線形再帰プログラムからの再帰除去法*

4J－11

二村良彦　大谷啓記
早稲田大学 理工学部

1. はじめに

再帰プログラムは書き易く読み易い場合が多いが，計算機で実行する際には手続き呼出しとスタック操作のためのオーバーヘッドが必要である．それ故，与えられた再帰プログラムを，スタックを使わない反復プログラムに変換するいわゆる再帰除去法の研究が，特に再帰を1度しか含まない線形再帰プログラムについて1970年中頃[3,4,7]から行われてきた．しかし，与えられた線形再帰プログラムが再帰除去できるか否かの判別法とその除去法[1,2,5,6,10]は，長い間研究されてきたが，副作用まで含む現実のプログラムに対して適用不能である．本稿では現実的な線形再帰プログラムを再帰除去する2つの系統的方法[8,9]，即ち，数学的計算に有効な”累積関数を用いた再帰除去法”及びリスト処理に有効な”擬似結合性を利用した再帰除去法”について述べる．

2. 再帰の形式

以下では式の評価規則は最左最内規則及びcall-by-value semanticsを仮定する．再帰プログラムは一般に下記の形式をしている：

$f(x) = \text{if } p(x) \text{ then } b(x)$
$\quad \text{else } a(c_0(x), f(d_1(x)), c_1(x), \ldots, f(d_n(x)), c_n(x))$

ここで，a, b, c_i 及び d_i を各々，補助関数，基底関数，制御関数及び後継関数と呼ぶ．そしてそれらは f への再帰呼出し及びその他の自由変数を含まないものとする．但し，x は変数ベクトル $x=\langle x_1,\ldots,x_m\rangle$ でも良い．

$n=1$ のとき再帰プログラムは線形であると呼ぶ．線形再帰プログラムには左線形，右線形及び中線形があるが，例えば右線形プログラムは下記の形式を有する：

$f(x) = \text{if } p(x) \text{ then } b(x) \text{ else } a(c(x), f(d(x)))$

そして左線形プログラムは下記の通りである：

$g(x) = \text{if } p(x) \text{ then } b(x) \text{ else } a(g(d(x)), c(x))$

副作用のないプログラムだけを扱う場合にはそれ等を区別する必要はなく，どの型でも例えば右線形にすることができる．プログラムに副作用がある場合，中線形は扱いにくいので，本稿では左右の線形のみを扱う．線形の内でも特に次の f_1 と f_2 の形式のものを各々末尾再帰及び単純再帰と呼ぶ：

$f_1(x) = \text{if } p(x) \text{ then } b(x) \text{ else } f_1(d(x))$

$f_2(x) = \text{if } p(x) \text{ then } b(x) \text{ else } a(f_2(d(x)))$

この2つについては簡単な再帰除去規則が知られている[7]．また一般的な線形プログラムについても，補助関数 a が結合法則を満たす（即ち結合的）ならば，容易に再帰除去が可能である[7]．その他の時は特殊な場合を除いて再帰除去規則は知られていない．再帰呼出しを本質的に2箇所以上で行う（即ち $n \geq 2$）場合の再帰除去法は動的計画法の応用が知られているが，一般性のあるのものは少ない[5]．

3. 累積関数を用いた再帰除去

ここでは補助関数 a が結合的である場合も含んだ，より適用範囲の広い再帰除去法について述べる．

定義 1: $f(x)$ は前述の右線形再帰プログラムとする．この時下記の性質を持つ関数 $h(v,u)$ を a に関する f の累積関数と呼ぶ：

「$not(p(u))$ ならば，任意の式 v に対して $a(v, f(u)) = a(h(v,u), f(d(u)))$．但し，$h$ は f への再帰呼出し及びその他の自由変数を含んではならない．」

