平成 14 年度 石田 (實) 記念財団研究発表会資料

高階書き換えシステムに基づくプログラム検証法

外山 芳人

東北大学 電気通信研究所

〒 980-8577 仙台市 青葉区 片平 2-1-1

1 はじめに

ソフトウェアに対する近年の多様な要求に答える形で，関数型言語や論理型言語などの新しい言語が提案されてきた．しかし，これらの新しい言語がその利点を十分に発揮するためには，言語に適したソフトウェアの理論とそれに基づいたプログラム検証支援技術の確立が不可欠である．とりわけ，計算機科学の多くの分野で近年広く利用されている等式推論による検証支援技術は今後ますます重要になると思われる．

等式推論の基本は，あらかじめ定められた等式（公理）の集合をもちいて，与えられた等式 $A = B$ が成立するか否かを決定することである．よく知られているように，ふつうの等式推論では“等しいもの”をつぎつぎに置き換えることによって左辺 A と右辺 B がつながるか否かを調べ，等式 $A = B$ が成立するか否かを決定する．しかし，このような素朴な等式推論法はきわめて効率が悪い．そこで，等式（公理）を複雑な式から単純な式への非可逆な一方向の書き換え規則としてもちいる推論法が提案された．たとえば，等式 $1+2=3$ は $1+2$ の計算結果として 3 が得られるのであって，その逆ではないとみなすわけである．この方向付けられた書き換えを $1+2$ から 3 へのリダクションとよび $1+2 \to 3$ で表わす．このように，等式推論をリダクションによる計算とみなすことによって得られる計算・証明システムが項書き換えシステムである．項書き換えシステムは，等式にもとづく柔軟な計算法と効率的な証明法を提供できるため，定理自動証明，関数型あるいは論理型言語，代数的仕様記述，記号処理，プログラムの検証，合成，変換など，さまざまな分野で現在広くもちいられている [2]．さらに，高階の概念を持つ 関数型言語の計算モデルとして，λ-記法等を導入して項の中に高階の変数を許した高階の項書き換えシステムも提案されている [12]．

ソフトウェアの新しい検証支援技術の形式的な基礎を与える目的で，我々はこれまで 1 階の書き換えシステムを基礎とした関数型プログラムの検証に関する研究を進めてきた．これらの成果を高階の書き換えシステムに拡張することで，高機能な関数型言語に適した検証技法を確立することが本研究の目的である．本研究では，書き換えシステムに基づくプログラム検証法を拡張して，高階のシステムを含む一般的な関数型プログラムに適用するための基礎検討を行なう．

2 研究内容

本研究では，プログラム検証で重要な役割をはたすプログラムの帰納的性質の自動検証手法についてまず検討する．我々は，リダクションシステムの強正規性と退行性に基づいて，書き換え帰納法の抽象的な枠組みをすでに提案している [16]．この枠組みは，プログラムの帰納的性質の解析手法として非常に強力であり，高階書き換えシステムに対しても容易に適用可能である．しかし，書き換え帰納法に基づく検証手法をプログラムに効果的に適用するためには，帰納的性質が成立するか否かが判定可能なプログラムのクラスを理論的に明らかにすることが必要である．ここでは，帰納的性質の検証問題を，抽象的なリダクションシステムの等価性判定問題

に形式化することにより，簡明で見通しの良い理論的枠組みを与える．ここで得られた決定可能条件は，従来知られていた条件の一般化になっており，高階書き換えシステムを含む，より広いクラスの帰納的性質の決定問題に対して適用可能である．

