書き換え帰納法に基づくプログラム融合変換

Fusion Transformation based on Rewriting Induction

坂本邦彦　青戸等人　外山芳人
Kunihiko SAKAMOTO　Takahito AOTO　Yoshihito TOYAMA

東北大学 電気通信研究所
RIEC, Tohoku University
{sakamoto,aoto,toyama}@nue.riec.tohoku.ac.jp

関数型プログラムの効果的なプログラム変換法として融合変換が知られている．一方，帰納的定理の強力な自動証明法として書き換え帰納法がある．本論文では，定理自動証明法である書き換え帰納法をもちいてプログラムの融合変換が実現できることを示す．書き換え帰納法に基づく融合変換システムの特徴は，自動証明システムとプログラム変換システムが一体化しているため，プログラム変換に必要な補題を自動証明しながら変換を進めることが可能となることである．われわれの実装は S 式書き換えシステムに基づいているため，S 式の表現力を生かした高階プログラムの融合変換も容易に実現可能である．

1　はじめに

プログラム融合変換 [2, 5] は，関数型プログラムの実行時に生成される中間データ構造をプログラム変換によって除去することで，効率的なプログラムを生成する．

例えば，2 つのリストを連接する関数 app は以下のように定義される．

$$\mathrm{app}(\mathrm{nil},\ ys) = ys$$
$$\mathrm{app}(x : xs,\ ys) = x : \mathrm{app}(xs,\ ys)$$

このとき app をもちいると，3 つのリストを連接する関数 h を以下のように定義できる．

$$\mathrm{h}(xs,\ ys,\ zs) = \mathrm{app}(\mathrm{app}(xs,\ ys),\ zs)$$

この関数 h は，まず xs と ys を連接したリストを作り，それを zs と連接する．xs と ys の連接リストは h の実行後には不要となり捨てられる．

ここで h を融合変換すると，以下のような関数定義が得られる．

$$\mathrm{h}(\mathrm{nil},\ ys,\ zs) = \mathrm{app}(ys,\ zs)$$
$$\mathrm{h}(x : xs,\ ys,\ zs) = x : \mathrm{h}(xs,\ ys,\ zs)$$

このとき関数は直接定義されているため，余分な中間リストは生成されず，効率の良い実行が可能となる．

融合変換に限らず多くのプログラム変換では，一般に自然数やリストなどのデータ構造に基づくプログラムの帰納的な性質を補題や変換規則として利用している．Reddy[3] により提案された書き換え帰納法は，項書き換えシステムに基づく帰納的定理の強力な自動証明法であり，プログラム変換で必要とされる補題の証明に適している．

本論文では，定理自動証明法である書き換え帰納法に基づいたプログラムの融合変換法を提案する．われわれの提案したプログラム融合変換法は， 書き換え帰納法を直接適用して融合変換を実現しているため，自動証明システムとプログラム変換システムが一体となって働く．したがって，プログラム変換で必要な補題を自動証明しながら，その証明結果を利用して変換を進めることが可能である．

関数型プログラムの形式的モデルとして，本論文では S 式書き換えシステムをもちいる [4] ．S 式書き換えシステムは，高階システムの取り扱いが簡単であるという特徴をもつ．さらに，書き換え帰納法で不可欠なリダクションの停止性を，高階にもかかわらず辞書式経路順序で簡単に判定することが可能である．本論文では，S 式書き換えシステムに対する書き換え帰納法を実装し，それをもちいてプログラム融合変換が実現できることを示す．

2　S 式書き換えシステム

ここでは，S 式書き換えシステム [4] について簡単に説明する．本論文でもちいる S 式書き換えシステムは型付きシステムであり，文献 [1] の単純型付き項書き換えシステムと等価である．

2.1 型

基本型の集合 $\mathcal{B}$ から構成される型の集合 $\mathcal{T}$ を，(i) $\tau \in \mathcal{B}$ なら $\tau \in \mathcal{T}$，(ii) $\tau_0, \ldots, \tau_n \in \mathcal{T}$ なら $\tau_1 \times \tau_2 \times \cdots \times \tau_n \to \tau_0$ $(n \geq 1)$ で定義する．$\tau \notin \mathcal{B}$ は関数型とよぶ．型 τ は $\tau = b_{1,1} \times \cdots b_{1,m} \to (b_{2,1} \times \cdots b_{2,n} \to (\cdots (b_{k,1} \times \cdots b_{k,p} \to b) \cdots))$ かつ $b_{i,j}, b \in \mathcal{B}$ のとき初等的という．

