ある種の木再帰プログラムからの再帰除去

3E－7 二村良彦† 大谷啓記†† 筧 一彦††† 坂本巨樹††† 小西善二郎†††
†早稲田大学理工学部情報学科 ††ＮＴＴコミュニケーションウェア(株)
†††早稲田大学理工学研究科

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，スタックを用いずしかも計算量を増加させないで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1970年中頃から行われてきた[1,2]．しかし，プログラムが再帰呼出しを実質的に2個所以上含むような非線形再帰の場合には，再帰除去は理論的に不可能な場合が多く，再帰除去の研究はあまり行なわれていない．我々は，再帰呼出しの箇所を実質的に減らすことの効果を調べた．本稿では，再帰呼出しを実質的に2個所含む非線型再帰プログラム(木再帰プログラム)から，一方の再帰を実質的に除去する方法およびその適用例（成功例と失敗例）について報告する．

2 準備

ここでは以下に述べる木再帰プログラムから再帰呼出しを実質的に一個所減らすために，文献[2]で述べられた線形再帰プログラムからの再帰除去に関する定理1を利用する．

定理1：左線形再帰プログラム g(x)= **if** p(x) **then** b(x) **else** a(g(d(x),c(x))が累積関数hを持つならば，g(x)はgtail(x)= **if** p(x) **then** b(x) **else** gtaill(d(x),c(x))と強等価である．ただし，gtaill(u,v)= **if** p(u) **then** a(b(u),v) **else** gtaill(d(u),h(u,v)).

ここで，Dg,pを関数gの定義域に含まれかつ述語pが成立しない要素の集合を表わせば，a(g(d(x)),c(x))=gtaill(d(x),c(x)) for x∈Dg,p．従ってu=d(x)，v=c(x)とおけば，次の系1が得られる．

Recursion Removal From Some Tree Recursive Programs, Yoshihiko Futamura†, Hirofusa Ohtani††, Kazuhiko Kakehi†, Naoki Sakamoto† and Zenjiro Konisi†, †School of Science and Engineering, Waseda University, ††NTT Communicationware corp.

系1：a(g(u),v)=gtaill(u,v) for u∈d(Dg,p) and v∈c(Dg,p).

3 木再帰プログラムの再帰除去

木再帰プログラムの一般形は大雑把には次の通りである：g(x)= **if** p(x) **then** b(x) **else** a(g(d1(x)),g(d2(x)),c(x)).ただし，aの引数の順序，a,b,c,d1,d2,x等に関する制限については文献[2]を参照されたい．本稿の以下の部分では，木再帰プログラムのうちで次の2つのサブクラスについて考える．

3.1 補助関数が結合的な単純木再帰プログラム

次のような形式をした木再帰プログラムを単純木再帰と呼ぶ：g(x)= **if** p(x) **then** b(x) **else** a(g(d1(x)),g(d2(x)))．ここでは補助関数aが結合的な単純木再帰プログラムについて考える．

d(x)=d1(x)，c(x)=g(d2(x))とおき，定理1を適用すると下記のgtail(x)が得られる．ただし，aが結合的なので定理1中のh(u,v)はa(c(u),v)である：gtail(x)= **if** p(x) **then** b(x) **else** gtaill(d1(x),g(d2(x)))．ただし，gtaill(u,v)= **if** p(u) **then** a(b(u),v) **else** gtaill(d(u),a(g(d2(u)),v)).

系1より，上記gtaill中のuとvについてu∈d1(Dg,p) and v∈g(d2(Dg,p))ならば，a(g(u),v)=gtaill(u,v)となりgが消去できる．従って次の定理2が成立する．

定理2：与えられた単純木再帰プログラムgにおいてその補助関数aが結合的であり，かつ上記gtaill中のuとvについて次の3条件が成立するものとする：(1)u∈d1(Dg,p)，(2)v∈g(d2(Dg,p))，(3)¬p(u) and d2(u)∈d1(Dg,p)．この時gは下記のgtailに変換できる：gtail(x)=**if** p(x) **then** b(x) **else** gtaill(d1(x),gtail(d2(x)))．ただしgtaill(u,v) = **if** p(u) **then** a(b(u),v) **else** gtaill(d1(u), gtaill(d2(u),v)).

