反証機能付き書き換え帰納法の推論規則

本論文では，反証機能付き書き換え帰納法の手続きを以下の戦略にもとづいて実現する．まず，与えられた $\langle E, H\rangle$ に対して，*Simplify* 規則を繰り返し適用し，等式集合 E の各等式の両辺の正規形を求める．ついで，*Delete* 規則を繰り返し適用し，等式集合 E から両辺が同じ等式を取り除いて $\langle E_0, H_0\rangle$ を得る．$\langle E_n, H_n\rangle$ から $\langle E_{n+1}, H_{n+1}\rangle$ $(n \geq 0)$ を得るためには，$\langle E_n, H_n\rangle$ に対して *Expand* 規則の適用を試み，適用が不可能である場合には証明は失敗して終了する．成功した場合には，*Simplify* 規則を繰り返し適用し，等式集合 E の各等式の両辺の正規形を求める．ついで，*Delete* 規則を繰り返し適用し，等式集合 E から両辺が同じ等式を取り除く．次に，等式集合 E の各等式に *Decompose* 規則を可能な限り適用する．*Decompose* 規則の適用中に *Disproof* 規則が適用できたら，非定理が存在するので終了する．

手続きで得られた系列を

$$\langle E_0, H_0\rangle \rightsquigarrow \langle E_1, H_1\rangle \rightsquigarrow \langle E_2, H_2\rangle \rightsquigarrow \cdots \rightsquigarrow \langle E_n, H_n\rangle \rightsquigarrow \langle E_{n+1}, H_{n+1}\rangle \rightsquigarrow \cdots$$

と記す．この系列が無限となる場合には，反証機能付き書き換え帰納法は**発散する**という．また，仮定集合 H_n $(n \geq 0)$ は単調増加で，$|H_n \setminus H_{n-1}| = 1$ であることに注意する．

2.3 書き換え帰納法における補題導入

書き換え帰納法が発散するときには，証明の途中で，適当な補題集合 L を等式集合 E に追加すると，証明に成功する場合が多い．補題導入のための推論規則は，次の *Postulate* 規則によって与えられる．

Postulate

$$\frac{\langle E,\ H\rangle}{\langle E \cup L,\ H\rangle}$$

Postulate 規則を加えた反証機能付き書き換え帰納法の導出を $\vdash_p$ と記す．*Postulate* 規則を反証機能付き書き換え帰納法の推論規則に加えると，一般には，反証機能付き書き換え帰納法の正当性は成立しなくなる．実際には，命題 1 のうち，性質 (1) はそのまま成立するが，性質 (2) が成立しなくなる．なぜなら，E を書き換え帰納法による証明に成功するような等式集合として，導出の途中において，*Postulate* 規則を用いて非定理 $L = \{\mathsf{true} \doteq \mathsf{false}\}$ を補題として追加し (ここで true, false は異なる構成子定数とする)，*Disproof* 規則を用いると，反証に成功してしまう．つまり，$\mathcal{R} \models_{ind} E$ にもかかわらず，$\langle E, \emptyset\rangle \vdash_p^* \text{Disproof}$ となってしまう．従って，反証機能の正当性を保証するためには，*Postulate* 規則の適用に制約を課す必要がある．

定義 1 (*Postulate* 規則の健全性)．*Postulate* 規則による導出 $\langle E, H\rangle \vdash_p \langle E \cup L, H\rangle$ が**健全**であるは，$\mathcal{R} \models_{\mathrm{ind}} E \cup H$ ならば $\mathcal{R} \models_{\mathrm{ind}} L$ が成立しているときをいう．また，導出 $\langle E, \emptyset\rangle \vdash_p^* \text{Disproof}$ が健全であるとは，この導出に含まれる *Postulate* 規則による導出が全て健全であるときをいう．

このとき，次の性質がなりたつ．

命題 2 (補題導入法の健全性 [6] [7])．$\mathcal{R}$ は基底合流性，

停止性と十分完全性をもつ構成子システムとする．このとき，*(1)* $\langle E, \emptyset\rangle \vdash_p^* \langle \emptyset, H\rangle$ ならば $\mathcal{R} \models_{\mathrm{ind}} E$．*(2)* $\langle E, \emptyset\rangle \vdash_p^*$ Disproof が健全であるならば，$\mathcal{R} \not\models_{\mathrm{ind}} E$．

なお，以下では，文脈から自明な場合，$\vdash_p$ の添字 $_p$ は省略する．また，簡単のために，反証機能付き書き換え帰納法を書き換え帰納法とよぶ．

3 補題の自動生成法

証明過程において補題集合 L を自動生成する手法として，Walsh の提案した発散鑑定法 (divergence critic) [17]，Urso と Kounalis が提案した健全一般化法 (sound generalization) [16] などが知られている．本節ではこれらの補題生成法について簡単に説明し，書き換え帰納法の手続きに組み込む方法について述べる．

3.1 発散鑑定法

書き換え帰納法が無限系列 $\langle E_i, H_i\rangle$ $(i \geq 0)$ を生じて発散する場合，同じパターンにより拡大する系列 (**発散系列**) が $H_i, H_{i+1}, \ldots$ において観測されることがある．発散鑑定法は $H_0 \subseteq H_1 \subseteq \cdots$ に含まれる書き換え規則における拡大パターンを発見した時に，その拡大パターンによる発散系列を収束させるような書き換え規則を構成する．以下では，発散鑑定法による補題生成を簡単な例で説明する．

例 2 (発散鑑定法による補題生成例)．リスト結合と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)．等式集合 E に書き換え帰納法を適用する．

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

$$E = \left\{ \mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) \doteq xs \mathbin{@} ys \right.$$

このとき E は $\mathcal{R}$ の帰納的定理である．しかし，書き換え帰納法は発散し，$H_i, H_{i+1}, \ldots$ において，以下の発散系列が観測される．

$$\begin{array}{rcll} \mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) & \to & xs \mathbin{@} ys & (1) \\ \mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys) \mathbin{@} [z]) & \to & z :: (zs \mathbin{@} ys) & (2) \\ \mathrm{rev}((\mathrm{rev}(vs @ ys) @ [v]) @ [z]) & \to & z :: (v :: (vs @ ys)) & \\ & \vdots & & (3) \end{array}$$

発散鑑定法は上記の発散系列の最初の 2 つの書き換え規則から拡大パターンを見つけ，そのパターンから補題を得る．拡大パターンを捉えるために 2 つの書き換え規則の**差分照合** (difference matching) を考える．たとえば，書き換え規則 (1) と (2) の差分照合は，書き換え規則 (2) に以下のような枠と下線を付け加えることによって表現される．

$$\mathrm{rev}(\boxed{\underline{\mathrm{rev}(zs \mathbin{@} ys)} \mathbin{@} [z]}) \quad \to \quad \boxed{z :: (\underline{zs \mathbin{@} ys})} \quad (4)$$

ここで，差分照合 (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 (1) と同じ書き換え規則が得られることに注意する．(枠 (wave-front) と下線 (wave-hole) の正確な定義については文献 [5] [9] を参照のこと．)

差分照合 (4) から補題を次のようにして生成する．差分照合 (4) の右辺の下線部 $zs \mathbin{@} ys$ は書き換え規則 (1) の右辺 $xs \mathbin{@} ys$ と照合するので，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等式 (5) を得る．

$$\mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys) \mathbin{@} [z]) \quad \doteq \quad z :: \mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys)) \quad (5)$$

