停止性検証器を利用した書き換え帰納法手続き

Rewriting Induction Using Termination Checkers

青戸等人

Takahito Aoto

東北大学 電気通信研究所

RIEC, Tohoku University

aoto@nue.riec.tohoku.ac.jp

項書き換えシステムの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．既存の書き換え帰納法に基づく証明器では，利用する簡約順序を手続き開始時にユーザーが与える必要がある．しかしながら，証明手続きの開始時点で適切な簡約順序を決定することは困難なことも多い．本研究では，簡約順序を入力として与えず，停止性検証器を利用して，適切な簡約順序の存在を途中で検証しながら証明を行なう書き換え帰納法手続きを提案する．

1 はじめに

プログラムの性質のうち，自然数やリストといったデータ構造に関する帰納法で示されるような性質をプログラムの帰納的な性質とよぶ．項書き換えシステムの帰納的定理の自動証明法として書き換え帰納法 [13] が知られている．

書き換え帰納法の原理は，簡約順序に基づくネーター帰納法であり，与えられた簡約順序の元で推論規則を適用することにより導出を行う．従って，用いる簡約順序により異なる導出列が得られ，一般的には，どのような簡約順序に基づくかにより証明の成否は変化する．

既存の書き換え帰納法に基づく証明器SPIKE[4, 5, 6] や NICE[15] などでは，利用する簡約順序を手続き開始前にユーザーが指定するようになっている．しかしながら，書き換え帰納法では，証明途中に生成される等式についても順序付けが必要となるため，証明手続きの開始時点で適切な簡約順序を決定することは一般には困難である．

一方，近年，強力な停止性の検証理論 [2, 11] や効率的な自動検証技術 [8, 10, 12] により，効率的な停止性検証が利用できるようになってきている．このため，従来，簡約順序を手続き開始時にユーザーが与えるのが一般的であった完備化手続きにおいても，利用する簡約順序を手続きの入力として固定するのではなく，適切な簡約順序の存在性を停止性検証器を利用して保証しながら，完備化を行う手続きが提案されている [14, 17].

本論文では，書き換え帰納法において，簡約順序を入力として与えず，停止性検証器を利用し，適切な簡約順序の存在を途中で検証しながら証明を行なう手続きを提案する．

2 書き換え帰納法

書き換え規則の集合のことを項書き換えシステムとよぶ．$\mathcal{R}$ として自然数の加算を行う書き換えシステムを考える．

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

ここで，自然数 $0,1,2,\ldots$ は，$0,\mathsf{s}(0),\mathsf{s}(\mathsf{s}(0)),\ldots$ と表現されている．自然数上の加算についての結合律は

$$\mathsf{plus}(\mathsf{plus}(x,y),z)=\mathsf{plus}(x,\mathsf{plus}(y,z)) \qquad (1)$$

と表現され，$\mathcal{R}$ の帰納的定理となる．以下では，書き換え帰納法の推論規則について簡単に説明した後，書き換え帰納法に基づく等式 (1) の証明を示す．

書き換え帰納法で重要な役割を果たすのは，帰納法の適用に対応する Expand 規則である．この推論規則では，等式の両辺のうち，与えられた簡約順序でより大きい方を $\mathcal{R}$ の書き換え規則に合わせて展開するとともに簡約を 1 ステップ行う．一方で，展開の元となった等式は書き換え規則として保存される．書き換え帰納法の残りの規則は Simplify 規則と Delete 規則であり，これらは，等式の両辺を簡約する規則

Simplify

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

Delete

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

Expand

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

図 1: 書き換え帰納法の推論規則

と，自明な等式 (両辺が等しい等式) を削除する規則である (図 1 参照)[1].