次に，定理自動証明によるプログラム自動変換法の基礎研究として，完備化手続きに基づくプログラムの融合変換法について検討する．プログラムの融合変換と書き換えシステムの完備化は，それぞれ異なる目的と背景をもつ手続きである．ところが，我々の研究によって，関数型プログラムを書き換えシステムとしてモデル化し，関数の重み付けを工夫すると，完備化手続きによってプログラムの融合変換のいくつかは実現できることがすでに明らかになっている [7]．しかし，融合変換における変換手続きの停止性は完備化では保証されておらず，両者の適用範囲や変換能力の関係については明らかになっていない．ここでは，プログラム変換の手法である融合変換と，定理自動証明の手法である完備化の特長を組み合わせた新しいプログラム変換の実験システムを計算機上に実装し，我々の提案したプログラム変換法の有効性を実験で確認する．

最後に，プログラムの変換パターンに基づく高階プログラム変換法について検討する．プログラムの変換パターンをデータベースとして用意しておき，与えられたプログラムが変換パターンに合致したら変換規則を適用し，より効率の良いプログラムに変換する手法は，関数型プログラムに基づくソフトウェアの開発で有効であると考えられる．しかし，従来の手法 [6, 3, 13] は，２階の変換パターンに基づいているため，高階の関数型プログラムの変換には適用することができなかった．我々は，３階の変換パターンによるプログラム変換法を提案し，変換パターンに基づく高階プログラム変換法の検討を行う [18]．

3　帰納的定理の決定手続き

等式論理において等式が帰納的に成立するか否かは一般に決定不能である．最近 Kapur ら [9, 5] は，項書き換えシステムの枠組みをもちいて，等式の中に出現する定義関数記号を被覆集合帰納法によって消去できる条件を明らかにし，帰納的定理の決定可能なクラスを特徴づけた．また，福井ら [4] は，被覆集合帰納法の帰納図式を一般化して相互再帰関数も取り扱えるようにすることで，Kapur らの決定可能なクラスの拡張を試みている．

しかし，これらの方法は被覆集合帰納法を基礎としているため，書き換え規則や対象とする等式に対して強い構文的制限が必要となる．さらに，議論の枠組となる項書き換えシステムも，合流性，停止性，十分完全性をみたす構成子システムに限定されている．したがって，項書き換えシステムに停止性を仮定しておきながら，帰納的定理の判定には帰納図式をもちいるなど，理論的枠組みは複雑で見通しの悪いものとなっている．

本報告では，被覆集合帰納法のかわりに書き換え帰納法 [16, 14] を基礎とすることで，等式の決定可能な条件を抽象的なリダクションシステムの等価性判定問題に形式化し，簡明で見通しの良い理論的枠組みを与える．ここで与えられた決定可能条件は，Kapur ら [9, 5] の条件の一般化となっており，より広いクラスの決定問題

に対して適用可能である.

3.1 抽象リダクションシステム

抽象リダクションシステム $R = \langle A, \rightarrow \rangle$ は，対象集合 A と A 上の二項関係 $\rightarrow$ の対で定められる [2]. $\rightarrow$ をリダクション関係と呼ぶ. R のリダクション系列とは $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots$ のことである. A 上の同一関係を $\equiv$ で表す. $\overset{*}{\rightarrow}$, $\overset{*}{\leftrightarrow}$ をそれぞれ $\rightarrow$ の反射・推移閉包，$\rightarrow$ から生成される同値関係とする. $x \rightarrow y$ なる $y \in A$ が存在しないとき，$x \in A$ は正規形であるという. $x \overset{*}{\rightarrow} y$ となる正規形 $y \in A$ が存在するとき，$x \in A$ は正規形 y をもつという. NF で正規形の集合を表す.

$R = \langle A, \rightarrow \rangle$ におけるすべてのリダクション系列が有限であるとき，R は強正規性あるいは停止性をもつといい，$\rightarrow$: SN と記す. R は次の条件を満たすとき，合流性あるいはチャーチ・ロッサ性をもつといい $\rightarrow$: CR と記す.