このように書き換え規則を逆向きに適用することによってもうひとつの書き換え規則の右辺を変形することを**書き換え規則変形操作** M (modification) とよぶ．

差分照合 (4) の左辺の下線部 $\mathrm{rev}(zs \mathbin{@} ys)$ は，差分情報から等式 (5) の右辺にも現れることがわかる．そこで，等式 (5) の両辺に現れる共通部分項 $\mathrm{rev}(zs \mathbin{@} ys)$ を新しい変数 ws で置き換えると以下の等式 (6) が補題として生成される．

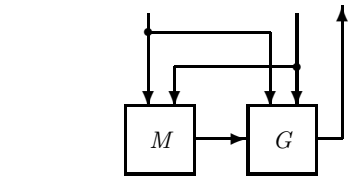

図 2 発散鑑定法付き書き換え帰納法 MG の補題生成

入力：仮定集合 H_{n-1}, H_n
出力：補題集合 $L = \{C_1\sigma[C_2[x]] \doteq C_3[C_1\sigma[x]]\}$

$$\begin{array}{rcll} C_1[s] & \to & t & \in H_{n-1} \\ C_1\sigma[\boxed{C_2[s\sigma]}] & \to & \boxed{C_3[t\sigma]} & \in H_n \setminus H_{n-1} \end{array}$$

(σ :変数改名)

$$\begin{array}{rcl} \stackrel{M}{\Longrightarrow} & C_1\sigma[C_2[s\sigma]] \doteq C_3[C_1\sigma[s\sigma]] \\ \stackrel{G}{\Longrightarrow} & C_1\sigma[C_2[x]] \doteq C_3[C_1\sigma[x]] \end{array}$$

(x:新しい変数)

図 3 発散鑑定法の手続き

$$\mathrm{rev}(ws \mathbin{@} [z]) \doteq z :: \mathrm{rev}(ws) \qquad (6)$$

等式 $s' \doteq t'$ が等式 $s \doteq t$ の**一般化**であるとは，ある代入 σ が存在して $s'\sigma = s$ かつ $t'\sigma = t$ となることをいう．補題 (6) は等式 (5) の一般化になっている．**等式一般化操作**を G (generalization) で表わす．

補題 (6) は，同じ拡大パターンによる発散系列を解消する補題となっている．実際，書き換え規則 (3) は，補題 (6) と書き換え規則 (2) を用いて自明な等式に変形できる．このため，補題 (6) の追加により発散が抑えられ，証明に成功する． □

発散鑑定法付き書き換え帰納法 MG における各手続きの接続を図 2 に示す．発散鑑定法では，書き換え帰納法の手続きで得られた系列 $\langle E_{n-1}, H_{n-1}\rangle \rightsquigarrow \langle E_n, H_n\rangle$ において，H_n に新しく追加された書き換え規則との間に差分をもつ H_{n-1} の書き換え規則を発見すると，その差分情報に基づいて，書き換え規則変形操作 M および等式一般化操作 G を適用し，補題を生成する．図 3 に発散鑑定法の手続きを示す．なお，1 ステップで生成される補題は高々 1 個であることに注意する．

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，図 3 の手続きは正しくない補題 (非定理) を生成することがある．

例 3 (発散鑑定法の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 $\mathcal{R}$ で与える．等式集合 E に書き換え帰納法を適用する．

$$\mathcal{R} = \left\{ \begin{array}{lcl} \mathrm{qrev}([\,], ys) & \to & ys \\ \mathrm{qrev}(x :: xs, ys) & \to & \mathrm{qrev}(xs, x :: ys) \end{array} \right.$$

$$E = \left\{ \mathrm{qrev}(\mathrm{qrev}(xs, [\,]), [\,]) \doteq xs \right.$$

このとき E は $\mathcal{R}$ の帰納的定理である．しかし，書き換え帰納法は発散し，$H_i, H_{i+1}, \ldots$ において，以下の発散系列が観測される．

$$\begin{array}{rcll} \mathrm{qrev}(\mathrm{qrev}(xs, [\,]), [\,]) & \to & xs & (7) \\ \mathrm{qrev}(\mathrm{qrev}(ys, y :: [\,]), [\,]) & \to & y :: ys & (8) \\ \mathrm{qrev}(\mathrm{qrev}(zs, z::(y::[\,])), [\,]) & \to & y::(z::zs) & (9) \\ & & \vdots & \end{array}$$

書き換え規則 (7) と (8) の差分照合は

$$\mathrm{qrev}(\mathrm{qrev}(ys, \boxed{y :: \underline{[\,]}}), [\,]) \to \boxed{y :: \underline{ys}} \qquad (10)$$

となる．差分照合 (10) の差分情報から操作 M をもちいて書き換え規則 (8) を書き換え規則

$$\mathrm{qrev}(\mathrm{qrev}(ys, y :: [\,]), [\,]) \to y :: \mathrm{qrev}(\mathrm{qrev}(ys, [\,]), [\,])$$

に変形する．この書き換え規則に操作 G をもちいて共通部分項 $[\,]$ を新しい変数 ws に置き換えると補題

$$\mathrm{qrev}(\mathrm{qrev}(ys, y :: ws), [\,]) \doteq y :: \mathrm{qrev}(\mathrm{qrev}(ys, ws), [\,])$$

を得る．しかし，この補題は非定理である． □

このように正しくない補題が生成されてしまった原因は，等式一般化操作 G が定理から非定理を生成してしまう可能性があるためである．例えば，等式 $\mathrm{rev}([\,]) \doteq [\,]$ は定理であるが，部分項 $[\,]$ を一般化して得られる等式 $\mathrm{rev}(xs) \doteq xs$ は定理でない．

発散鑑定法はこのような非定理の生成を防ぐために，複数の差分照合を観察する．たとえば，書き換え規則 (8) と書き換え規則 (9) の差分照合

$$\mathrm{qrev}(\mathrm{qrev}(zs, \boxed{z :: (\underline{y :: [\,]})}), [\,]) \to y :: \boxed{(\underline{z :: zs})}$$

の右辺は，発散鑑定法の手続きに記述されている差

分照合の右辺の形と異なるために補題を生成できない．複数の差分照合を観察することで，補題生成に必要な計算時間が増加するが，非定理を生成する可能性は低くなると考えられる．

文献[17]では，経験的に3つの書き換え規則から2つの差分照合を観察する方法が有効であると述べられている．本論文の発散鑑定法もこの方法を採用する．つまり，H_n に新しく追加された書き換え規則との間に差分をもつ H_{n-1} の書き換え規則 $l \to r$ を発見した場合，H_{n-1} の書き換え規則の中に，$l \to r$ との間に同じ差分をもつ書き換え規則が存在するかを探索し，存在しなければ補題の生成を見送る．このようにして例3の非定理の生成を防止している．

3.2 健全一般化法

健全一般化法は，単相項書き換えシステム $\mathcal{R}$ の構造にもとづき，等式に出現する共通部分項のうち，適当な変数に置き換えても定理が非定理になることがない部分項を特定し，そのような部分項のみを一般化することによって健全性を保証する[16]．以下では，まず，単相項書き換えシステムについて簡単に紹介する．