2.2 S 式

型 τ の関数記号の集合を $\mathcal{F}^\tau$，変数の集合を $\mathcal{V}^\tau$ とし，$\mathcal{F} = \bigcup_\tau \mathcal{F}^\tau$ および $\mathcal{V} = \bigcup_\tau \mathcal{V}^\tau$ とする．型 τ の S 式の集合 $\mathcal{S}(\mathcal{F}, \mathcal{V})^\tau$ は，(i)$a \in \mathcal{F}^\tau \cup \mathcal{V}^\tau$ なら $a \in \mathcal{S}(\mathcal{F}, \mathcal{V})^\tau$，(ii) $t_0 \in \mathcal{S}(\mathcal{F}, \mathcal{V})^{\tau_1 \times \tau_2 \times \cdots \times \tau_n \to \tau}$ かつ $t_i \in \mathcal{S}(\mathcal{F}, \mathcal{V})^{\tau_i}$ $(i = 1, \cdots, n)$ なら $(t_0\ t_1\ \cdots\ t_n) \in \mathcal{S}(\mathcal{F}, \mathcal{V})^\tau$ で定義される．$\mathcal{S}(\mathcal{F}, \mathcal{V}) = \bigcup_\tau \mathcal{S}(\mathcal{F}, \mathcal{V})^\tau$ とする．変数を含まない S 式 (基底 S 式) の集合を $\mathcal{S}(\mathcal{F})$ と記す．S 式 t の型 τ を明示するときは t^τ と記す．$\mathcal{V}(t)$ で t に出現する変数集合を表す．

2.3 S 式書き換えシステム

書き換え規則 $l \to r$ は，S 式 l と r が同じ型をもち，$l \notin \mathcal{V}$ かつ $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ をみたすものとする．(型付き)S 式書き換えシステム $\mathcal{R}$ は，書き換え規則の有限集合である．通常の項書き換えシステムと同様に，$\mathcal{R}$ によって $\mathcal{S}(\mathcal{F}, \mathcal{V})$ 上のリダクション関係 $\to_\mathcal{R}$ を定義する．このとき，$\to_\mathcal{R}$ から得られる推移閉包を $\overset{+}{\to}_\mathcal{R}$，同値関係を $\leftrightarrow^*_\mathcal{R}$ と記す．

S 式 s と t が等しい基本型をもつとき，基本型の等式 $s = t$ が帰納的定理とは，任意の基底代入 θ_g に対して $s\theta_g \leftrightarrow^*_\mathcal{R} t\theta_g$ が成立することである [1]．

2.4 構成子システム

関数記号の集合 $\mathcal{F} = \mathcal{D} \cup \mathcal{C}$ は定義関数記号の集合 $\mathcal{D}$ と構成子記号の集合 $\mathcal{C}$ に分割されており，構成子記号の型は初等的であるものとする．S 式書き換えシステムが構成子システムであるとは，書き換え規則 $l \to r$ の左辺 $l = ((\cdots((f\ s_{1,1}\ \cdots s_{1,m})\ s_{2,1}\ \cdots s_{2,n})\ \cdots)\ s_{k,1}\ \cdots\ s_{k,p})$ の頭部記号 f が定義関数記号であり，さらに $s_{i,j}$ が基本型なら $s_{i,j} \in \mathcal{S}(\mathcal{C}, \mathcal{V})$，関数型なら $s_{i,j} \in \mathcal{V}$ をみたすことである．ただし，l に出現する関数型変数はすべて異なるものとする．

S 式書き換えシステム $\mathcal{R}$ が十分完備であるとは，任意の基本型基底 S 式 $s \in \mathcal{S}(\mathcal{F})^b$ に対して基底構成子 S 式 $t \in \mathcal{S}(\mathcal{C})^b$ が存在して $s \overset{*}{\to}_\mathcal{R} t$ となることである．停止性をみたす構成子システムの十分完備性は決定可能であることが知られている [1]．以下では，S 式書き換えシステム $\mathcal{R}$ は停止性をみたす十分完備な構成子システムであると仮定する．