$\forall x, y, z[x \overset{*}{\rightarrow} y \wedge x \overset{*}{\rightarrow} z \Rightarrow \exists w.\ y \overset{*}{\rightarrow} w \wedge z \overset{*}{\rightarrow} w]$

3.2 書き換え帰納法による等価条件

ここでは，書き換え帰納法をもちいて 2 つの抽象リダクションシステムの等価性を判定する手法を示す. 次節で説明するように，等式論理における帰納的定理の判定問題は，等価性判定問題の特殊な形として取り扱うことが可能である.

抽象リダクションシステム $R_1 = \langle A, \rightarrow_1 \rangle$ と $R_2 = \langle A, \rightarrow_2 \rangle$ は同じ対象集合 A をもつものとする. それぞれのシステムによって定められる同値関係を $\overset{*}{\leftrightarrow}_i$, 正規形の集合を $NF_i (i = 1, 2)$ で表す. ここで，$\rightarrow_i$ や $\overset{*}{\leftrightarrow}_i$ は $A \times A$ の部分集合であることに注意する. たとえば，$\rightarrow_1 \ \subseteq \ \rightarrow_2$ は $\forall x, y \in A[x \rightarrow_1 y \Rightarrow x \rightarrow_2 y]$ を意味する.

補題 1 R_1 と R_2 が次の条件をみたすものと仮定する.

(1) $\rightarrow_1 \ \subseteq \ \rightarrow_2$

(2) $\rightarrow_2$: SN

(3) $\rightarrow_1$: CR

このとき，以下が成立する.

(A) $\forall x, y[x \rightarrow_2 y \Rightarrow \exists z \in NF_2.$
$x \rightarrow_1 \overset{*}{\rightarrow}_2 z \overset{*}{\leftarrow}_2 y] \ \Rightarrow \overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$

(B) $\exists x, y[x \rightarrow_2 y \wedge \exists z, w \in NF_2.$
$x \rightarrow_1 \overset{*}{\rightarrow}_2 z \neq w \overset{*}{\leftarrow}_2 y] \ \Rightarrow \overset{*}{\leftrightarrow}_1 \neq \overset{*}{\leftrightarrow}_2$

(C) $\exists x, y[x \rightarrow_2 y \wedge x \in NF_1] \ \Rightarrow \overset{*}{\leftrightarrow}_1 \neq \overset{*}{\leftrightarrow}_2$

注意 補題1の条件(1)(2)(3)のもとで，$\overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$ の必要十分条件は $\rightarrow_2$: CR であることに注意する．

リダクションシステム R_1 と R_2 が補題1の条件 (1) (2) (3) をみたすものとする．このとき，同値関係 $\leftrightarrow_1$ と $\overset{*}{\leftrightarrow}_2$ が等価であるか否かの判定は，補題1を利用した以下の手続きで実現できる．

等価性判定手続き

(1) $x \rightarrow_2 y \in \rightarrow_2 - \rightarrow_1$ なる x と y をとる．

(2) $x \in NF_1$ ならば補題1 (C) より $\overset{*}{\leftrightarrow}_1 \neq \overset{*}{\leftrightarrow}_2$ となる．

(3) $x \notin NF_1$ ならば $x \rightarrow_1 u$ なる u が存在する．ここで，R_2 は強正規なので，R_2 に対する u と y の正規形 z と w は必ず求まる．このとき，$z \equiv w$ ならば，補題1 (A) より $\overset{*}{\leftrightarrow}_1 = \overset{*}{\leftrightarrow}_2$ となる．一方，$z \not\equiv w$ ならば，補題1 (B) より $\overset{*}{\leftrightarrow}_1 \neq \overset{*}{\leftrightarrow}_2$ となる．

3.3 項書き換えシステム

関数記号の有限集合 F，変数記号の集合 V から生成される項の集合を $T(F,V)$ と記す．変数を含まない項を基底項と呼びその集合を $T(F)$ と記す．D と C はそれぞれ定義関数記号と構成子記号の集合であり $F = D \cup C$ かつ $D \cap C = \phi$ とする．

