LB-1

単一後継関数を持つ再帰プログラムからの再帰除去及び閉式化

Recursion Removal and Solving from Recursive Programs with only one Descent Function

市川 祐輔* 小西 善二郎† 二村 良彦‡
Yuusuke Ichikawa Zenjiro Konishi Yoshihiko Futamura

1 はじめに

再帰プログラムは反復プログラムに比べ読みやすく書きやすいが，計算機で実行する際には再帰呼び出しとスタック操作が必要である．そのため，インライン展開ができない，局所参照性が悪いなどのプログラム最適化上の問題を引起す．従って，再帰プログラムを計算量を増加させずにスタックを使用しない反復プログラムに変換する再帰除去法が1970年代より研究されてきた．

本稿では，単一後継関数を持つ非線形相互再帰プログラムからの再帰除去及び閉式化のアルゴリズム（累積関数法）を提案する．そしてこの理論に基づく，数値計算（用）再帰プログラムに対する再帰除去及び閉式化システムについて報告をする．ただし，**単一後継関数を持つ再帰プログラム**とは，以下の形式のプログラムであり，累積関数法はこの相互再帰形式に対しても適用可能である．ここで，a，b，c及びdはfに対する再帰呼び出しを含まないものとし，それぞれ補助関数，基底関数，制御関数そして後継関数と呼ばれる．

$$f(x) = \textbf{if } p(x) \textbf{ then } b(x) \textbf{ else } a(c(x), f(d(x)), f(d^2(x)), \ldots, f(d^n(x))) \quad (1)$$

このような非線形再帰プログラムからの再帰除去法としては従来からタプリング法が使われてきた．しかし，タプリング法を機械的に適用するためには，後継関数$d(x)$がある意味での逆関数をもつ必要がある[3, 6]．従って図1に示したプログラムにタプリング法を機械的に適用することは不可能である．機械的にタプリング法を適用すると，反復プログラムには変換できても，$d(x)$に関する計算量を元のプログラムよりも増やすことになり，プログラムの高性能化が達成できない可能性があるからである．一方，累積関数法は$d(x)$の性質には依存しない．また，$c(x)$が副作用を持ち，$c(x)$の計算順序を変更できない場合にはタプリング法は適用不可能である．なぜならば，タプリング法では$c(x)$を逆に，即ちボトムアップに計算するからである．一方，累積関数法は$c(x)$の計算順序を保存するので，$c(x)$が副作用を持つ場合にも適用可能である（紙面の都合により，この点に関する議論には本稿ではこれ以上立ち入らない）．従って，累積関数法とタプリング法は相補的な関係にある．さらに，累積関数法は，再帰除去のみならずプログラムの閉式化にも応用が可能である．実際，本システムによって図1の機械的な再帰除去が可能であり，さらにドミノ詰めパズル[2]のような相互再帰プログラム（図2）の自動閉式化も可能である．以下では，まず拡張された累積関数の定義及びこれを用いた再帰除去定理を述べ，次に本稿のシステムによる再帰除去及び閉式化の例を説明する．本稿では，再帰プログラムの計算を最左最内規則に基づいて行なう．

*早稲田大学理工学研究科
†早稲田大学ソフトウェア生産技術研究所
‡早稲田大学理工学部

2 累積関数

累積関数を用いた再帰除去法は，我々が1998年に提案した方法であり，多くの線形再帰プログラム（再帰呼び出しを実質的に1つしか含まない再帰プログラム）に対する再帰除去を可能にした[1]．さらに数値計算（用）再帰プログラムの一部に対しては，累積関数により閉じた式への変換（閉式化，即ち漸化式の求解）も可能にした[7]．累積関数法は，末尾再帰からの再帰除去及び結合性を用いた再帰除去を含む方法である．本研究では，累積関数を次の三種の再帰プログラムに適用するよう拡張した．1) 単一後継関数を持つ再帰プログラム，2) 相互再帰プログラム，3) 多重else-if文(case文)を持つ再帰プログラム．本稿では1)に対する累積関数の拡張のみを述べ，2)及び3)は複雑なためここでは省略する．なお，1)の制限された形式である木再帰プログラム（$n = 2$の場合）への拡張は[5]により行なわれており，それに基づく1)への拡張は第三著者の二村が行なった．また，2)，3)への拡張は第一著者の市川が行なった．