ここで，gtaillの中に現れる2つの再帰呼出しの一

方は末尾再帰となり，実質的には再帰呼出しになっていないことに注意されたい．

例1：関数flattenへの適用

flatten(x)=**if** atom(x) **then** (**if** null(x) [] **else** [x])**else** append(flatten(car(x)),flatten(cdr(x))).

ここで，a(x,y)=append(x,y)，b(x)=**if** null(x) **then** [] **else** [x]，d_1(x)=car(x),d_2(x)=cdr(x)，p(x)=atom(x)，g(x)=flatten(x)とおけば，定理2の条件(1)～(3)が成立する．従って，関数flattenには定理2が適用可能であり，その結果，consの実行回数がO(n log n)からO(n)に減少する．ただしnは，xに含まれるセル(ノード)の個数である．

例2：紙面の都合で詳しい説明は省略するが，mergesortプログラムに対しても，flatten同様，定理2が適用可能である（mergeが結合的であるから）．しかし，gtaill が停止する時にmergeが実行され，そこでO(n)時間要する．従って，全体の実行時間は，O(n log n)から$O(n^2)$に悪化する可能性がある．何故ならば，この変換によりmergesortが挿入ソートに変わってしまうのである．このように，分割統治法の効果により O(n log n)で計算できていた木再帰プログラムから再帰除去することにより，その性能を劣化させる場合もありうる．

3.2 補助関数が結合的な半単純木再帰プログラム

次のような形式をした木再帰プログラムを半単純木再帰と呼ぶ：g(x)= **if** p(x) **then** b(x) **else** a(g(d_1(x)),e(g(d_2(x)),x)).ここでは補助関数aが結合的な半単純木再帰プログラムについて考える．d(x)=d_1(x)，c(x)=e(g(d_2(x)),x)とおき，定理1を適用すると下記のgtail(x)が得られる：gtail(x)= **if** p(x) **then** b(x) **else** gtaill(d(x),e(g(d_2(x)),x))．ただし，gtaill(u,v) =**if** p(u) **then** a(b(u),v) **else** gtaill(d(u)，a(e(g(d2(u)),u),v)).成立する．

定理3：与えられた半単純木再帰プログラムgにおいてその補助関数aが結合的でありかつ，上記gtaill中のuとvについて次の4条件が成立するものとする：(1) a(e(g(d_2(u)),u),v)= e'(a(g(d_2(u)),v),u)なるe'が存在する，(2)u∈d_1(Dg,p)，(3)v∈g(d_2(Dg,p))，(4) ¬p(u) and d_2(u) ∈d_1(Dg,p)．この時gは次のgtailに変換可能である：gtail(x)= **if** p(x) **then** b(x) **else** gtaill(d_1(x)，e(gtail(d_2(x)),x))．ただし gtaill(u,v)= **if** p(u) **then** a(b(u),v) **else** gtaill(d_1(u)，e'(gtaill(d_2(u),v),u)).

例3：関数twistへの適用例

twist(x)= **if** atom(x) **then** x **else** append(twist(cdr(x))，[twist(car(x))])．ただし，appendは次のように定義する：append(x,y)= **if** null(y) **then** x **else** (**if** null(x) **then** y **else** cons(car(x)，append(cdr(x),y)))).

e(x,y)=list(x)=[x]，e'(x,y)=cons(x,y)と置けば，append(e(x,y),z)=append([x],z)=cons(x,z)=e'(x,z)．従って，append([g(u)],v)=e'(g(u),v)．従って，定理3より，twistは下記のように再帰除去可能である．twisttail(x)= **if** atom(x) **then** x **else** twisttaill(cdr(x),[twisttail(car(x))])．ただし，twisttaill(u,v)= **if** atom(u) **then** v **else** twisttaill (cdr(u)，cons(twisttaill(car(u),v),u)).

appendの除去が行なわれたためconsの実行回数がO(n log n)からO(n)に減少している．ただしnは，xに含まれるセル(ノード)の個数である．例1と同じく，変換によりプログラムの性能が飛躍的に向上している．

4 おわりに

補助関数が結合的である木再帰プログラムに線形再帰除去法を適用することにより，下記の2点が実現できた：(1) 2つの再帰呼出しのうちの一方の末尾再帰化，(2) 補助関数の結合性の利用によるプログラム計算量の減少．

また，上記(2)が常に可能とは限らないことも示した(例2)．本稿で提案した方法が有効な場合の判別および再帰除去法の自動化が今後の課題である．

参考文献

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

[2] 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.