項書き換えシステム R は有限個の書き換え規則 $l \rightarrow r$ で定義される項の集合上のリダクションシステムである [2]．等式 $s = t$ が項書き換えシステム R における帰納的定理であるとは，s と t に対する任意の基底代入 $\theta_g : V \rightarrow T(F)$ に対して $s\theta_g \overset{*}{\leftrightarrow} t\theta_g$ となることである．

例1 次の項書換えシステム R を考える．

$$R: \begin{cases} x + 0 \rightarrow x \\ x + s(y) \rightarrow s(x + y) \end{cases}$$

このとき，$0 + x \overset{*}{\leftrightarrow} x$ は成立しない．しかし，$\forall\theta_g\ [(0 + x)\theta_g \overset{*}{\leftrightarrow} x\theta_g]$ は成立するので，等式 $0 + x = x$ は R の帰納的定理となる．□

項書き換えシステム R が構成子システムであるとは，書き換え規則の左辺 $f(t_1, \cdots, t_n)$ が $f \in D$, $t_i \in T(C,V)$ $(i = 1, \ldots, n)$ となることである．R が十分完全とは $\forall s \in T(F) \exists t \in T(C)[s \overset{*}{\rightarrow} t]$ が成立することである．構成子システムの十分完全性に関しては，以下の命題が知られている．

命題1 [8] 項書き換えシステム R は合流性と停止性をみたす構成子システムとする．このとき，R の十分完全性は決定可能である．

代入の有限集合 $\{\sigma_i\}_i$ が項 $s \in T(F,V)$ の被覆代入集合とは $\sigma_i : V \to T(C,V)$ かつ $\forall\theta_g, \exists\sigma_i, \exists\theta'_g\ [s\theta_g \xrightarrow{*} s\sigma_i\theta'_g]$ が成立することである．

以下では，項書き換えシステム R を**合流性，停止性，十分完全性をみたす構成子システム**と仮定する．R が合流性と停止性をみたすなら，等式 $s = t$ が R の定理 (つまり $s \overset{*}{\leftrightarrow} t$) であるか否かは良く知られているように決定可能である [2]．しかし，$s = t$ が R の帰納的定理であるか否かは決定不能である．実際，以下の命題が知られている．

命題 2 項書き換えシステム R は合流性，停止性，十分完全性をみたす構成子システムとする．このとき，等式 $s = t$ が R の帰納的定理であるか否かは決定不能である．

したがって，項書き換えシステム R を合流性，停止性，十分完全性をみたす構成子システムに限定しても，帰納的定理の判定問題は自明とはならない．なお，左線形で重なりがないという条件を R に付け加え，R に対する制限をさらに強くしても，命題 2 は成立する．

3.4 帰納的定理の決定可能性

項書き換えシステム R は合流性，停止性，十分完全性をみたす構成子システムとする．ここでは，R のもとで等式 $s = t$ が帰納的定理か否かを補題 1 をもちいて判定する．まず，$R_1 = R$，$R_2 = R_1 \cup \{s \to t\}$ とおく．このとき，R_2 が強正規であると仮定する．すると，R_1 と R_2 は補題 1 の条件 (1)(2)(3) をみたすことに注意する．

定理 1 $s \in T(F,V)$ の被覆代入集合 $\{\sigma_i\}_i$ を考える．さらに，任意の i に対して $p_i, q_i \in T(C,V)$ が存在し $s\sigma_i \to_1 \overset{*}{\to}_2 p_i$ かつ $t\sigma_i \overset{*}{\to}_2 q_i$ が成立するものとする．このとき，等式 $s = t$ が帰納的定理となるための必要十分条件は，任意の i に対して $p_i \equiv q_i$ が成立することである．

系 1 $s \in T(F,V)$ の被覆代入集合 $\{\sigma_i\}_i$ を考える．さらに，任意の i に対して $p_i, q_i \in T(C,V)$ が存在し，$s\sigma_i \to_1 \overset{*}{\to}_2 p_i$ かつ $t\sigma_i \overset{*}{\to}_2 q_i$ が成立するものとする．このとき，等式 $s = t$ が帰納的定理であるか否かは決定可能である．