ソート集合 $\mathcal{S}$ とソート付き関数記号集合 $\mathcal{F}$ の対 $\langle \mathcal{S}, \mathcal{F} \rangle$ をシグネチャとよぶ．関数記号 f がソート $\alpha_1 \times \cdots \times \alpha_n \to \beta$ を持つことを，$f : \alpha_1 \times \cdots \times \alpha_n \to \beta$ と記す．特に $n = 0$ のとき，$f : \to \beta$ は単に $f : \beta$ と書く．異なるソート α と β に対して，ある構成子記号 $f : \alpha_1 \times \cdots \times \beta \times \cdots \times \alpha_n \to \alpha \in \mathcal{C}$ が存在するときに $\alpha \succ_{\mathcal{S}} \beta$ と記す．**単相シグネチャ** (monomorphic signature) とは次の条件をみたすシグネチャをいう：(1) $\mathcal{S}$ 上の関係 $\succ_{\mathcal{S}}$ は狭義の半順序，(2) 任意のソート $\alpha \in \mathcal{S}$ に対して，(2-a) 構成子記号 $f : \alpha \in \mathcal{C}$ がただ一つ存在し，(2-b) 構成子記号 $f : \alpha_1 \times \cdots \times \alpha_n \to \alpha \in \mathcal{C}$ $(n \geq 1)$ に対して，$\alpha = \alpha_i$ となる i がただ一つ存在する．単相シグネチャの直感的な意味は，単相シグネチャ上の構成子項がリスト構造になることである．

例 4 (単相シグネチャ)．

$$\mathcal{S} = \{\mathrm{Nat},\ \mathrm{List}\},$$
$$\mathcal{C} = \left\{ \begin{array}{l} 0 : \mathrm{Nat},\ \mathrm{s} : \mathrm{Nat} \to \mathrm{Nat}, \\ [\,] : \mathrm{List},\ :: : \mathrm{Nat} \times \mathrm{List} \to \mathrm{List} \end{array} \right\},$$
$$\mathcal{D} = \left\{ \begin{array}{l} + : \mathrm{Nat} \times \mathrm{Nat} \to \mathrm{Nat}, \\ @\ : \mathrm{List} \times \mathrm{List} \to \mathrm{List} \end{array} \right\}.$$

このとき，シグネチャ $\langle \mathcal{S}, \mathcal{C} \cup \mathcal{D} \rangle$ は単相である． □

単相項書き換えシステムとは，単相シグネチャ上の基底合流性，停止性と十分完全性をもつ左線形項書き換えシステムをいう[16]．健全一般化法による補題生成を簡単な例で説明する．

例 5 (健全一般化法による補題生成例)．例1の単相項書き換えシステム $\mathcal{R}$ をもちいて，以下の等式集合 E に対して書き換え帰納法を適用する．

$$E = \{\ \ (x + x) + x \doteq x + (x + x) \qquad (11)$$

Expand 規則および *Simplify* 規則を E に適用すると，等式 $\mathrm{s}((x + \mathrm{s}(x)) + \mathrm{s}(x)) \doteq \mathrm{s}(x + \mathrm{s}(x + \mathrm{s}(x)))$ が得られる．しかし，*Expand* 規則で得られた書き換え規則 $(x + x) + x \to x + (x + x) \in H$ はこの等式に適用できない．このため，等式は再び *Expand* 規則により展開される．この過程を繰り返すことにより書き換え帰納法は発散する．

一方，等式 (11) の共通部分項 x の一部を変数 y, z に置き換えて一般化して得られる等式 (12) を補題として E に追加した場合を考える．

$$E' = \left\{ \begin{array}{l} (x + x) + x \doteq x + (x + x) \\ (x + y) + z \doteq x + (y + z) \end{array} \right. \qquad (12)$$

Expand 規則を等式 (12) に適用すると，等式 $\mathrm{s}((x + y) + z) \doteq \mathrm{s}(x + (y + z))$ が得られる．今度は，*Expand* 規則で得られた書き換え規則 $(x + y) + z \to x + (y + z) \in H$ は，この等式の左辺に適用できるだけでなく，等式 $(x + x) + x \doteq x + (x + x)$ にも適用できるので，等式 (11) だけの場合に生じていた発散は生じず，証明に成功する． □

この例のように，一般化した等式を補題として追加することで証明に成功する場合がある．しかし，等式の共通部分項を適当に変数で置き換えるような一般化法では非定理となる補題を生成することがある．

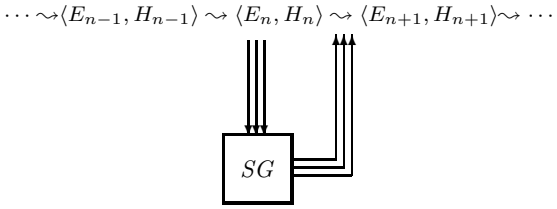

図 4 健全一般化法付き書き換え帰納法 *SG* の補題生成

これに対し，健全一般化法ではそのようなことはない．健全一般化法では，構成子記号の反射的引数位置 (reflective argument position) や定義記号の下降位置 (downward position) および上昇位置 (upward position) などを解析して，一般化すべき共通部分項の位置を特定するためである [16]．このような**健全一般化操作**を SG (sound generalization) と記す．

命題 3 (健全一般化法の健全性 [16])．$\mathcal{R}$ は単相項書き換えシステム，等式 $s' \doteq t'$ は等式 $s \doteq t$ から健全一般化法により生成された補題とする．このとき，$\mathcal{R} \models_{\mathrm{ind}} s \doteq t$ ならば $\mathcal{R} \models_{\mathrm{ind}} s' \doteq t'$．

Postulate 規則で導入される補題が健全一般化法で生成されるものとすると，命題 3 から *Postulate* 規則は健全となるので，命題 2 より反証機能の正当性は保証される．

健全一般化法付き書き換え帰納法 SG における手続きの接続を図 4 に示す．SG では，書き換え帰納法で得られた $\langle E_n, H_n\rangle$ $(n \geq 0)$ の等式集合 E_n に含まれる等式それぞれに対して，健全一般化法を用いて補題生成を試みる．なお，E_n のすべての等式に対して一般化を行うため，系列の各ステップで生成される補題の最大個数は E_n の要素数となることに注意する．

3.3 発散鑑定法と健全一般化法の組み合わせ

後述する 5 節の表 1 で明らかになるが，発散鑑定法と健全一般化法の有効範囲が異なることを示す例題が存在する．そこで，書き換え帰納法の手続きの系列の各ステップにおいて，発散鑑定法で生成される補題と健全一般化法で生成される補題を同時に追加することで，両者の有効範囲を併せもつ補題生成を実現する方法が考えられる．発散鑑定・健全一般化法付き書き換え帰納法 MG+SG における各手続きの接続は

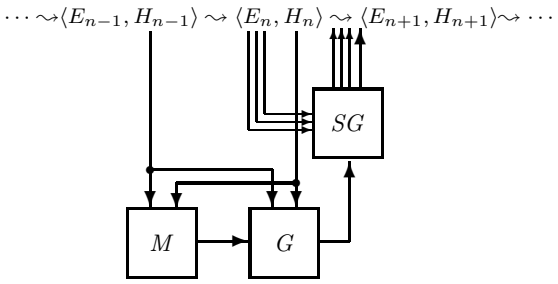

図 5 発散鑑定・健全一般化法付き書き換え帰納法 *MG*+*SG* の補題生成

図 5 となる．ここでは，発散鑑定法で生成された補題に対して健全一般化法を適用するので，MG よりも一般的な補題生成が可能となる．したがって，MG と SG のどちらも発散する場合にも MG+SG は証明に成功することがある．一方，この組み合わせでは，発散鑑定法が健全でないため，導入前に補題の正当性を別に証明しておかないことには，反証機能の正しさを保証することが出来ない．