定義 2.1 累積関数
プログラム(1)に対して，下記の性質を持つ関数$h(v, u, d(u), d^2(u), \ldots, d^{n-1}(u))$を$a$に関する$f$の**累積関数**(cumulative function)と呼ぶ:
「$\neg p(u)$ならば任意の式vに対して，
$a(v, f(u), f(d(u)), f(d^2(u)), \ldots, f(d^{n-1}(u)))$
$= a(h(v, u, d(u), d^2(u), \ldots, d^{n-1}(u)), f(d(u)),$
$f(d^2(u)), \ldots, f(d^n(u)))$ となる．ただし，hはfへの再帰呼び出し及びその他の自由変数を含んではならず，$p(d^i(x))$を満たす最小の自然数iをNとするとき，$N + n > i \geq N$なる任意の自然数iに対して$p(d^i(x))$は成立するものとする.」

定理 2.2 累積関数による再帰除去定理
プログラム(1)が累積関数を持つならば，反復プログラム

$f_{loop}(x) =$ **if** $p(x)$ **then return** $b(x)$ **else**
begin $v := c(x)$; $w[1] := d(x)$;
for $i := 2$ **to** n **do** $w[i] := d(w[i-1])$;
while not $p(w[1])$ **do begin**
$v := h(v, w[1], \ldots, w[n])$;
for $i := 1$ **to** $n-1$ **do** $w[i] := w[i+1]$;
$w[n] := d(w[n])$ **end**;
return $a(v, b(w[1]), \ldots, b(w[n]))$ **end**

に変換できる．

この証明は，文献[1]と同様である．c，dが副作用を持つ場合も正しくなるように累積関数が定義されていることに注意されたい．累積関数法は，補助関数の任意性を除いてアルゴリズムである．補助関数決定のアルゴリズムは現在研究中であるが，最大不変量の法則[1]を戦略とする．本稿のシステムでは，対

$f(x) = \textbf{if } x \leq 1 \textbf{ then } 2 \textbf{ else}$
$xf(\lfloor\frac{x}{2}\rfloor) + 3xf(\lfloor\frac{x}{4}\rfloor) + 4xf(\lfloor\frac{x}{8}\rfloor) + 2xf(\lfloor\frac{x}{16}\rfloor)$

図 1: 後継関数が逆関数を持たない再帰プログラム

$U(x) = \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else if } x = 1$
$\textbf{then } 0 \textbf{ else } \quad 2V(x-1) + U(x-2)$

$V(x) = \textbf{if } x = 0 \textbf{ then } 0 \textbf{ else if } x = 1$
$\textbf{then } 1 \textbf{ else } \quad U(x-1) + V(x-2)$

図 2: ドミノ詰めパズルの相互再帰プログラム

象プログラムを数値計算（用）再帰プログラムに限ったため，補助関数及びそれに対応する累積関数をより単純な戦略である，数式データベースに対するパターンマッチにより求めた．

3 システムによる変換例

本稿のシステムによる再帰除去及び閉式化の例を示す．本稿のシステムは Common Lisp により実装され，閉式化は数式処理システムと連携して行なわれた．

3.1 再帰除去

再帰除去の例として，図 1 の再帰プログラムを取り上げる．この再帰プログラムの後継関数 ($d(x) = \lfloor\frac{x}{2}\rfloor$) は逆関数を持たないため，タプリング法では $d(x)$ の計算回数を増加させずに再帰除去を行なうのは不可能である．累積関数法では以下のように再帰除去が可能である．まず補助関数を，$a(v_1, v_2, v_3, v_4, u_1, u_2, u_3, u_4) = v_1u_1 + v_2u_2 + v_3u_3 + v_4u_4$ と設定する．そして次の 4 つの累積関数を求める．$\vec{v} = (v_1\ v_2\ v_3\ v_4)$ として，$h_1(\vec{v}, x) = xv_1 + v_2$，$h_2(\vec{v}, x) = 3xv_1 + v_3$，$h_3(\vec{v}, x) = 4xv_1 + v_4$，$h_4(\vec{v}, x) = 2xv_1$．本稿のシステムによって，実際にこれらの累積関数が数式データベースから導かれ，定理により再帰除去が行なえる．再帰除去の効果を示す計算時間の比較は表 1 のとおりである．

3.2 閉式化

閉式化の例として図 2 の相互再帰プログラムにより表現されるドミノ詰めパズル [2] の閉式化を取り上げる．この問題に対する自動的な閉式化は，我々の調査範囲内では他に見出せなかった．