例 2 次の項書換えシステム R を考える．

$$R : \begin{cases} x + 0 \to x \\ x + s(y) \to s(x + y) \end{cases}$$

このとき，$R_1 = R$ は合流性，停止性，十分完全性をみたす構成子システムである．

まず，等式 $s(0)+x = s(x)$ が帰納的定理か否かを判定する．$R_2 = R_1 \cup \{s(0)+x \to s(x)\}$ とおく．このとき，R_2 は強正規となる．ここで，$s(0)+x$ に対する被覆代入集合 $\{[x := 0], [x := s(y)]\}$ を考える．すると，$(s(0)+x)[x := 0] \equiv s(0)+0 \to_1 s(0) \equiv x[x := 0]$ および $(s(0)+x)[x := s(y)] \equiv s(0)+s(y) \to_1 s(s(0)+y) \to_2 s(s(y)) \equiv s(x)[x := s(y)]$．よって定理 1 より，等式 $s(0)+x = s(x)$ は帰納的定理となる．

次に，等式 $z+x=x$ が帰納的定理か否かを判定する．$R_2=R_1\cup\{z+x\to x\}$ とおき，同様の被覆代入集合を考えると，$(z+x)[x:=0]\equiv z+0\to_1 z\neq 0\equiv x[x:=0]$. よって定理1より，等式 $z+x=x$ は帰納的定理ではない．□

等式 $s=t$ が単純とは $s\equiv f(x_1,\cdots,x_n)$，$f\in D$，$t\in T(C,V)$ のことである．ただし，$x_i\not\equiv x_j$ $(i\neq j)$.

定理2 項書き換えシステム R の書き換え規則 $l\to r$ の右辺 r に出現する定義関数記号は左辺 l に出現しているものとする．このとき，単純な等式 $f(x_1,\cdots,x_n)=t$ が帰納的定理か否かは決定可能である．

例3 次の項書換えシステムRを考える．

$$R:\begin{cases} I(0)\to 0 \\ I(s(0))\to s(0) \\ I(s(s(x)))\to s(I(s(I(x)))) \end{cases}$$

$R_1=R$ は合流性，停止性，十分完全性をみたす構成子システムである．ここで，等式 $I(y)=y$ が帰納的定理か否かを判定する．等式 $I(y)=y$ は単純なので，定理2より帰納的定理か否かは決定可能である．まず，$I(y)$ に対する被覆代入集合 $\{[y:=0],[y:=s(0)],[y:=s(s(x))]\}$ を考える．すると，$I(y)[y:=0]\equiv I(0)\to_1 0\equiv y[y:=0]$ および $I(y)[y:=s(0)]\equiv I(s(0))\to_1 s(0)\equiv y[y:=s(0)]$．さらに，$I(y)[y:=s(s(x))]\equiv I(s(s(x)))\to_1 s(I(s(I(x))))\to_2 s(I(s(x)))\to_2 s(s(x))\equiv y[y:=s(s(x))]$ を得る．したがって，定理1より $I(y)=y$ は帰納的定理となる．□

例3では書き換え規則 $I(s(s(x)))\to s(I(s(I(x))))$ の右辺に出現する定義関数 I が入れ子になっていることに注意する．被覆帰納法に基づくKapurら[9, 5]の方法では，右辺に出現する定義関数の内側には構成子しか出現を許されず，例3のような書き換えシステムのクラスを取り扱うことはできない．一方，書き換え帰納法に基づく我々の方法では，書き換えシステム R の停止性のみに基づいた整礎帰納法をもちいているため，書き換え規則に関する構文的な制限は不要となり，より一般的なクラスに適用することが可能となる．

4 完備化手続きによるプログラム変換

関数型言語におけるプログラム変換では，変換手続きの停止性に関してさまざま研究[19, 15]がなされている．これに対して，完備化手続きを直接用いたプログラム融合変換の停止条件については，ほとんど知られていない．本研究では，完備化手続きを用いたプログラム融合変換手続きの停止条件を考察し，停止性を保証するための融合変換手続きの改良方法を提案する．

4.1 完備化を用いたプログラム融合変換

完備化を用いたプログラム融合変換は以下の手順で行なう.