推論は等式集合 E (証明すべき等式の集合) と項書き換え規則集合 H (帰納法の仮定や証明済みの規則の集合) の対を対象として行われる．次は E と H の初期状態である．

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

$$H\left\{\right.$$

Expand 規則の適用により，

$$E\left\{\begin{array}{l}\mathsf{plus}(y_0,z) = \mathsf{plus}(\mathsf{0},\mathsf{plus}(y_0,z))\\ \mathsf{plus}(\mathsf{s}(\mathsf{plus}(x_1,y_1)),z) = \mathsf{plus}(\mathsf{s}(x_1),\mathsf{plus}(y_1,z))\end{array}\right.$$

$$H\left\{\ \mathsf{plus}(\mathsf{plus}(x,y),z) \to \mathsf{plus}(x,\mathsf{plus}(y,z))\right.$$

Simplify 規則の適用により，

$$E\left\{\begin{array}{l}\mathsf{plus}(y_0,z) = \mathsf{plus}(y_0,z)\\ \mathsf{s}(\mathsf{plus}(x_1,\mathsf{plus}(y_1,z)) = \mathsf{s}(\mathsf{plus}(x_1,\mathsf{plus}(y_1,z))\end{array}\right.$$

$$H\left\{\ \mathsf{plus}(\mathsf{plus}(x,y),z) \to \mathsf{plus}(x,\mathsf{plus}(y,z))\right.$$

Delete 規則の適用により，

$$E\left\{\right.$$

$$H\left\{\ \mathsf{plus}(\mathsf{plus}(x,y),z) \to \mathsf{plus}(x,\mathsf{plus}(y,z))\right.$$

なる導出列が得られる．$E = \emptyset$ となったとき，書き換え帰納法の手続きは成功し，導出が終了する．

推論規則による導出関係を $\rightsquigarrow$ と書き，その反射推移閉包を $\overset{*}{\rightsquigarrow}$ と書く．このとき，以下のように書き換え帰納法による帰納的定理証明の健全性が成立する．

命題 1 ([1, 13]) $\mathcal{R}$ を擬簡約な構成子システム，E を等式集合，$>$ を $\mathcal{R} \subseteq >$ なる簡約順序とする．ある項書き換え規則集合 H が存在して，図 1 の推論規則により $\langle E, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ となるならば，E 中の等式は $\mathcal{R}$ の帰納的定理である．

3 停止性検証を利用した書き換え帰納法

図 1 の推論規則や命題 1 に見られるように，書き換え帰納法の推論規則は，入力として与えられた簡約順序 $>$ の基づいて定められている．簡約順序 $>$ は，Expand 規則で等式を展開する毎に，どちらの等式のどちらの辺を展開するか (もしくは，展開できるか) を決定するために用いられる．このため，簡約順序 $>$ が異なれば，一般に導出は変わってくる．

例 2 以下の項書き換えシステム $\mathcal{R}$ と等式集合 E を考える．

$$\mathcal{R}\left\{\begin{array}{lcl}\mathsf{plus}(\mathsf{0},y) & \to & y\\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y))\\ \mathsf{double}(\mathsf{0}) & \to & \mathsf{0}\\ \mathsf{double}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))\end{array}\right.$$

$$E\left\{\ \mathsf{double}(\mathsf{plus}(x,y)) = \mathsf{plus}(\mathsf{double}(x),\mathsf{double}(y))\right.$$

このとき，$\mathcal{R}\cup E \subseteq >$ となる簡約順序も $\mathcal{R}\cup E^{-1} \subseteq >$ となる簡約順序も存在する．

従って，簡約順序を予め固定せずに，簡約順序の存在性を検証しながら書き換え帰納法の推論を進める方がより柔軟な探索が可能となり，証明成功に至る導出列を発見できる可能性が高まると考えられる．