ここで，v が例えば2次元の変数ベクトルである場合には，$h(\langle v_1, v_2\rangle, u)$ の代わりに $h(v_1, v_2, u)$ と表記しても良いことにする．また h が例えば2次元の関数ベクトル $\langle h_1, h_2\rangle$ の場合には $a(v_1, v_2, f(u)) = a(h_1(v_1,v_2,u), h_2(v_1,v_2,u), f(d(u)))$ である．3次元以上の場合や u についても同様にして扱う．

例 1: 補助関数 a が結合的である場合を考える．即ち補助関数 a は $a(x, a(y,z)) = a(a(x,y),z)$ を満たす．この時，$f(u) = a(c(x), f(d(u)))$ より $a(v, f(u)) = a(v, a(c(u), f(d(u))))$ である．従って，$h(v,u) = a(v, c(u))$ とすれば，h は f の累積関数である．

例 2: 補助関数 a が逆べき乗関数，即ち $a(x,y) = y^x$ である場合を考える．$f(u) = a(c(x), f(d(u)))$ より $a(v, f(u)) = a(v, a(c(u), f(d(u))))$ である．$a(v, f(u)) = a(h(v,u), f(d(u)))$ を解いて，$h(v,u) = c(u) * v$ を得る．従って，h は f の累積関数である．

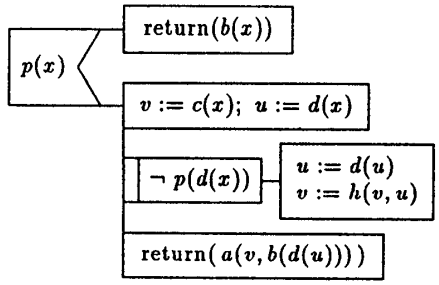
例 3: $f(x) = \text{if } p(x) \text{ then } b(x) \text{ else } c_1(x) + c_2(x) * f(d(x))$ である場合を考える．$a(v_1, v_2, u) = v_1 + v_2 * u$ であるから，$a(v_1, v_2, f(u)) = a(h_1(v_1,v_2,u), h_2(v_1,v_2,u), f(d(u)))$ を解いて，$h_1(v_1, v_2, u) = v_1 + v_2 * c_1(u)$ 及び $h_2(v_1, v_2, u) = v_2 * c_2(u)$ を得る．従って，$\langle h_1, h_2\rangle$ は f の累積関数である．

定理 1: 右線形再帰プログラム $f(x)$ が累積関数を持てば，それは下記の反復型プログラム $floop(x)$ に変換

*Recursion Removal Techniques for Linear Recursive Programs, by Yoshihiko FUTAMURA and Hirofusa OTANI, School of Science and Engineering, Waseda University.

することができる．c 及び d が副作用を持つ場合にも floop は正しく動作する（証明略）．

$floop(x) =$



定義 2: $g(x)$ は前述の左線形再帰プログラムとする．この時下記の性質を持つ関数 $h(u,v)$ を a に関する f の累積関数と呼ぶ:

「$not(p(u))$ ならば，任意の式 vに対して $a(f(u),v) = a(g(d(u)),h(u,v))$．但し，$h$ は gへの再帰呼出し及びその他の自由変数を含んではならない．」

gを反復型にした $gloop$ も $floop$ と同様に定義できる．しかしこの場合に c または d に副作用がある場合には，$c(x)$ と $d^i(x)$ の実行順序が入れ代わるので $gloop$ は正しく動作しない可能性がある（証明略）．

4. 疑似結合性を利用した再帰除去

ここではリスト処理プログラムの再帰除去に有効な方法について述べる．以下では LISP の関数を使うが，読み書きのしやすさの関係からリストを角括弧を用いて表す．例えば NIL= [], $(A\ B\ C) = [A\ B\ C]$, list(car(x))=[car(x)], cons(A,B)=[A.B] と表記する．

定義 3: a を右線形再帰プログラム $f(x)$ の補助関数とする．a に対して下記の性質を持つ関数 a'が存在する時，a を疑似結合的と呼ぶ:

「$a(x,y)$=prog($a'(x,y),x$) かつ $a(x,a(y,z))$=prog($a'(a'(x,y),z),x$)．但し，prog(u,v) は u と vを実行し，vの値を返す関数である．」

例 4: $f(x) = append(x,k)$ とおく（但し，kは任意の常リスト）．即ち $f(x)$=if x=[] then k else f(cdr(x))．この時，cons(x,y)=rplacd([x], y) より，$f(x)$ を次のように書き換えることができる:

$f(x)$ = if x = [ ] then k else rplacd([car(x)], f(cdr(x)))

ここで $a(x,y)$ = rplacd(x,y), $c(x)$ = [car(x)], $d(x)$ = cdr(x) 及び $a'(x)$ = prog(rplacd(x,y), y) とする．この時，$a(x,a(y,z))$ = prog($a'(a'(x,y),z),x$), 従って a は疑似結合的である．

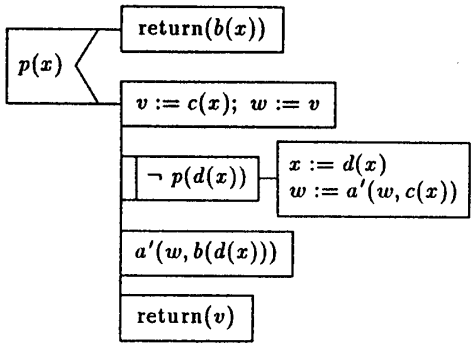
定理 2: 右線形再帰プログラム $f(x)$ を考える．補助関数 a が疑似結合的ならば，$f(x)$ を下記の反復型プログラム $floop(x)$ に変換することができる（証明略）．

定義 4: a を左線形再帰プログラム $g(x)$ の補助関数とする．a に対して下記の性質を持つ関数 a'が存在する時，a を疑似結合的と呼ぶ:

「$a(x,y)$ = prog($a'(x,y),x$) かつ $a(a(x,y),z))$ = prog($a'(x,a'(y,z)),z$)．」

gを反復型にした $gloop$ も $floop$ と同様に定義できる．しかしこの場合に定義 2 と同様の理由から $gloop$ は正しく動作しない可能性がある．

$floop(x) =$



5. おわりに

数学的計算及びリスト処理線形再帰プログラムから再帰を除去する新しい方法について報告した．

今後の課題は，下記の 2 点である．

1. 現実のプログラムに対する本方式の適応範囲の調査
2. 本方式の LISP コンパイラへの組込み

また，累積関数を系統的に求める方法については，現在検討中であり，別途報告する予定である．

参考文献

[1] Arsac, J. and Kodratoff, Y. : Some Techniques for Recursion Removal from Recursive Functions, ACM TOPLAS., Vol.4, no.2, 1982.

[2] Arsac, J. : Foundations of Programming, Academic Press, 1985.

[3] Boyer, R.S. and Moore, J.S. : Proving theorems about LISP functions, JACM, Vol.22, No.1, 1975.

[4] Burstall, R.M. and Darlington, J. : A Transformation System for Developing Recursive Programs, JACM, Vol.24, No.1, 1977.

[5] Cohen, N.H. : Eliminating Redundant Recursive Calls, ACM TOPLAS., Vol.5, No.3, 1983.

[6] Colussi, L : Recursion As an Effective Step in Program Development, ACM TOPLAS., Vol.6, No.1, 1984.

[7] Darlington, J. and Burstall, R.M. : A System which Automatically Improves Programs, Acta Informatica, Vol.6, No.1, 1976.

[8] 二村良彦，大谷啓記：オペレータの疑似結合性を利用した再帰除去法，日本ソフトウエア科学会第 11 回大会，D4-1, 1994.

[9] 二村良彦，他：プログラム変換方式，特願平 6-263439, 1994 年 10 月．

[10] Paull, M.C. : Algorithm Design, John Wiley & Sons, 1988.