(a) 融合させる合成関数の定義 E_h を，項 t を用いて次のように与える.

$$h(v_1, \ldots, v_n) = t$$

(b) 融合項 t に関する等式と，その他の基本関数に関する等式とを合せた集合 E を完備化する.

(c) 完備化手続きが成功し，停止したときに得られる規則 R のうち，関数 h を定義している規則 R_h が融合変換で得られたプログラムに対応する.

[例 1: *Length* と@ の合成]

$H(xs, ys) = Len(xs@ys)$ はリスト xs と ys の連接リストの長さを返す関数である. 融合変換された $H(xs, ys)$ の定義は R_h となる [?].

$$\text{入力 } E \begin{cases} E_h \{ \underline{H(xs, ys) = Len(xs@ys)} \\ Len([\,]) = 0 \\ Len(x :: xs) = S(0) + Len(xs) \\ 0 + x = x \\ S(x) + y = S(x + y) \\ [\,]@y = y \\ (x :: xs)@ys = x :: (y@z) \end{cases}$$

$$\text{出力 } R \supset R_h \begin{cases} \underline{H((x :: xs), ys) \rightarrow S(H(xs, ys))} \\ \underline{H([\,], ys) \rightarrow Len(ys)} \end{cases}$$

4.2 変換手続きの停止条件

完備化手続きを用いたプログラム融合変換の実験を計算機上で行ない，変換手続きが発散して失敗する原因を調査した. その結果，ほとんどの失敗例では，以下の方法を組み合わせることにより，変換手続きを停止させることができることが明らかになった.

(a) 関数の帰納的性質を規則として追加

(b) 完備化する等式の制限

(c) 関数の重みの調整

(d) 反復パターンを新しい関数として定義

このうち，(a)〜(c) はこれまでの我々の研究ですでに知られていた手法であり，(d) は本研究で新たに提案された. 完備化手続きを用いた融合変換が停止しない原因の一つは，危険対が発散することである. これは，類似した重なりが繰り返し出現し，次々と危険対を生成するためである. このとき，危険対により生成される規則を新

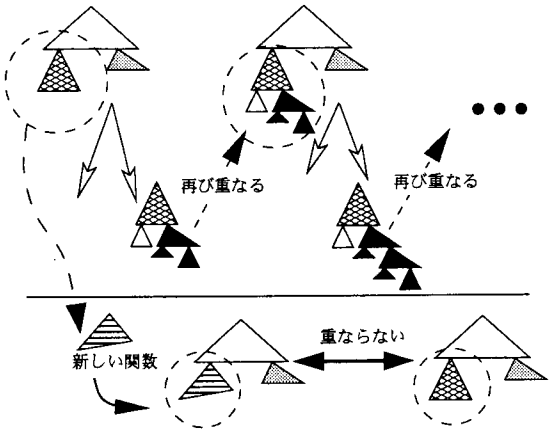

図 1: **繰り返し生成される危険対とその回避策**

しい関数記号で定義すると，類似した危険対の生成パターンが繰り返し出現することを回避できる場合がある (図 1).

関数型言語におけるプログラム変換では，変換の適用パターンの有限性に基づいて停止性が示されている [19, 15]. 同様に，完備化手続きよるプログラム変換においても，危険対の生成パターンの有限性に基づいて停止性が保証可能と考えられる.