例 1 以下の S 式書き換えシステム $\mathcal{R}$ を考える．ここで，$\mathcal{B} = \{N, L\}$，$\mathcal{F} = \mathcal{D} \cup \mathcal{C}$，
$\mathcal{D} = \{\mathrm{map}^{(N\to N)\times L\to L}, +^{N\to N\to N}\}$，
$\mathcal{C} = \{:^{N\times L\to L}, \mathrm{nil}^L, \mathrm{s}^{N\to N}, 0^N\}$ とする．

$$\mathcal{R}\begin{cases} ((\mathrm{map}\ f)\ \mathrm{nil}) \to \mathrm{nil} \\ ((\mathrm{map}\ f)\ (:\ x\ y)) \to (:\ (f\ x)\ ((\mathrm{map}\ f)\ y)) \\ ((+\ (\mathrm{s}\ x))\ y) \to (\mathrm{s}\ ((+\ x)\ y)) \\ ((+\ 0)\ y) \to y \end{cases}$$

このとき，$\mathcal{R}$ は停止性をみたす十分完備な構成子システムであり，以下のリダクションが得られる．

$$\begin{aligned} &((\mathrm{map}\ (+\ (\mathrm{s}\ 0)))\ (:\ (\mathrm{s}\ 0)\ (:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ \mathrm{nil}))) \to_\mathcal{R} \\ &(:\ ((+\ (\mathrm{s}\ 0))\ (\mathrm{s}\ 0)) \\ &\qquad ((\mathrm{map}\ (+\ (\mathrm{s}\ 0)))\ (:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ \mathrm{nil}))) \overset{+}{\to}_\mathcal{R} \\ &(:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ ((\mathrm{map}\ (+\ (\mathrm{s}\ 0)))\ (:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ \mathrm{nil}))) \to_\mathcal{R} \\ &(:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ (:\ ((+\ (\mathrm{s}\ 0))\ (\mathrm{s}\ (\mathrm{s}\ 0)) \\ &\qquad ((\mathrm{map}\ (+\ (\mathrm{s}\ 0)))\ \mathrm{nil})))) \overset{+}{\to}_\mathcal{R} \\ &(:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ (:\ (\mathrm{s}\ (\mathrm{s}\ (\mathrm{s}\ 0))))) \\ &\qquad ((\mathrm{map}\ (+\ (\mathrm{s}\ 0)))\ \mathrm{nil}))) \to_\mathcal{R} \\ &(:\ (\mathrm{s}\ (\mathrm{s}\ 0))\ (:\ (\mathrm{s}\ (\mathrm{s}\ (\mathrm{s}\ 0)))\ \mathrm{nil})). \end{aligned}$$

以下では，関数定義記号，構成子記号，型などが明らかな場合にはとくに明示しない．

3 書き換え帰納法

Reddy[3] によって提案された書き換え帰納法は，項書き換えシステムに基づく帰納的定理の自動証明法である．本論文では，S 式書き換えシステムに基づく書き換え帰納法を提案する．

3.1 基本部分項

S 式書き換えシステム $\mathcal{R}$ は停止性をみたす十分完備な構成子システムとし，ある簡約順序 $>$ と両立しているものとする (つまり，$\forall l \to r \in \mathcal{R}.\ l > r$)．S 式 t の部分項 s が基本 部分項とは，s が基本型であり，$s = ((\cdots((f\ s_{1,1}\ \cdots s_{1,m})\ s_{2,1}\ \cdots s_{2,n})\ \cdots)\ s_{k,1}\ \cdots\ s_{k,p})$ とすると頭部記号 f が定義関数記号，$s_{i,j}$ が基本型なら $s_{i,j} \in \mathcal{S}(\mathcal{C}, \mathcal{V})$ となることである．S 式 t