そこで，この問題点を解決するために，次節では発散鑑定法の手続きの一部を健全一般化法におきかえた新しい補題自動生成法 (健全発散鑑定法) を提案する．

4 健全な発散鑑定法

発散鑑定法による補題生成の本質は，発散系列の差分情報にもとづく書き換え規則変形操作 M と等式一般化操作 G にあり，操作 M は健全一般化法のような静的な一般化のみでは困難な補題生成にきわめて有効に働く．一方，差分情報にもとづく等式一般化操作 G は，発散鑑定法の健全性が保証されない原因となっている．そこで，発散鑑定法の等式一般化操作 G を，健全一般化操作 SG に置き換えることが考えられる．この置き換えによって，従来の発散鑑定法の有効範囲を保ったままで健全性が保証されることが期待できる．このような健全発散鑑定法付き書き換え帰納法 MSG における各手続きの接続を図 6 で示す．以下では，簡単な例で健全発散鑑定法の手続きを説明する．

例 6 (健全発散鑑定法による補題生成例)．例 2 の単相項書き換えシステム $\mathcal{R}$ をもちいて，以下の等式集合 E に対して書き換え帰納法を適用する．

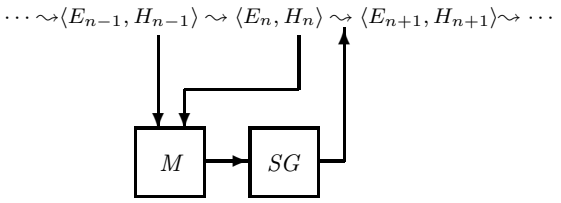

図 6　健全発散鑑定法付き書き換え帰納法 *MSG* の補題生成

入力：仮定集合 H_{n-1}, H_n
出力：補題集合 $L = \{u \doteq v\}$

$$
\begin{array}{lcll}
s & \to & t & \in H_{n-1} \\
s' & \to & C[t\sigma] & \in H_n \setminus H_{n-1} \qquad (\sigma : \text{変数改名}) \\
\stackrel{M}{\Longrightarrow} & s' & \doteq & C[s\sigma] \\
\stackrel{SG}{\Longrightarrow} & u & \doteq & v
\end{array}
$$

図 7　健全発散鑑定法の手続き

$$E = \left\{ \ \mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) \quad \doteq \quad xs \mathbin{@} ys \right.$$

このとき，例 2 ですでに述べたように，書き換え帰納法は発散し，$H_i, H_{i+1}, \ldots$ において，以下の発散系列を観測する．

$$
\begin{array}{rcll}
\mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) & \to & xs \mathbin{@} ys & (13) \\
\mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys) \mathbin{@} [z]) & \to & z :: (zs \mathbin{@} ys) & (14) \\
\mathrm{rev}((\mathrm{rev}(vs @ ys) @ [v]) @ [z]) & \to & z :: (v :: (vs @ ys)) & \\
\vdots & & &
\end{array}
$$

ここで，書き換え規則 (13) を右辺から左辺へ逆向きに使って，書き換え規則 (14) の右辺を書き換えると次の等式 (15) を得る．

$$\mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys) \mathbin{@} [z]) \quad \doteq \quad z :: \mathrm{rev}(\mathrm{rev}(zs \mathbin{@} ys)) \qquad (15)$$

このとき，書き換え規則 (13) と (14) が帰納的定理ならば，(14) を (13) によって変形して得られた等式 (15) も帰納的定理となる．次に，等式 (15) に対して健全一般化法を適用すると，両辺の共通部分項 $\mathrm{rev}(zs \mathbin{@} ys)$ を新しい変数 ws に置き換えた等式 (16) が補題として生成される．

$$\mathrm{rev}(ws \mathbin{@} [z]) \quad \doteq \quad z :: \mathrm{rev}(ws) \qquad (16)$$

ここで，$\mathcal{R}$ は単相項書き換えシステムであるから，命題 3 より等式 (16) は健全な補題であることが保証される．さらに，例 2 で発散鑑定法によって生成された補題 (6) と補題 (16) は一致していることに注意する．つまり，発散鑑定法で生成された補題が，発散系列に含まれる書き換え規則の変形と健全一般化の組み合わせで健全に生成されたことになる． □

健全発散鑑定法の手続きを図 7 に示す．ここでは，発散系列に含まれる 2 つの書き換え規則をもちいて変形を行い，その結果に健全一般化法を適用している．健全発散鑑定法では，補題生成に差分情報が必要ないので，発散鑑定法の書き換え規則変形操作 M よりも一般的な条件で書き換え規則の変形を行っている．単相項書き換えシステムに対して，この手続きが健全な補題を生成することは以下の定理により保証される．

定理 1 (健全発散鑑定法の健全性)．$\mathcal{R}$ は単相項書き換えシステム，等式 $u \doteq v$ は書き換え規則集合 $H_n, H_{n-1} \subseteq H_n$ から健全発散鑑定法により生成された補題とする．このとき，$\mathcal{R} \models_{\mathrm{ind}} H_n$ ならば $\mathcal{R} \models_{\mathrm{ind}} u \doteq v$．

(証明) このとき，ある $s \to t \in H_{n-1}$，$s' \to C[t\sigma] \in H_n \setminus H_{n-1}$ が存在して，$s \to t, s' \to C[t\sigma] \stackrel{M}{\Longrightarrow} s' \to C[s\sigma]$ および $s' \to C[s\sigma] \stackrel{SG}{\Longrightarrow} u \doteq v$ とおける (図 7)．$\mathcal{R} \models_{\mathrm{ind}} H_n$ と仮定する．すると，$s \doteq t, s' \doteq C[t\sigma]$ は $\mathcal{R}$ の帰納的定理であるから，任意の σ_g について，$s'\sigma_g \stackrel{*}{\leftrightarrow}_{\mathcal{R}} C[t\sigma]\sigma_g = C\sigma_g[t\sigma\sigma_g] \stackrel{*}{\leftrightarrow}_{\mathcal{R}} C\sigma_g[s\sigma\sigma_g] = C[s\sigma]\sigma_g$ が成立し，$s' \doteq C[s\sigma]$ も $\mathcal{R}$ の帰納的定理である．ゆえに，健全一般化法の健全性 (命題 3) より，$\mathcal{R} \models_{\mathrm{ind}} u \doteq v$． □

Postulate 規則で導入される補題が健全発散鑑定法で生成されるものとすると，定理 1 から *Postulate* 規則は健全となるので，健全一般化法の場合と同様に命題 2 より反証機能の正当性は保証される．

このようにして得られる健全発散鑑定法は，次節で示すように単相項書き換えシステムに対して発散鑑定法と類似の補題を生成し，また，健全一般化法とは異なる有効範囲を持つ．このため，従来の健全一

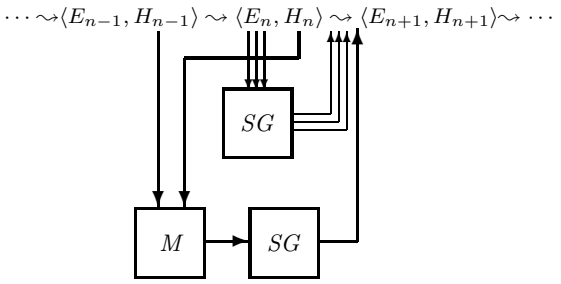

図 8 健全発散鑑定・健全一般化法付き書き換え帰納法 *MSG+SG* の補題生成

般化法と健全発散鑑定法を組み合わせることにより，より強力な健全補題自動生成法が実現できる．このような健全発散鑑定・健全一般化法付き書き換え帰納法 *MSG+SG* における各手続きの接続を図 8 で示す．なお，命題 2，命題 3，定理 1 より，健全発散鑑定・健全一般化法付き書き換え帰納法 *MSG+SG* の反証機能の正当性は明らかである．