5 変換パターンによる高階プログラム変換

関数型言語のプログラム変換法として，2 階変換パターンに基づくプログラム変換法が有効である [6, 3, 13]. しかし，実際の関数型プログラムでは，関数引数をもつ高階関数が広く使用されているため，より高階な変換パターンに基づいたプログラム変換法が必要となる. 本研究では，3 階変換パターンに基づくプログラム変換法を提案し，高階関数を含むプログラムが 3 階変換パターンによって変換可能であることを例で示す. さらに，高階プログラム変換の問題点についても考察する.

5.1 変換パターンに基づくプログラム変換

引数として二項演算子 g，リスト x，初期値 y をとり，g を x に作用させる高階関数プログラム P を考える.

$P:\ f(g,x,y) \Leftarrow \textbf{if}\ Null(x)\ \textbf{then}\ y$
$\qquad\qquad\qquad\qquad \textbf{else}\ g(Car(x), f(g, Cdr(x), y))$

このプログラムでは $f(+,[1,3,5],0) = 1{+}3{+}5{+}0 = 9$ や $f(*,[2,4,6],1) = 2{*}4{*}6{*}1 = 48$ 等が求まる. 一方，プログラム P は次のプログラム P' で表現することもできる.

$P':\ f'(g,x,y) \Leftarrow \textbf{if}\ Null(x)\ \textbf{then}\ y$
$\qquad\qquad\qquad\qquad \textbf{else}\ f'(g, Cdr(x), g(Car(x), y))$

P と P' の実行プロセスを比較すると，P が再帰的プロセスであるのに対し，P'

は反復的プロセスとなっており，メモリの使用領域からみても P' の方が効率の良いプログラムとなっている．つまり，あるプログラム中に P が存在するなら，それを P' に置き換えることで，そのプログラムを効率の良いプログラムに変換することができる．

プログラム P と P' を一般化すると，次のパターン Σ と Σ' が得られる．

$\Sigma : f(g,x,y) \Leftarrow \textbf{if } a(x) \textbf{ then } y$
$\qquad\qquad \textbf{else } h(g,d(x),f(g,e(x),y))$

$\Sigma' : f'(g,x,y) \Leftarrow \textbf{if } a(x) \textbf{ then } y$
$\qquad\qquad \textbf{else } f'(g,e(x),h(g,d(x),y))$

この Σ，Σ' に次の代入 σ をそれぞれ適用することで
P，P' となる．

$$\sigma : \begin{cases} a \rightarrow \lambda x_1.Null(x_1) \\ d \rightarrow \lambda x_1.Car(x_1) \\ e \rightarrow \lambda x_1.Cdr(x_1) \\ h \rightarrow \lambda x_3.\lambda x_2.\lambda x_1.x_3(x_2,x_1) \end{cases}$$

したがって，プログラム P から P' への変換 $P \Rightarrow P'$ は，パターン Σ から Σ' への変換 $\Sigma \Rightarrow \Sigma'$ に一般化できる．このとき，$\Sigma \Rightarrow \Sigma'$ を変換パターンとよぶ．ただし，Σ と Σ' が等価となるためには，以下の条件が必要である．

$\mathcal{X} : \forall xy \;\; h(g,x,h(g,y,z)) = h(g,h(g,x,y),z)$

5.2 考察

変換パターンに基づく高階プログラム変換を実現するためには，以下の 3 点が不可欠である．

(1) 有効な 3 階変換パターン $\Sigma \Rightarrow \Sigma'$

(2) 効率的なパターン照合アルゴリズム

(3) 等価条件 $\mathcal{X}$ の検証手続き

(1): 2 階変換パターンは [6] で提案されているが，3 階変換パターンに関しては，我々が提案した変換パターン以外は知られていない．

(2): 3 階パターンの照合に関して，文献 [6, 3] のアルゴリズムは適用することが困難である．また，文献 [13] のアルゴリズムも，2 階パターンまでしか正当性は保証されていない．しかし，本報告の 3 階パターンに対しては，文献 [13] のアルゴリズムで照合が成功することを確認できた．