の最内基本部分項の集合を $IB(t)$ とする．$\mathcal{R}$ が十分完備な構成子システムであることより，任意の基本部分項 s に対して，ある書き換え規則 $l \to r \in \mathcal{R}$ が存在して，s と l は単一化可能である．このとき，s と l の最汎単一化子を $mgu(s,l)$ で表す．また，s に対する最汎単一化子の集合を $\Theta(s) = \{\theta \mid \theta = mgu(s,l) \text{ for some } l \to r \in \mathcal{R}\}$ とする．

3.2 推論規則

S 式書き換えシステムに基づく書き換え帰納法の手続きを，Reddy[3] と同様に集合の組 (E,H) に対する以下の推論規則として与える．ここで，E は証明しようとする基本型の等式の集合，H は帰納法の仮定として利用可能な書き換え規則の集合である．$s \doteq t$ は等式 $s = t$ あるいは $t = s$ を表す．

$$\textit{Expand} \quad \frac{(E \cup \{s \doteq t\}, H)}{(E \cup E', H \cup \{s \to t\})} \quad \begin{array}{l}\text{if } s > t,\ u \in IB(s),\\ E' = \{s' \doteq t\theta \mid s\theta \\ \to_{\mathcal{R}} s' \text{ for some} \\ \theta \in \Theta(u)\}\end{array}$$

$$\textit{Simplify} \quad \frac{(E \cup \{s \doteq t\}, H)}{(E \cup \{s' \doteq t\}, H)} \quad \text{if } s \to_{\mathcal{R} \cup H} s'$$

$$\textit{Delete} \quad \frac{(E \cup \{s \doteq s\}, H)}{(E, H)}$$

上の推論規則を n 回 $(n \geq 0)$ 適用して (E_0,H_0) から (E_n,H_n) が導出されるなら，$(E_0,H_0) \vdash (E_1,H_1) \vdash \cdots \vdash (E_n,H_n)$
あるいは $(E_0,H_0) \vdash^* (E_n,H_n)$ と記す．このとき，Reddy[3] と同様に，S 式書き換えシステムの書き換え帰納法に対しても以下の定理が成立する．

定理 1 $(E,\emptyset) \vdash^* (\emptyset,H)$ なら E の各等式は $\mathcal{R}$ の帰納的定理である．

4 S 式上の辞書式経路順序

ここでは，書き換え帰納法でもちいる簡約順序として，S 式上の辞書式経路順序を定義する [4]．

定義 (S 式上の辞書式経路順序) 関数記号集合 $\mathcal{F}$ 上の半順序を $>$ とする．このとき，$\mathcal{S}(\mathcal{F},\mathcal{V})$ 上の辞書式経路順序 $\succ_{lps}$ は以下のよう定義される．
$s \succ_{lps} t$ が成立するのは，$s = (s_1\ \cdots\ s_m)$ あるいは $s \in \mathcal{F}$, $t = (t_1\ \cdots\ t_n)$ あるいは $t \in \mathcal{F} \cup \mathcal{V}$，かつ以下のどれかが成り立つことである．

(S0) $s,t \in \mathcal{F}$ かつ $s > t$，
(S1) $s \in \mathcal{F}$ かつ $\forall i.\ s \succ_{lps} t_i$，
(S2) $\exists i.\ s_i \succeq_{lps} t$，
(S3) $m > n$ かつ $s_1 = t_1, \cdots, s_n = t_n$，
(S4) $\exists i.\ s_1 = t_1, \cdots, s_{i-1} = t_{i-1}, s_i \succ_{lps} t_i, s \succ_{lps} t_{i+1}, \cdots, s \succ_{lps} t_n$.

辞書式経路順序 $\succ_{lps}$ は簡約順序であり以下の定理が成立する [4]．

定理 2 S 式書き換えシステム $\mathcal{R}$ が辞書式経路順序 $\succ_{lps}$ と両立するならば $\mathcal{R}$ は停止性をみたす．

5 書き換え帰納法による融合変換

最初に示した $\mathrm{app}(\mathrm{app}(xs,ys),zs)$ の融合変換を例に，書き換え帰納法をもちいた融合変換法を説明する．

関数 app の定義を S 式書き換えシステムで記述すると，以下のようになる．

$$\mathcal{R}\begin{cases} (\text{app nil } ys) & \to & ys & \text{(R1)} \\ (\text{app } (:\ x\ xs)\ ys) & \to & (:\ x\ (\text{app } xs\ ys) & \text{(R2)} \end{cases}$$