累積関数は行列で表現が可能な場合がある．これを累積行列と呼ぶ．本稿の閉式化法はこの累積行列を用いる．即ち，再帰除去定理の反復部分を累積行列のべき乗により表現することで閉式化を行なう．実際，ドミノ詰めパズルの $U(x)$ を再帰呼び出しとする閉式は以下のように求まる．まず補助関数を $a(v_{11}, v_{12}, v_{21}, v_{22}, u_{11}, u_{12}, u_{21}, u_{22}) = v_{11}u_{11} + v_{12}u_{12} + v_{21}u_{21} + v_{22}u_{22}$ と設定し，後継関数を $d(x) = x - 1$ とする．$\vec{v} = (v_{11}\ v_{12}\ v_{21}\ v_{22})$ とし，次の 4 つの累積関数が求まる．$h_1(\vec{v}, x) = v_{12} + v_{21}$，$h_2(\vec{v}, x) = v_{11}$，$h_3(\vec{v}, x) = 2v_{11} + v_{22}$，$h_4(\vec{v}, x) = v_{21}$．これらの累積関数は行列を用いて表現できる．この累積行列から数式処理システムによりジョルダン標準形を求め，さらにべき乗計算を行う．この指数は $z = \min\{i | p(d^{i+1}(x)) \wedge i$ は 1 以上の整数 $\}$ を満たす z として求められるので，$z = x - 2$ が指数となり閉式化が行なえる．閉式化の効果を示す計算時間の比較は表 2 のとおりである．

表 1: 再帰プログラムと反復プログラムの計算の比較

	再帰 (msec)	反復 (msec)	改善率
$x = 500$	13.922	1.157	12.0
$x = 1000$	27.094	1.297	20.9
$x = 1500$	52.328	1.453	36.0

表 2: ドミノ詰めパズルの再帰プログラムと閉式の計算の比較

	再帰 (msec)	閉式 (msec)	改善率
$x = 15$	60.62	0.21	2.9×10^2
$x = 20$	680.47	0.15	4.5×10^3
$x = 25$	7,574.07	0.16	4.7×10^4

4 関連研究

最近でも再帰除去に関する研究が行なわれているが [4]，単一後継関数を持つ一般的な再帰プログラムからの再帰除去に関する報告は，我々の調査範囲内では他に見出すことができなかった．

従来の累積関数法は，マージソート等のリスト処理プログラム及び数値計算 (用) 再帰プログラムのいずれにも適用可能である．本稿では，数値計算 (用) 再帰プログラムのみ扱った．累積関数による，単一後継関数を持つリスト処理再帰プログラムへの再帰除去法は現在研究中である．

閉式化は数式処理システム (Macsyma 等) 単独でも可能である．しかしそれらは，単一後継関数を持つ再帰プログラムの特殊な場合（即ち，ある種の定数係数線形漸化式）のみ閉式化可能である．

5 おわりに

累積関数の拡張により，単一後継関数を持つ非線形相互再帰プログラムからの再帰除去及び閉式化のアルゴリズムを提案した．さらにこれに基づき実装を行い，数値計算（用）再帰プログラムに対する再帰除去及び閉式化システムを実現した．今後の課題としては，1) 補助関数決定アルゴリズム及び 2) 単一後継関数を持つリスト処理プログラムの累積関数法の開発があげられる．

参考文献

[1] 二村良彦, 大谷啓記: 線形再帰プログラムからの再帰除去とその実際的効果, コンピュータソフトウェア, Vol. 15, No. 3, pp. 38–49 (1998).

[2] Graham, R. L. and Knuth, D. E. and Patashnik, O.: Concrete Mathematics. p. 329 (1991)

[3] Hu, Z. and Iwasaki, H. and Takeichi, M.: Cheap Tupling Transformation, *METR 96-08*, (1996)

[4] Liu, Y. A. and Stoller, S. D.: From recursion to iteration: what are the optimizations?, *PEPM2000*, pp. 22–23 (2000).

[5] 松谷将寛: 線形再帰方程式の累積関数を用いた求解法 (2000). 早稲田大学理工学研究科. 修士論文.

[6] Pettorossi, A. and Proietti, M.: Rules and Strategies for Transforming Functional and Logic Programs, *ACM Comput. Surv.*, Vol. 28, No. 2, 1996, pp. 360-414.

[7] 坂本巨樹, 二村良彦: 分割統治法に基づくアルゴリズムの計算量自動評価の試み, 日本ソフトウェア科学会第 15 回大会 (1998 年度) 論文集, pp. 39–42.