6 結論

本研究では，高階書き換えシステムに基づくプログラム検証法の基礎を明らかにすることを目的に，書き換え帰納法による帰納的定理の決定手続き [17]，完備化手

続きによるプログラム融合変換 [7]，変換パターンに基づく高階プログラム変換 [18] を考察した．ここで得られた結果は，高階書き換えシステムに基づく関数型プログラムの開発・検証の自動化の基礎として有効であると考えられる．

今後の目標としては，書き換えシステムの帰納的性質の検証技術と，プログラムの自動変換手法を組み合わせた，より強力なソフトウェア検証システムの提案がある．そのための，ここで開発されたシステムの拡張，高階理論の整備，検証システムの効率的な実現法は今後に残された課題である．

謝辞

本研究に対して多大な御支援を賜りました財団法人石田 (實) 記念財団に心より感謝いたします．

参考文献

[1] 秋谷 賢司，草刈 圭一朗，外山 芳人，項書き換え系の高速実行と柔軟実行の融合，電気関係学会東北支部連合大会予稿集 (2002-8) p.139.

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

[3] R. Curien, Z. Qian and H. Shi, Efficient Second-Order Matching, Lecture Notes in Computer Science 1103 (1996) 317–331.

[4] 福井信一，外山芳人，帰納的定理の決定可能なクラスについて，信学技報 COMP2001-81 (2002) 57-64.

[5] J. Giesl and D. Kapur, Decidable class of inductive theorems, LNCS 2083 (2001) 469-484.

[6] G. Huet and B. Lang, Proving and Applying Program Transformations Expressed with Second-Order Patterns, Acta Informatica 11 (1978) 31–55.

[7] 伊藤 芳浩，草刈 圭一朗， 外山 芳人，完備化手続きによるプログラム融合変換の停止条件，電気関係学会東北支部連合大会予稿集 (2002-8) p.140.

[8] D. Kapur, P. Narendran and H. Zhang, On Sufficient-Completeness and Related Properties of Term Rewriting Systems, Acta Informatica 24 (1987) 395-415.

[9] D. Kapur and M. Subramaniam, Extending decision procedures with induction schemes, LNCS 1831 (2000) 324-345.

[10] 小池広高，外山芳人，潜在帰納法と書換え帰納法の比較，コンピュータソフトウェア 17 (2000) 1-12.

[11] K. Kusakari and Y. Toyama, On Proving AC-termination by AC-dependency pairs, *IEICE TRANS. INF. & SYST., Vol. E84-D, No. 5* (2001) 604-612.

[12] R. Mayr and T. Nipkow, Higher-order rewrite systems and their confluence, Theoretical Computer Science 192 (1998) 3–29.

[13] M. Oege and S. Ganesh, Higher-order matching for program transformation, Theoretical Computer Science 269 (2001) 135–162.

[14] U. S. Reddy, Term rewriting induction, Lecture Notes in Computer Science 449 (1990) 162-177.

[15] H. Seidl and M. H. Srensen, Constraints to Stop Deforestation, *Science of Computer Programming* 32 (1998) 73–107.

[16] Y. Toyama and M. Oyamguchi, Conditional Linearization of Non-Duplicating Term Rewriting Systems, *IEICE TRANS. INF. & SYST., Vol. E84-D, No. 4* (2001) 439-447.

[17] 外山 芳人，書き換え帰納法による帰納的定理の決定手続き，日本ソフトウェア科学会第１９回大会論文集 (2002-9) 3A-2.

[18] 鶴川 敏孝， 草刈 圭一朗， 外山 芳人，変換パターンに基づく高階プログラム変換，電気関係学会東北支部連合大会予稿集 (2002-8) p.141.

[19] P. Wadler, Deforestation:Transforming Programs to eliminate trees, *Theoretical Computer Science* 73 (1990) 231–248.