線形再帰プログラムからの再帰除去法の実現

3 E − 6

坂本巨樹† 川本史生† 小西善二郎† 二村良彦††

†早稲田大学大学院 理工学研究科

††早稲田大学 理工学部

1 はじめに

我々は，線形再帰プログラム(再帰呼び出しを実質的に1個所でしか行なわないプログラム)を計算量やスペース使用量を増やさずに反復型プログラムに変換する方法(再帰除去法)について先に報告した[6].その後我々は[6]に基づき，線形再帰プログラムを能率のよい反復型プログラムに自動変換する再帰除去システムを Lisp (Allegro Common Lisp) を用いて実現した．本稿では，その実現法及び適用例について報告する．本稿で扱う再帰プログラムは f(x)=if p(x) then b(x) else a(c(x),f(d(x)))(左線形再帰)や g(x)=if p(x) then b(x) else a(g(d(x)),c(x))(右線形再帰)の形式をしたものである(a,b,c,d,x 等に関する制限の詳細については文献[6]を参照されたい)．また，本稿で扱うプログラムは線形再帰プログラムの中でも累積関数を有するものや，補助関数 a が擬似結合的[6]であるものに限る．そのような制限をつけた上でも，実用上有用な多くの例題が自動変換できることを示す．

2 再帰除去の自動化

末尾再帰プログラムからの再帰除去は単純なので幾つかの言語処理系でも組み込まれている．しかし，末尾再帰よりも複雑な線形再帰プログラムについては，補助関数 a が結合性を有する場合[5]以外には，再帰除去自動化の報告を我々は聞いていない．ある程度複雑な再帰プログラムから再帰除去をする系統的方法は多数報告されている[1,2,3,4,11]．しかし，それらの方法は完全自動化できるほどには機械化されていない．我々は，補助関数が擬似結合的である場合および累積関数を有する場合[6]に，線形再帰プログラムから自動的に再帰除去を行なう実験を行なった．完全自動化を行なうためには，より強力なプログラム変換システム(例えば一般部分計算[8])や数式処理システム[9]が必要である．将来は，一般部分計算器等[7,8,9]と相互に呼び合う形で使用することを想定している．しかし，本稿では，[6]で述べた再帰除去規則を単独で使用した場合の成功例について報告する．

3 補助関数 a が擬似結合的である場合

リスト処理でよく現れる関数 cons は結合的ではないが，擬似結合的である[6]．従って，[6]の再帰除去規則を利用した再帰除去プログラム recursion-removal を適用すれば，次のような形式で反復型プログラムが出力される．

```
> (recursion-removal
        '(if (p x)
          (b x)
          (cons (c x) (f (d x)))))
>(IF (P X) (B X) (LET* ((V (LIST (C X)))
(W V) (U X)) (DO NIL (((LAMBDA (A) (P A))
(D U))) (SETF U (D U)) (SETF W ((LAMBDA (A)
(PROGN (RPLACD W A) A)) (LIST (C U)))))
((LAMBDA (A) (RPLACD W (B A))) (D U)) V))
```

3.1 自動変換可能な関数の例

(1) pair-merge(x)

/*リスト x の隣り合う要素をマージする*/

```
pair-merge(x) = if x=[] then []
 else if atom(car(x)) then
 { if cdr(x)=[] then [[car(x)]] else
 [mer-a(car(x),cadr(x)).pair-merge(cddr(x))] }
 else{ if cdr(x)=[] then x else
 [merge(car(x),cadr(x)).pair-merge(cddr(x))] }
```

例えば pair-merge([3 1 4 1 5 9 2])=[[1 3][1 4][5 9][2]]

pair-merge([[1 3][1 4][5 9][2]])=[[1 1 3 4][2 9 5]]

(これを繰り返すことでマージソートが可能)

(2) distribute(x,y)

/*y を x の各要素に cons したリストを作る*/

```
distribute(x,y) = if x=[] then []
        else [ [y.car(x)].distribute(cdr(x).y)]
```

例えば distribute([b c d],a)=[[b.a][c.a][d.a]]

(例えば，集合の全ての部分集合を求めるときに現れる)

上記の pair-merge(x)のような 1 引数の場合だけでな

Implementation of Recursion Removal System for Linear Recursive Programs: Naoki Sakamoto, Fumio Kawamoto, Zenjiro Konisi, Yoshihiko Futamura, Graduate School of Science and Technology, Waseda University