この R のもとで，
$E_0 = \{(\text{h } xs\ ys\ zs) = (\text{app } (\text{app } xs\ ys)\ zs)\}$
$H_0 = \emptyset$
として書き換え帰納法を実行する．簡約順序としては，関数記号の重みを $\mathrm{app} > \mathrm{h} > : > \mathrm{nil}$ とした辞書式経路順序をもちいる．

書き換え帰納法の実行過程を図 1 に示す．書き換え帰納法の手続きは，等式集合が以下の E_4 なった時点で，どの推論規則も適用できなくなり停止する．

$$E_4\begin{cases} (\text{h nil nil } zs) & = & zs \\ (\text{h nil } (:\ y\ ys)\ zs) & = & (:\ y\ (\text{h nil } ys\ zs)) \\ (\text{h } (:\ x\ xs)\ ys\ zs) & = & (:\ x\ (\text{h } xs\ ys\ zs)) \end{cases}$$

このとき，E_4 は関数 h の定義を融合変換したものになっている．

6 融合変換の成功例

以下では，現在実装中の書き換え帰納法をもちいた融合変換の成功例をいくつか示す．

例 2 以下の R と $E(=E_0)$ をもちいて書き換え帰納法を実行する．

$$R\begin{cases} (\text{len nil}) & \to & 0 \\ (\text{len } (:\ x\ xs)) & \to & (\text{s } (\text{len } xs)) \\ (\text{app nil } ys) & \to & ys \\ (\text{app } (:\ x\ xs)\ ys) & \to & (:\ x\ (\text{app } xs\ ys)) \end{cases}$$

Rule	E_i	H_{i+1}, simplifications
Expand	$(\mathrm{h}\ xs\ ys\ zs) = (\mathrm{app}\ \boxed{(\mathrm{app}\ xs\ ys)}\ zs)$	$(\mathrm{app}\ (\mathrm{app}\ xs\ ys)\ zs) \to (\mathrm{h}\ xs\ ys\ zs)$ (H1)
Simplify	$(\mathrm{h}\ \mathrm{nil}\ ys\ zs) = (\mathrm{app}\ ys\ zs)$ $(\mathrm{h}\ (:\ x\ xs)\ ys\ zs) = (\mathrm{app}\ (:\ x\ (\mathrm{app}\ xs\ ys))\ zs)$	R1, H1
Expand	$(\mathrm{h}\ \mathrm{nil}\ ys\ zs) = \boxed{(\mathrm{app}\ ys\ zs)}$ $(\mathrm{h}\ (:\ x\ xs)\ ys\ zs) = (:\ x\ (\mathrm{h}\ xs\ ys\ zs)$	$(\mathrm{app}\ ys\ zs) \to (\mathrm{h}\ \mathrm{nil}\ ys\ zs)$ (H2)
Simplify	$(\mathrm{h}\ \mathrm{nil}\ \mathrm{nil}\ zs) = zs$ $(\mathrm{h}\ \mathrm{nil}\ (:\ y\ ys)\ zs) = (:\ y\ (\mathrm{app}\ ys\ zs))$ $(\mathrm{h}\ (:\ x\ xs)\ ys\ zs) = (:\ x\ (\mathrm{h}\ xs\ ys\ zs))$	H2
	$(\mathrm{h}\ \mathrm{nil}\ \mathrm{nil}\ zs) = zs$ $(\mathrm{h}\ (:\ y\ ys)\ zs) = (:\ y\ (\mathrm{h}\ \mathrm{nil}\ ys\ zs))$ $(\mathrm{h}\ (:\ x\ xs)\ ys\ zs) = (:\ x\ (\mathrm{h}\ xs\ ys\ zs))$	

図 1: 書き換え帰納法による融合変換の過程

$$E\left\{\begin{array}{lcl}(\mathrm{h}\ xs\ ys) & = & (\mathrm{len}\ (\mathrm{app}\ xs\ ys))\end{array}\right.$$