5 実験および考察

これまで議論してきた補題生成手法を組み込んだ書き換え帰納法の実装および実験を行った．実装には関数型言語 SML をもちい，プログラムコードは約 2000 行である．本実験では，単相項書き換えシステム $\mathcal{R}$ を対象に，(1) 書き換え帰納法 *RI*，(2) 発散鑑定法付き書き換え帰納法 *MG*，(3) 健全一般化法付き書き換え帰納法 *SG*，(4) 健全発散鑑定法付き書き換え帰納法 *MSG*，(5) 健全発散鑑定・健全一般化法付き書き換え帰納法 *MSG+SG* の証明能力および反証能力の比較を行った．

なお，健全発散鑑定法や健全一般化法は，単相な項書き換えシステムに対してのみ，適用可能であるが，発散鑑定法は，単相でない項書き換えシステムに対しても適用が可能であることに注意する．

さらに，我々のシステムの有効性の指標として定理自動証明システム SPIKE と NICE との比較も行った．NICE は帰納的定理の証明手法として 2 つの機能をもつ．ひとつは健全一般化法付き書き換え帰納法にもとづく手法 (NICE$_{SG}$ と記す) と，もうひとつは項分割 (term partition) [15] とよばれる証明手法 (NICE$_{TP}$ と記す) である．以下では，両者の機能をもつシステムを NICE と記す．なお，我々の実験システムでは，等式論理上の帰納的定理のみを取り扱うが，SPIKE，NICE では，擬等式論理 (条件付等式やホーン節) 上の帰納的定理の扱いも可能である．

システムの証明能力を比較するため，*RI* で発散する 33 個の帰納的定理 (等式 1–33) の証明を試みた．自然数上の加算・乗算，リスト結合およびリスト反転を項書き換えシステム $\mathcal{R}$ で与える．$\mathcal{R}$ は，合流性，停止性および十分完全性をもっており，単相項書き換えシステムでもある．なお，$\mathcal{R}$ が合流性をみたせば，基底合流性もみたしていることに注意する．

$$\mathcal{R} = \begin{cases} 0 + y & \to & y \\ \mathrm{s}(x) + y & \to & \mathrm{s}(x + y) \\ 0 * y & \to & 0 \\ \mathrm{s}(x) * y & \to & y + (x * y) \\ [\,] \mathbin{@} ys & \to & ys \\ (x :: xs) \mathbin{@} ys & \to & x :: (xs \mathbin{@} ys) \\ \mathrm{rev}([\,]) & \to & [\,] \\ \mathrm{rev}(x :: xs) & \to & \mathrm{rev}(xs) \mathbin{@} [x] \\ \mathrm{qrev}([\,], ys) & \to & ys \\ \mathrm{qrev}(x :: xs, ys) & \to & \mathrm{qrev}(xs, x :: ys) \end{cases}$$

簡約順序としては，関数記号上の順序 qrev $>$ rev $>$ $*$ $>$ $+$ $>$ @ $>$ :: $>$ [ ] $>$ s $>$ 0 に基づく辞書式経路順序を用いた．また，タイムアウトを 10 秒に設定し，証明が終了しないときは発散としている．使用した簡約順序やタイムアウトは，これ以降の実験例についても同様である．

実験結果を表 1 に示す．*MG* は 8 個の等式の証明に成功し，*SG* は 14 個の等式の証明に成功した．このうち，*MG* と *SG* の両方で成功した等式は 3 個のみであった．*MSG* は *MG* で成功した 8 個の等式のうち 7 個について証明に成功した．*MSG+SG* は，*MSG* または *SG* で成功したすべての等式の証明に成功した．

SG で成功した例は全て NICE$_{SG}$ でも成功した．NICE$_{SG}$ で成功した例のうち，等式 4 については，NICE$_{SG}$ の証明出力をチェックしたところ，出力された証明に間違いがあり，この結果は NICE$_{\mathcal{G}}$ の実装上のバグによるものと考えられる．これ以外では，NICE$_{SG}$ で証明に成功した等式は *SG* より 4 個多かっ

表 1　補題自動生成法の比較 (○：定理，∞：発散，−： 失敗)

