関数型プログラムにおけるプログラム変換の研究

佐賀 正芳

北陸先端科学技術大学院大学 情報科学研究科

1998 年 2 月 13 日

キーワード： 関数型プログラム, プログラム変換, 切り払い (deforestation).

関数型プログラムでは、基本的な関数を組み合わせてプログラムを記述するのが一般的である。このことは、機能のモジュール化を高めると共に、人が理解しやすい関数記述に貢献する。しかし、多数の関数を組み合わせることにより、関数間の情報の受け渡し役となる中間的なデータ構造が実行時に生成される。この中間データ構造は、関数実行後の値に直接反映されるものではなく、一時的に生成され最終的には不要となるものであるから、実行時の効率を阻害する要因となっている。

このような中間データ構造を除去すべく、Wadler(1990) は切り払い (deforestation) という関数型プログラムのプログラム変換を提案した。切り払いは、プログラムに 7 つの規則を繰り返し適用することで、中間データ構造の除去を図る。しかし、この手法は再帰関数を無制限に展開させないようにするため、一度展開した関数呼び出しの形をメモしておき、同じ形をした関数呼び出しの形が現れたら新しい関数を定義するという繁雑な処理を必要とした。これに対し Gill ら (1993) は、リストを扱う基本的な関数である $foldr$を用いてリストに関する再帰関数を抽象化し、抽象化された関数だけを切り払いの対象とすることで、メモ化を不要にした。だが再帰関数は抽象化されていることが限定であったこと、リストを高速反転させる関数に対応していないという問題がある。また、従来の研究の多くは遅延評価 (lazy evaluation) の言語上でなされ、先行評価 (strict evaluation) の言語上ではほとんど研究されていない。

そこで本研究では、値呼び (call-by-value) 簡約の言語において、再帰関数の抽象化なしの部分計算的切り払いという新しい変換方法をを提案する。さらに、実行結果に直接反映される値を引数として渡すことで計算を行なう再帰関数についても、ある条件の下では切り払いが可能であることを示す。

部分計算的切り払いの処理の流れは、関数定義の分類、切り払い実行の 2 つに大別される。

関数定義の分類では、関数をその性質に基づき逐次形関数、一括形関数、その他の関数に分類する。前 2 者はいずれも出力の型が明示的な再帰関数である。これら 2 関数の特徴を以下に示す。

- 逐次形関数…入力データの要素に対して等しく処理がなされ、関数呼出毎に最終的な値の一部が形成される。
- 一括形関数…入力データの要素に対して等しく処理がなされ、その値は最終的な値の一部ではあるものの、データ構造の転換を伴い引数を介して自己呼び出しに使用される。最終的な値はその引数の値そのものとなる。

部分計算的切り払いでは、変換対象となる項を主としてこれらの関数の組合せからなる項に絞る。よって組合せ別の 4 通りの規則を用いて新関数を定義し、それを用いて変換対象である項を書き換える。新関数を導出する際の基本方針は次の通りである。

- 内側の関数を展開し、各分岐の右辺を外側の関数に適用させる。
- 簡約可能な箇所は簡約する。
- 変換対象の関数合成とマッチする項は、新関数を用いて書き換える。

これら基本方針に加え、各組合せから行なうべき処理を抽出し、新関数あるいは変換後の項に採り入れることで切り払いを実行する。切り払い実行後の項は、再帰的に切り払い手続き処理がなされる。

この切り払い手続きの停止性、変換前後の項の等価性は示された。効率性については、実装した結果、複数回のデータ参照が１回のデータ参照で済むようになったことから、計算ステップ数の減少が認められた。但し、効率は入力となるデータの大小に左右され、一般的な効率の向上率といった指標を得るには至らなかった。

この部分計算的切り払いは、入出力間の型が統一されていること、関数定義で出力の型が明示的であることという条件があった。そのため、最終的な値に直接反映する引数を介して再帰を行なう関数が扱えるようになった反面、Gill らの手法では扱えていた、帰納分岐時の出力の型が明示的でない関数は扱えない。また、引数を変化させてデータの要素毎に異なる処理を施すような関数も扱えない。

そこで、出力の型が明示的でなく、データの要素に対する処理が均一でない場合も取り扱えるような部分計算的切り払いの拡張を試みた。しかし、型が明示的であるから可能だった行なうべき処理の抽出が困難となったこと、データの要素に対する処理が均等でない場合、一括形と逐次形の組合せでは要素に対する処理の抽出が困難であることが判明した。この解決策としては、関数定義の分類に際して、出力の型が明示的か、データの要素処理が均一かといった視点も加えて分類すべきである。そして、その分類毎に規則を作り切り払いが行なえるような項は行なうという解決策が挙げられる。但し、このような分類は複雑である。

以上、値呼び簡約の言語における、プログラムの抽象化なしの部分計算的切り払いという新しい変換手法を提案し、その有効性と問題点、その解決策について検討した。今後、

部分計算的切り払いをさらに発展させるには、再帰関数の分類基準について研究がなされなければならない。そうすれば、データ構造の転換を伴う異なる型間のスムーズな切り払いが可能になる。また、値呼び簡約の言語上ではほとんど研究されていないので、簡約規則の違いという視点からの研究は、今後の研究方向として興味深い課題である。