くdistribute(x,y)のように，2引数(多引数)であっても2番目以降の引数が定数のような働きをする関数(c(x,y)で，y の中身を見ない．d(x,y),p(x,y)についても同様)ならば自動変換可能である．

つまり，この他にもappend(x,k)やrandom-list(x,n)(n以下の長さxの乱数列を作る)なども自動変換可能となる．この他にも，補助関数aがconsである場合は多数あるので，適用範囲は広い．(変換されたプログラムの性能評価は，[9]参照)

4 関数f(x)が累積関数hを有する場合

例えば補助関数 a が結合的であるならば，f(x)は累積関数 h(u,v)=a(c(u),v)を有する[6]．従って，[6]の再帰除去規則を利用した再帰除去プログラム recursion-removal を適用すれば，次のような形式で反復型プログラムが出力される．

```
> (recursion-removal
    '(if (p x)
         (b x)
         (aa (c x) (f (d x)))) 'aa)
>(IF (P X) (B X) (LET ((V (C X)) (U X)) (DO
NIL (((LAMBDA (A) (P A)) (D U))) (SETF U
(D U)) (SETF V (AA V (C U))))) ((LAMBDA (A)
(AA V (B A))) (D U))))
```

これにより例えば，a(x,y)=append(x,y)のとき，再帰除去可能である．何故ならば，append を補助関数とする線形再帰プログラムは左線形になることが多いので，特に副作用の心配が無い場合には，a(x,y)=aa(y,x)となる関数を新しく宣言してやることで右線形になる．適用できる例としては，reverse(x)(リスト x を反転させる)がある．また，再帰還元(再帰を減らすことで計算量を少なくする)を行なえるプログラムflatten(x), twist(x)などにも適用できる[7]．

5 おわりに

線形再帰プログラムの再帰除去について補時間 a が限られた性質を有する場合について，自動変換が可能であることを紹介した．そのような制限の下でも多くの例が適用可能であることを示した．今後の課題としては，次のものがあげられる．

(1)多引数関数(merge(x,y)のようにが，c(x,y)で y の中身を参照する(d(x,y),p(x,y)についても同様)場合についても紹介した方法が適用できるようにする．

(2)適用できる範囲を広げるため，今まで発見されている例については，自動再帰除去可能にする．

(3)累積関数や擬似結合性の実現関数を求めるアルゴリズムを発見する．

実際(3)の課題が解決されれば，再帰除去を自動化することは，かなり容易になる．しかし，累積関数や擬似結合性の実現関数を機械的に求めることはかなり難しいと思われる．そのためには，一般部分計算[8]や数式処理[10]を利用する必要がある．当面は，今回の報告のような個々の例を(1),(2)に関して増やしていくことによって，適用可能な再帰除去の範囲を広げ，かつ再帰除去に関する知見を深めていくことが求められる．それと同時に(3)の課題を進行していくことが理想的であると考えている．

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, pp.295-322.

[2] Burstall, R.M. and Darlington, J. : A Transformation System for Developing Recursive Programs, JACM, Vol.24, No.1, 1977, pp.44-67.

[3]Cohen, N.H. :Eliminating Redundant Recursive Calls, ACM TOPLAS., Vol.5, No.3, 1983, pp.265-299.

[4]Colussi, L : Recursion As an Effective Step in Program Development, ACM TOPLAS., Vol.6, No.1, 1984, pp.55-67.

[5]Darlington, J. and Burstall, R.M. : A System which Automatically Improves Programs, Acta Informatica, Vol.6, No.1, 1976, pp.41-60.

[6]二村,大谷 :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, 1998.

[7]二村, 大谷, 筧, 坂本, 小西:3E-07 ある種の木再帰プログラムからの再帰除去, 情報処理学会第 56 回全国大会論文集 3E-07, 1998 年 3 月.

[8] Futamura, Y., K. Nogi and A. Takano : Essence of generalized partial computation, Theoretical Computer Sciences 90, 1991, pp.61-79.

[9] 筧, 坂本, 二村:3E-08 再帰除去のゴミ回避効果, 情報処理学会第56回全国大会論文集3E-08, 1998年3月.

[10] 川本, 辰巳, 二村:3E-05 母関数を用いたプログラム変換, 情報処理学会第 56 回全国大会論文集3E-05, 1998 年 3 月.

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