No	証明する等式	*RI*	*MG*	*SG*	*MSG*	*MSG*+*SG*	SPIKE	NICE$_{SG}$	NICE$_{TP}$
1	$x + \mathrm{s}(x) \doteq \mathrm{s}(x + x)$	∞	○	○	○	○	∞	○	○
2	$(x + x) + x \doteq x + (x + x)$	∞	∞	○	∞	○	∞	○	○
3	$(x + y) * z \doteq (x * z) + (y * z)$	∞	∞	∞	∞	∞	∞	○	○
4	$x * s(y) \doteq (x * y) + x$	∞	∞	∞	∞	∞	∞	○	−
5	$(x * y) * z \doteq x * (y * z)$	∞	∞	∞	∞	∞	∞	○	○
6	$\mathrm{rev}(\mathrm{rev}(xs)) \doteq xs$	∞	○	∞	○	○	∞	∞	○
7	$\mathrm{rev}(\mathrm{rev}(xs) \mathbin{@} [\,]) \doteq xs$	∞	○	∞	○	○	∞	∞	○
8	$\mathrm{rev}(\mathrm{rev}(xs \mathbin{@} [\,])) \doteq xs$	∞	○	∞	○	○	∞	∞	○
9	$\mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) \doteq xs \mathbin{@} ys$	∞	○	∞	○	○	∞	∞	○
10	$\mathrm{rev}(xs \mathbin{@} \mathrm{rev}(ys)) \doteq ys \mathbin{@} \mathrm{rev}(xs)$	∞	∞	∞	∞	∞	∞	∞	○
11	$\mathrm{qrev}(\mathrm{rev}(xs), ys) \doteq xs \mathbin{@} ys$	∞	○	∞	∞	∞	∞	∞	−
12	$\mathrm{qrev}(xs, [\,]) \doteq \mathrm{rev}(xs)$	∞	∞	○	∞	○	∞	○	○
13	$\mathrm{qrev}(xs \mathbin{@} [\,], [\,]) \doteq \mathrm{rev}(xs)$	∞	∞	○	∞	○	∞	○	○
14	$\mathrm{qrev}(xs, ys) \doteq \mathrm{rev}(xs) \mathbin{@} ys$	∞	∞	○	∞	○	∞	○	○
15	$\mathrm{rev}(xs \mathbin{@} ys) \doteq \mathrm{rev}(ys) \mathbin{@} \mathrm{rev}(xs)$	∞	∞	∞	∞	∞	∞	○	○
16	$\mathrm{rev}(xs \mathbin{@} xs) \doteq \mathrm{rev}(xs) \mathbin{@} \mathrm{rev}(xs)$	∞	∞	∞	∞	∞	∞	○	○
17	$\mathrm{rev}(\mathrm{qrev}(xs, ys)) \doteq \mathrm{rev}(ys) \mathbin{@} xs$	∞	∞	○	∞	○	∞	○	○
18	$\mathrm{qrev}(xs \mathbin{@} ys, [\,]) \doteq \mathrm{rev}(xs \mathbin{@} ys)$	∞	∞	○	∞	○	∞	○	○
19	$\mathrm{qrev}(xs, \mathrm{rev}(ys)) \doteq \mathrm{rev}(xs) \mathbin{@} \mathrm{rev}(ys)$	∞	∞	○	∞	○	∞	○	○
20	$\mathrm{qrev}(xs \mathbin{@} ys, [\,]) \doteq \mathrm{rev}(ys) \mathbin{@} \mathrm{rev}(xs)$	∞	∞	∞	∞	∞	∞	○	−
21	$\mathrm{qrev}(xs \mathbin{@} ys, zs) \doteq \mathrm{rev}(xs \mathbin{@} ys) \mathbin{@} zs$	∞	∞	○	∞	○	∞	○	○
22	$\mathrm{qrev}(\mathrm{qrev}(xs, ys), [\,]) \doteq \mathrm{rev}(ys) \mathbin{@} xs$	∞	∞	○	∞	○	∞	○	−
23	$\mathrm{qrev}(\mathrm{qrev}(xs, ys), zs) \doteq (\mathrm{rev}(ys) \mathbin{@} xs) \mathbin{@} zs$	∞	∞	○	∞	○	∞	○	−
24	$\mathrm{qrev}(\mathrm{qrev}(xs, ys), zs) \doteq \mathrm{rev}(ys) \mathbin{@} (xs \mathbin{@} zs)$	∞	∞	○	∞	○	∞	○	−
25	$\mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) \doteq \mathrm{rev}(\mathrm{rev}(xs)) \mathbin{@} ys$	∞	○	○	○	○	∞	∞	○
26	$\mathrm{rev}(\mathrm{rev}(xs \mathbin{@} ys)) \doteq xs \mathbin{@} \mathrm{rev}(\mathrm{rev}(ys))$	∞	○	○	○	○	∞	○	○
27	$\mathrm{rev}(\mathrm{rev}(xs)) \mathbin{@} [\,] \doteq xs$	∞	∞	∞	∞	∞	∞	∞	○
28	$\mathrm{qrev}(\mathrm{rev}(xs), [\,]) \doteq xs$	∞	∞	∞	∞	∞	∞	∞	−
29	$\mathrm{rev}(\mathrm{qrev}(xs, [\,])) \doteq xs$	∞	∞	∞	∞	∞	∞	∞	○
30	$\mathrm{qrev}(\mathrm{qrev}(xs, [\,]), [\,]) \doteq xs$	∞	∞	∞	∞	∞	∞	∞	−
31	$\mathrm{qrev}(\mathrm{qrev}(xs, [\,]), zs) \doteq xs \mathbin{@} zs$	∞	∞	∞	∞	∞	∞	∞	−
32	$\mathrm{rev}(\mathrm{rev}(xs) \mathbin{@} \mathrm{rev}(ys)) \doteq ys \mathbin{@} xs$	∞	∞	∞	∞	∞	∞	∞	−
33	$\mathrm{rev}(\mathrm{rev}(xs) \mathbin{@} ys) \doteq \mathrm{rev}(ys) \mathbin{@} xs$	∞	∞	∞	∞	∞	∞	○	○
	成功数	0	8	14	7	18	0	19	23

た．このような差があったのは，*Expand* 規則で展開する部分項について，NICE$_{SG}$ では我々のシステムより幅広い選択を行っていることが原因である．実際，我々のシステムでは，*Expand* 規則で展開する部分項を基本項に限っているが，これを擬基本項[2] に緩和したところ，NICE$_{SG}$ と *SG* は同じ等式について証明に成功した．従って，我々の健全一般化法の実装は，文献[16] とほぼ同等の補題生成能力を達成している．

一方，SPIKE はすべての帰納的定理の証明で発散した．また，SPIKE に備わっている発散鑑定法は補題生成に非常に時間がかかるため，実験時間内で補題生成を行うことが困難であった．なお，SPIKE のマニュアルに掲載されている補題生成例については補題生成が確認できた．

次に，帰納的定理の自動証明における標準的な例題集である Dream Corpus[†1] から例を抜粋し同様の実験を行った．Dream Corpus はエジンバラ大学の Bundy らのグループにより，Boyer-Moore[8] Corpus

†1 Dream corpus of induction conjectures
`http://dream.inf.ed.ac.uk/dc/lib.html`
`http://www.tptp.org/`

表 2 **Dream Corpus(抜粋) による補題自動生成法の比較 (○：定理，∞：発散，−：失敗)**

No	証明する等式	*RI*	*MG*	*SG*	*MSG*	*MSG*+*SG*	SPIKE	NICE_{SG}	NICE_{TP}
3	$x+(y+z) \doteq y+(x+z)$	−	−	−	−	−	○	○	−
4	$x+y \doteq y+x$	−	−	−	−	−	○	○	−
25	$\text{rev}(xs \mathbin{@} ys) \doteq \text{rev}(ys) \mathbin{@} \text{rev}(xs)$	∞	∞	∞	∞	∞	∞	○	○
27	$x*(y+z) \doteq (x*y)+(x*z)$	∞	∞	∞	∞	∞	∞	∞	−
28	$x*\text{s}(y) \doteq x+(x*y)$	−	−	−	−	−	∞	∞	−
29	$x*y \doteq y*x$	−	−	−	−	−	∞	∞	−
30	$x*(y*z) \doteq y*(x*z)$	−	−	−	−	−	∞	∞	−
31	$(x*y)*z \doteq x*(y*z)$	∞	∞	∞	∞	∞	∞	○	○
43	$\text{rev}(\text{rev}(x :: xs)) \doteq x :: xs$	∞	○	∞	○	○	∞	∞	○
47	$\text{len}(\text{rev}(xs)) \doteq \text{len}(xs)$	∞	○	∞	∞	∞	∞	∞	−
60	$\text{dbl}(i) \doteq \text{s}(\text{s}(0))*i$	∞	○	○	○	○	∞	○	○
63	$\text{last}(\text{rev}(x :: xs)) \doteq x :: [\,]$	∞	∞	∞	∞	∞	∞	∞	−
64	$\exp(i, j+k) \doteq \exp(i,j)*\exp(i,k)$	∞	∞	∞	∞	∞	∞	○	−
79	$\text{factloop}(j,i) \doteq i*\text{fact}(j)$	∞	∞	∞	∞	∞	∞	∞	−
80	$\text{fact2}(i) \doteq \text{fact}(i)$	∞	∞	∞	∞	∞	∞	∞	−
81	$\text{qrev}(xs, ys) \doteq \text{rev}(xs) \mathbin{@} ys$	∞	∞	○	∞	○	∞	○	○
82	$\text{qrev}(xs, [\,]) \doteq \text{rev}(xs)$	∞	∞	○	∞	○	∞	○	○
83	$\text{rev2}(xs \mathbin{@} ys) \doteq \text{rev2}(ys) \mathbin{@} \text{rev2}(xs)$	∞	∞	−	∞	−	∞	○	−
84	$\text{rev2}(\text{rev2}(x :: xs)) \doteq x :: xs$	∞	∞	∞	∞	∞	∞	∞	−
111	$(y+x)-x \doteq y$	∞	○	∞	∞	∞	∞	∞	−
113	$x*(y-z) \doteq (y*x)-(z*x)$	−	−	−	−	−	∞	∞	−
116	$\text{s}(x+y)-y \doteq \text{s}(x)$	∞	○	∞	∞	∞	∞	∞	−
158	$\text{timeslist}(xs@ys) \doteq \text{timeslist}(xs)*\text{timeslist}(ys)$	∞	∞	∞	∞	∞	∞	○	−
232	$\text{s}(\text{s}(0))*x \doteq x+x$	∞	∞	∞	∞	∞	∞	○	○
236	$\exp(i*j,k) \doteq \exp(i,k)*\exp(j,k)$	∞	∞	∞	∞	∞	∞	∞	−
237	$\exp(\exp(i,j),k) \doteq \exp(i, j*k)$	∞	∞	∞	∞	∞	∞	∞	−
270	$\text{s}^4(0)*x \doteq \text{s}^2(0)*(\text{s}^2(0)*x)$	∞	∞	∞	∞	∞	∞	○	○
300	$\text{h}(x,\text{h}(y,z)) \doteq \text{h}(y,\text{h}(x,z))$	−	−	−	−	−	○	○	−
318	$\text{fn2}(x,y) \doteq \text{fn2}(y,x)$	−	−	−	−	−	○	○	−
319	$\text{foldrfn}(xs,i) \doteq \text{foldlfn}(\text{rev}(xs),i)$	∞	−	∞	∞	∞	∞	○	−
370	$(x*x)-\text{s}(0) \doteq \text{s}(x)*\text{sub}(x)$	∞	∞	∞	∞	∞	∞	∞	−
375	$\text{sub}((x-\text{s}^2(0))*(x-\text{s}^2(0))) \doteq (x-\text{s}^3(0))*\text{sub}(x)$	∞	∞	∞	∞	∞	∞	○	−
694	$\text{timesfn}(i,j,\text{ans}) \doteq (i*j)+\text{ans}$	−	−	−	−	−	∞	○	−
1052	$(x+y)*z \doteq (x*z)+(y*z)$	∞	∞	∞	∞	∞	∞	○	○
	成功数/失敗数	0/9	5/10	3/10	2/9	4/10	4/0	18/0	9/25