簡約順序とは，代入および文脈に閉じた整礎な半順序のことをいう．以下のよく知られた事実を用い

Simplify

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

Delete

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

Expand

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

図 2: 停止性検証を利用した書き換え帰納法の推論規則

て，簡約順序の存在性を停止性の検証と結びつけることが出来る．

命題 3 ([3]) $\mathcal{R}$ が停止性を持つとき，$\overset{+}{\to}_{\mathcal{R}}$ は簡約順序である．

項書き換えシステム $\mathcal{R}$ が停止性を持つことを $\mathrm{SN}(\mathcal{R})$ と記す．図 2 に，停止性検証を利用した書き換え帰納法の推論規則を記す．以下のように停止性検証を利用した書き換え帰納法の健全性が成立する．

定理 4 $\mathcal{R}$ を擬簡約な構成子システム，E を等式集合，ある項書き換え規則集合 H が存在して，図 2 の推論規則により $\langle E, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ となるならば，E 中の等式は $\mathcal{R}$ の帰納的定理である．

証明: $\langle E, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ より，$\mathrm{SN}(\mathcal{R}\cup H)$ が成立しており，命題 3 より，$\overset{+}{\to}_{\mathcal{R}\cup H}$ は簡約順序となる．$>$ を $\overset{+}{\to}_{\mathcal{R}\cup H}$ とおく．このとき，$\overset{+}{\to}_{\mathcal{R}\cup H}$ の定義より，$\mathcal{R} \subseteq >$ かつ任意の $l \to r \in H$ について $l > r$ が成立する．従って，図 2 の推論規則による導出 $\langle E, \emptyset\rangle \overset{*}{\rightsquigarrow} \langle \emptyset, H\rangle$ は，図 1 の推論規則による導出と見做すことが出来る．よって，命題 1 より題意が導かれる． □

4 実装および実験

停止性検証を利用した書き換え帰納法を関数型プログラム言語 SML/NJ を用いて実装した．停止性の検証には，辞書式経路順序に基づく停止性検証プログラムを利用した．Expand 規則による等式の展開は向きや項の位置の選択に任意性があるため，失敗や発散を検出した場合にバックトラックを行う．発散の検出はヒューリスティックスに基づく．

以上の実装を用い，いくつかの標準的な例について実験を試みたところ，帰納的定理証明に成功した (表 1)．しかしながら，我々のシステムは，基本的な書き換え帰納法に基づいているため，強力な帰納的定理証明システムとなってはいない．

5 おわりに

本論文では，停止性検証を利用した書き換え帰納法を提案し，その健全性を示した．また，停止性検証器を利用した書き換え帰納法手続きを実装し，いくつかの標準的な例について帰納的定理証明に成功することを確認した．

停止性検証を利用した書き換え帰納法では，簡約順序を固定としないため，従来法と比べて等式の向きの選択に任意性が増える．これにより，より柔軟な探索が可能となり，証明成功に至る導出列を発見できる可能性が高まると考えられる．

また，依存対手法 [2, 11] などのより強力な停止性検証理論に基づく停止性検証を用いたり，より強力な書き換え帰納法 [1, 5, 9] や補題発見機構 [7, 15, 16] などを組み込み，より強力な帰納的定理証明システムを実現することは今後の課題である．

参考文献

[1] T. Aoto. Dealing with non-orientable equations in rewriting induction. In *Proc. of RTA 2006*, Vol. 4098 of *LNCS*, pp. 242–256. Springer-Verlag, 2006.