すると，以下の E' が得られて融合変換は成功する．

$$E'\left\{\begin{array}{lcl}(\mathrm{h}\ \mathrm{nil}\ ys) & = & (\mathrm{len}\ ys)\\ (\mathrm{h}\ (:\ x\ xs)\ ys) & = & (\mathrm{s}\ (\mathrm{h}\ xs\ ys))\end{array}\right.$$

例 3 以下の R と $E(=E_0)$ をもちいて書き換え帰納法を実行する．

$$R\left\{\begin{array}{lcl}(\mathrm{len}\ \mathrm{nil}) & \to & 0\\ (\mathrm{len}\ (:\ x\ xs)) & \to & (\mathrm{s}\ (\mathrm{len}\ xs))\\ (\mathrm{app}\ \mathrm{nil}\ ys) & \to & ys\\ (\mathrm{app}\ (:\ x\ xs)\ ys) & \to & (:\ x\ (\mathrm{app}\ xs\ ys))\end{array}\right.$$

$$E\left\{\begin{array}{lcl}(\mathrm{h}\ xs) & = & (\mathrm{len}\ (\mathrm{app}\ xs\ xs))\\ (\mathrm{len}\ (\mathrm{app}\ xs\ (:\ y\ ys))) & = & (\mathrm{s}\ (\mathrm{len}\ (\mathrm{app}\ xs\ ys)))\end{array}\right.$$

すると，以下の E' が得られて融合変換は成功する．

$$E'\left\{\begin{array}{lcl}(\mathrm{h}\ \mathrm{nil}) & = & 0\\ (\mathrm{h}\ (:\ x\ xs)) & = & (\mathrm{s}\ (\mathrm{s}\ (\mathrm{h}\ xs)))\end{array}\right.$$

この例では，E の最初の式のみで融合変換を試みると，書き換え帰納法の手続きが停止しなくなる．E の 2 番目の式を同時に与えることにより，この式が補題としてもちいられ，融合変換に成功する．さらに，融合変換と同時に補題の証明も行っている．

実装では S 式書き換えシステムをもちいているため，高階関数の融合変換も可能である．

例 4 以下の R と $E(=E_0)$ をもちいて書き換え帰納法を実行する．

$$R\left\{\begin{array}{lcl}((\mathrm{map}\ f)\ \mathrm{nil}) & \to & \mathrm{nil}\\ ((\mathrm{map}\ f)\ (:\ x\ xs)) & \to & (:\ (f\ x)\ ((\mathrm{map}\ f)\ xs))\\ ((\mathrm{foldr}\ f)\ e\ \mathrm{nil}) & \to & e\\ ((\mathrm{foldr}\ f)\ e\ (:\ x\ xs)) & \to & (f\ x\ ((\mathrm{foldr}\ f)\ e\ xs))\end{array}\right.$$

$$E\left\{\begin{array}{lcl}(\mathrm{h}\ f\ g\ e\ xs) & = & ((\mathrm{foldr}\ f)\ e\ ((\mathrm{map}\ g)\ xs))\end{array}\right.$$

すると，以下の E' が得られて高階システムの融合変換は成功する．

$$E'\left\{\begin{array}{lcl}(\mathrm{h}\ f\ g\ e\ \mathrm{nil}) & = & e\\ (\mathrm{h}\ f\ g\ e\ (:\ x\ xs)) & = & (f\ (g\ x)\ (\mathrm{h}\ f\ g\ e\ xs))\end{array}\right.$$

謝辞 本研究の一部は科学研究費 16016202，14580357 の援助を受けている．

参考文献

[1] T. Aoto, T. Yamada, and Y. Toyama, Inductive theorems for higher-order rewriting, *Lecture Notes in Comput. Sci. 3091* (2004) 269-284.

[2] W.-N. Chin, Safe fusion of functional expressions II: further improvements, *Journal of Functional Programming 4(4)* (1994) 515–555.

[3] U. S. Reddy, Term rewriting induction, *Lecture Notes in Comput. Sci. 449* (1990) 162–177.

[4] Y. Toyama, Termination of S-expression rewriting systems: lexicographic path ordering for higher-order terms, *Lecture Notes in Comput. Sci. 3091* (2004) 40-54.

[5] P. Wadler, Deforestation: transforming programs to eliminate trees, *Theoretical Computer Science 73* (1990) 231–248.