(Dmacs Corpus) から抜粋・自動変換により得られた述語論理上の帰納的定理の例題集 (中に多数の重複を含む) である．そのうち，本論文で対象としている等式論理上の帰納的定理の例題は 69 個 (重複削除後) 含まれている．そのうち 35 個は *RI* で証明に成功した．失敗もしくは発散した 34 個の等式についての結果を表 2 に示す．表中の番号は Dream Corpus での例題番号を表わす．なお，これらの例題では，単相項書き換えシステムが用いられている．

MG は 5 個の等式の証明に成功し，*SG* は 3 個の等式の証明に成功した．*MSG* は *MG* で成功した 5 個の等式のうち 2 個について証明に成功した．*MG* と *SG* の両方で成功した等式は 1 個あった．*MSG*+*SG* は，*MSG* または *SG* で成功したすべての等式の証明に成功した．以上より，差は僅かとなったものの補題生成能力については，表 1 と同様の傾向が観察できた．

我々のシステムの失敗例は全て *Expand* 規則における等式の向き付け失敗によるものである．SPIKE, NICE は，簡約順序による向き付けに失敗した等式を取り扱うための拡張機能 [6][7] を備えているが，我々のシステムではその拡張機能が組み込まれていないためである．

表 3　補題自動生成法の反証能力の比較 (×：非定理，∞：発散，−：失敗)

No	証明する等式	*RI*	*MG*	*SG*	*MSG*	*MSG+SG*	SPIKE
1	$x+y=y$	×	−	×	×	×	×
2	$s(x+y)=x$	×	−	×	×	×	×
3	$s(x)+y=s(s(y)+x)$	−	−	−	−	−	×
4	$x+s(0)=x+x$	−	−	−	−	−	×
5	$x*(x+y)=0$	×	∞	×	×	×	×
6	$x*y+z=(x+y)*z$	×	∞	×	×	×	×
7	$y+x*y=x+s(y)$	×	∞	×	×	×	×
8	$y+x*y=y*(s(s(0))+x)$	−	−	−	−	−	×
9	$xs@ys=ys$	×	−	×	×	×	×
10	$x::(xs@ys)=x::(ys@xs)$	−	−	−	−	−	×
11	$xs@(ys@zs)=ys@(xs@zs)$	−	−	−	−	−	×
12	$xs@ys=ys@xs$	−	−	−	−	−	×
13	$\mathrm{rev}(xs@ys)=ys@xs$	×	−	×	×	×	×
14	$\mathrm{rev}(\mathrm{rev}(xs))=\mathrm{rev}(xs)$	×	−	×	×	×	×
15	$\mathrm{rev}(\mathrm{rev}(xs@ys))=ys@xs$	∞	−	∞	−	−	×
16	$\mathrm{rev}(\mathrm{rev}(xs@ys))=\mathrm{rev}(xs@ys)$	×	−	×	×	×	×
17	$\mathrm{rev}(\mathrm{rev}(xs@ys))=\mathrm{rev}(ys@xs)$	−	−	∞	−	∞	×
18	$\mathrm{rev}(\mathrm{rev}(xs))@ys=ys@xs$	−	−	−	−	−	×
19	$\mathrm{rev}(\mathrm{rev}(xs))@ys=xs@\mathrm{rev}(ys)$	−	−	−	−	−	×
20	$\mathrm{rev}(xs)@ys=\mathrm{rev}(xs)@\mathrm{rev}(ys)$	×	−	∞	×	∞	×
21	$\mathrm{qrev}(xs,xs)=\mathrm{rev}(xs)$	×	∞	×	×	×	×
22	$\mathrm{qrev}(\mathrm{rev}(\mathrm{rev}(xs)),xs)=\mathrm{qrev}(\mathrm{qrev}(xs,[\,]),xs)$	×	∞	×	×	×	×
23	$\mathrm{qrev}(xs,ys)=\mathrm{qrev}(xs,xs)$	−	−	−	−	−	×
24	$\mathrm{qrev}(xs,\mathrm{rev}(\mathrm{rev}(ys)))=\mathrm{qrev}(xs,\mathrm{rev}(ys))$	∞	−	∞	∞	∞	×
25	$\mathrm{qrev}(xs@ys,xs)=\mathrm{rev}(xs@(ys@xs))$	∞	∞	−	∞	−	×
26	$\mathrm{qrev}(ys@xs,xs)=\mathrm{rev}(xs@(ys@xs))$	×	∞	∞	×	∞	×
27	$\mathrm{qrev}(\mathrm{rev}(xs),\mathrm{rev}(xs))=\mathrm{qrev}(xs,xs)$	∞	∞	∞	∞	∞	×
	成功数/失敗数	13/10	0/19	11/10	13/11	11/11	27/0

SG で成功した例は全て NICE$_{SG}$ でも成功した．ただし，NICE$_{SG}$ の証明出力をチェックしたところ，等式 64, 158, 319, 375 については出力された証明に間違いがあり，この結果は NICE$_{\mathcal{G}}$ の実装上のバグによるものと考えられる．これ以外の差については，表 1 と時と同様，*Expand* 規則で展開する部分項について，NICE$_{SG}$ では我々のシステムより幅広い選択を行っていることが原因であった．

次に，システムの反証能力を比較するため，非定理 (等式 1–27) の反証を試みた．実験結果を表 3 に示す．NICE$_{SG}$ および NICE$_{TP}$ は反証機能をもつが，基底項上の等式の反証にしか適用できないため，記載していない．また，*MG* は補題生成機能が健全でないため，反証機能を無効にした．SPIKE は全ての反証に成功し，我々のシステムはいくつかの非定理で反証に失敗した．しかしながら，補題自動生成法と反証の成功・非成功の間に有意な考察は出来なかった．