[2] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

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

	$\mathcal{R}$	E
(1)	$\left\{\begin{array}{lcl}\mathsf{plus}(0,y) & \to & y\\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y))\end{array}\right.$	$\{\ \mathsf{plus}(x,\mathsf{s}(y)) = \mathsf{s}(\mathsf{plus}(x,y))$
(2)	$\left\{\begin{array}{lcl}\mathsf{plus}(0,y) & \to & y\\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y))\end{array}\right.$	$\{\ \mathsf{plus}(\mathsf{plus}(x,y),z) = \mathsf{plus}(x,\mathsf{plus}(y,z))$
(3)	$\left\{\begin{array}{lcl}\mathsf{diff}(0,x) & \to & 0\\ \mathsf{diff}(\mathsf{s}(x),0) & \to & \mathsf{s}(x)\\ \mathsf{diff}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{diff}(x,y)\end{array}\right.$	$\{\ \mathsf{diff}(x,x) = 0$
(4)	$\left\{\begin{array}{lcl}\mathsf{app}(\mathsf{cons}(x,xs),ys) & \to & \mathsf{cons}(x,\mathsf{app}(xs,ys))\\ \mathsf{app}(\mathsf{nil},ys) & \to & ys\end{array}\right.$	$\{\ \mathsf{app}(\mathsf{app}(xs,ys),zs) = \mathsf{app}(xs,\mathsf{app}(ys,zs))$
(5)	$\left\{\begin{array}{lcl}\mathsf{double}(0) & \to & 0\\ \mathsf{double}(\mathsf{s}(x)) & \to & \mathsf{s}(\mathsf{s}(\mathsf{double}(x)))\\ \mathsf{half}(0) & \to & 0\\ \mathsf{half}(\mathsf{s}(0)) & \to & 0\\ \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) & \to & \mathsf{s}(\mathsf{half}(x))\end{array}\right.$	$\{\ \mathsf{half}(\mathsf{double}(x)) = x$
(6)	$\left\{\begin{array}{lcl}\mathsf{diff}(0,x) & \to & 0\\ \mathsf{diff}(\mathsf{s}(x),0) & \to & \mathsf{s}(x)\\ \mathsf{diff}(\mathsf{s}(x),\mathsf{s}(y)) & \to & \mathsf{diff}(x,y)\\ \mathsf{sub}(0) & \to & 0\\ \mathsf{sub}(\mathsf{s}(x)) & \to & x\end{array}\right.$	$\{\ \mathsf{diff}(x,\mathsf{s}(0)) = \mathsf{sub}(x)$
(7)	$\left\{\begin{array}{lcl}\mathsf{plus}(0,y) & \to & y\\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y))\\ \mathsf{app}(\mathsf{cons}(x,xs),ys) & \to & \mathsf{cons}(x,\mathsf{app}(xs,ys))\\ \mathsf{app}(\mathsf{nil},ys) & \to & ys\\ \mathsf{len}(\mathsf{cons}(x,xs)) & \to & \mathsf{s}(\mathsf{len}(xs))\\ \mathsf{len}(\mathsf{nil}) & \to & 0\end{array}\right.$	$\{\ \mathsf{len}(\mathsf{app}(xs,ys)) = \mathsf{plus}(\mathsf{len}(xs),\mathsf{len}(ys))$

表 1: 停止性検証を利用した書き換え帰納法の実験例

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

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

[6] A. Bouhoula and M. Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14:189–235, 1995.

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

[8] M. Codish, V. Lagoon, and P. J. Stuckey. Solving partial order constraints for LPO termination. In *Proc. of RTA 2006*, Vol. 4098 of *LNCS*, pp. 4–18. Springer-Verlag, 2006.

[9] N. Dershowitz and U. S. Reddy. Deductive and inductive synthesis of equational programs. *Journal of Symbolic Computation*, 15:467–494, 1993.

[10] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. of IJCAR 2006*, Vol. 4130 of *LNAI*, pp. 281–286. Springer-Verlag, 2006.

[11] N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proc. of RTA 2004*, Vol. 3091 of *LNCS*, pp. 249–268. Springer-Verlag, 2004.

[12] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205(4):474–511, 2007.

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

[14] 瀬古寛幸, 青戸等人, 外山芳人. 自動順序付けに基づく完備化手続き. 電気関係学会東北支部連合大会, 平成 18 年度 1E-9, p. 116, 2006.

[15] P. Urso and E. Kounalis. Sound generalizations in mathematical induction. *Theoretical Computer Science*, 323:443–471, 2004.

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

[17] I. Wehrman, A. Stump, and E. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *Proc. of RTA 2006*, Vol. 4098 of *LNCS*, pp. 287–296. Springer-Verlag, 2006.