SPIKE が全ての反証に成功したのは，SPIKE が反証についての完全性 (非定理の反証に必ず成功する性質) [6] [7] を持つためであると考えられる．これに対して，我々のシステムは反証についての完全性を備えていない．反証についての完全性には，簡約順序による向き付けに失敗した等式を取り扱うための拡張機能が必要であるが，我々のシステムはそれを実装していないためである．

最後に，補題生成法の導入による実行時間のオーバーヘッドを確認するため，実行時間の比較を行った．結果を表 4 に示す．まず，Dream Corpus の例のうち，*RI* で成功する 35 例を実行した場合の合計時間を計測した．*RI* で成功するこれらの例において補題生成法によるオーバーヘッドはなかった．次に，表 1, 2 の例のうち，*MG*,*SG*,*MSG*,*MSG+SG* の 4 つ全

表 4 補題自動生成法のオーバーヘッドの比較 (単位： msec.)

証明する等式	*RI*	*MG*	*SG*	*MSG*	*MSG+SG*
Dream Corpus *RI* 成功 35 例	2613	2614	2557	2615	2557
表 1 − 1	-	143	57	142	57
表 1 − 25	-	174	257	174	278
表 1 − 26	-	207	116	207	116
表 2 − 60	-	180	92	178	92
表 2 失敗 9 例	436	435	436	436	433
表 3 成功 11 例	3737	-	1427	3760	1427
表 3 − 22	2535	-	227	2557	229
表 3 失敗 9 例	382	-	380	379	380

てで成功した例についてそれぞれの実行時間を計測した．生成される補題によって証明過程が異なるため，実行時間にばらつきがあった．また，*MSG* は *MG* と比較して同等のオーバーヘッドしかなかった．次に，表 2 に掲げた例のうち，*RI*,*SG*,*MSG*,*MSG+SG* の 4 つ全てで証明に失敗した 9 例について実行時間の合計を示す．これらの例で実行時間に差はなかった．次に，表 3 に掲げた例のうち，*RI*,*SG*,*MSG*,*MSG+SG* の 4 つ全てで反証に成功した 12 例について実行時間の合計を示す．このうち 11 例については，実行時間はほぼ同じであった．実行時間が大きく違った例 22 についての実行時間を表に示した．補題生成によって効率的に反証に成功する場合があることがわかる．最後に，*RI*,*SG*,*MSG*,*MSG+SG* の 4 つ全てで反証に失敗した 9 例について実行時間の合計を示す．これらの例については実行時間に差は見られなかった．

以上の考察から，単相書き換えシステムに対しては，*MSG* は，*MG* と類似の例に対して有効性を持ち，また，*SG* と異なった有効範囲をもつ．さらに，*SG* と *MSG* を組み合わせた *MSG+SG* は，*SG* と *MSG* の両者の有効範囲を兼ね備えている．このため，従来の健全一般化法 *SG* に比べて，*MSG+SG* は，より強力な健全補題自動生成法を実現している．

我々のシステムは反証に失敗する場合があるが，SPIKE のように反証についての完全性を実現すれば，補題生成は反証の成否には影響しないと考えられる．

補題生成の成功によって証明経過が変更される要因以外では，補題生成法によるオーバーヘッドは証明にかかる時間に比べ非常に小さい．また，*MSG* は *MG* と比較して同等のオーバーヘッドしかない．

6 おわりに

本論文では，書き換え帰納法の補題自動生成法である発散鑑定法から，その手続の一部である等式一般化操作を健全一般化法に置き換えることによって得られる健全発散鑑定法という新しい補題自動生成法を提案した．そして，健全発散鑑定法が，健全一般化法と同じく，単相項書き換えシステムに対して，(発散鑑定法には欠けていた) 健全性が保証されていることを示した．

さらに，これらの補題生成法を反証機能付き書き換え帰納法上に実装し，補題生成能力の比較実験を行った．我々の提案する健全発散鑑定法が，単相項書き換えシステムについて，従来の発散鑑定法と類似の対象に対しての有効性をもつことおよび健全一般化法と異なる有効範囲を有していることを実験により確認した．また，健全発散鑑定法が発散鑑定法と比較して同等のオーバーヘッドしかないこと，補題生成の成功によって証明経過が変更されるような場合以外では，補題生成法によるオーバーヘッドは証明にかかる時間に比べ非常に小さいことも確認した．

以上の結果により，従来知られていた健全一般化法に加えて，我々の提案する健全発散鑑定法を組み合わせることにより，効率的でより強力な健全自動補題生成法が実現できる．健全な補題自動生成法は，これまで健全一般化法しか知られておらず， 本論文の結果は，効率的な反証機能付き自動証明システムを見通しよく実現する基礎技術として有効であると考える．

なお，本論文では，補題自動生成のための変形操作として書き換え規則変形操作 M を用いた．しかし，

文献[17] の発散鑑定法では，他にもアキュムレータを用いた規則における変形操作も提案されている．また，文献[9] には，より高度なリップリング法が提案されている．これらを参考にして，より強力かつ健全な補題自動生成法を提案することは今後の検討課題である．

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 17700002, 19500003 の補助を受けて行われた．

参 考 文 献

[1] Aoto, T.: Dealing with non-orientable equations in rewriting induction, *Proc.17th RTA*, LNCS, Vol. 4098, 2006, pp. 242–256.

[2] Aoto, T.: Designing a rewriting induction prover with an increased capability of non-orientable equations, *Proc. of SCSS*, RISC Technical Report, Vol. 08-08, 2008, pp. 1–15.

[3] Aoto, T.: Soundness of rewriting induction based on an abstract principle, *IPSJ Trans.Prog.*, Vol. 49,No. SIG 1 (PRO 35)(2008), pp. 28–38.

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

[5] Basin, D. and Walsh, T.: Difference matching, *Proc.11th CADE*, LNCS, Vol. 607, 1992, pp. 295–309.

[6] Bouhoula, A.: Automated theorem proving by test set induction, *J.Symbolic Comput.*, Vol. 23(1997), pp. 47–77.

[7] Bouhoula, A., Kounalis, E., and Rusinowitch, M.: Automated mathematical induction, *J.Logic and Comput.*, Vol. 5,No. 5(1995), pp. 631–668.

[8] Boyer, R. S. and Moore, J. S.: *A Computational Logic*, Academic Press, New York, 1979.

[9] Bundy, A., Basin, D., Hutter, D., and Ireland, A.: *Rippling: Meta-Level Guidance for Mathematical Reasoning*, Cambridge University Press, Cambridge, 2005.

[10] Huet, G. and Oppen, D. C.: Equations and rewrite rules: a survey, Technical report, Stanford University, Stanford, CA, USA, 1980.

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

[12] 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.

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

[14] Toyama, Y.: How to prove equivalence of term rewriting systems without induction, *Theor.Comput.Sci.*, Vol. 90,No. 2(1991), pp. 369–390.

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

[16] Urso, P. and Kounalis, E.: Sound generalizations in mathematical induction, *Theor.Comput.Sci.*, Vol. 323(2004), pp. 443–471.

[17] Walsh, T.: A divergence critic for inductive proof, *J.Artif.Intell.Res.*, Vol. 4(1996), pp. 209–235.