論　文　の　内　容　の　要　旨

論文題目：Fusion Transformation of Functional Programs
(関数プログラムの融合変換)

氏　名：　高野 明彦

過去30年間、プログラム生産性向上の研究は、構造化プログラミング、抽象データ型、オブジェクト指向プログラミングなどを産み出した。これらの研究成果は実際のプログラム生産現場にも適用されて一定の成果を上げているが、生産性の向上率は高々数倍程度に止まっている。その原因は、どのようなプログラミング技法や言語を用いても、仕様に基づくプログラム作成の本質的な部分が人手に任されている限り、最終的に得られるプログラムの信頼性を確保できないことにある。

この困難を克服してソフトウェア生産を工業化し、プログラムの生産性を飛躍的に向上させるためには、信頼性の保証された共通ソフトウェア部品の整備と、それらを組み合せて高信頼で高性能のソフトウェアを作成する方法の開発が必須である。実際、抽象データ型やオブジェクト指向プログラミングでは、仕様を満たす信頼性の高い部品プログラムを整理・保存しておき、それらを組み合せてプログラムを作成している。しかし現状では、

プログラムの実行性能を確保するためにプログラマが職人的な勘を頼りに部品に安易な変更を加える場合も多く、その結果著しい信頼性低下を招いている。人手に頼らず自動的に性能を確保する技術が求められている。

本研究の目的は、数学的取扱いに適した関数プログラミング言語を用いて、正しさの保証された汎用部品プログラムを自由に組み合せて高信頼プログラムを構築する、新しいプログラミング方法論を確立することにある。そこでは、プログラムは次のように作成される。

1. 汎用部品ライブラリには、部品仕様を満たすことが数学的に証明された正しい部品プログラムを、その仕様と共に保存しておく。

2. 与えられたプログラムの仕様を部品仕様の組み合せにより表現し、その表現の正当性(首尾一貫性)を数学的に証明する。

3. 部品仕様の組み合せにしたがって対応する部品プログラムを組み合せることにより、与えられた仕様を満たすことが数学的に保証されたプログラムを得る。

4. 上記プログラムを機能的には等価で、より効率の良いプログラムへ自動的に変換し、要求される性能を満足するプログラムを得る。

この方法論でアプリケーションごとに人手が本質的に介在するのは、上記2で部品仕様の組み合せによる表現を作成する過程だけである。その結果の正しさ(首尾一貫性)は、機械の補助を受けながら数学的厳密さで確認できる。

本論文では、この方法論の実用化における重要な課題である、上記4の自動変換法にアプローチしている。「プログラム融合変換」とは、汎用部品の組み合せを、利用の文脈(組み合せ方)に応じて効率の良い形へ特殊化する変換をいう。ここでは次の2種類の方法を提案している。

1. [一般部分計算法]

   従来の解釈依存部分計算法が、入力データが静的／動的かの情報のみ

に基づいて特殊化を行うのに対し、一般部分計算法は、プログラムに内在する条件分岐構造等の文脈情報を最大限に利用して高効率の部品へ簡約・変換する。さらに、文脈情報の制約解消に“Disunification”と呼ばれる手法を用いることにより、上記変換法を改良している。

2. ［構成的アルゴリズム論に基づくプログラム融合変換］
プログラム部品の表現を数学的な「良構造」に限定することにより、その構造の数学的性質を利用した自動変換が可能となる。具体的には、構成的アルゴリズム論に基づくHylomorphismを部品表現の基本形に用い、その数学的性質を利用して融合変換を定式化した。特に、ここで提案したHylomorphismの3つ組表現は、プログラム構造の表現方法として十分な表現能力を備えており、プログラム変換のための中間表現に好適である。2つのHylomorphismの合成が1つに融合できるための十分条件を酸性雨定理と呼ばれる2種類の融合変換規則として与え、それに基づいて融合変換法を定式化している。

一般部分計算法は、従来の解釈依存部分計算法に比べてはるかに強力な変換法であるものの、その目標は「どのようなプログラムが与えられても、その効率を良くすること」であるため、入力プログラムが特定の構造を持つことを仮定する手法は避けられてきた。その結果、我々が目標とするプログラミング方法論における、部品プログラムの組み合せとして記述されたプログラムの性能を確保するための変換としては、しばしば不十分な変換能力しか得られなかった。

この問題を解くカギは、プログラム部品の表現を数学的な「良構造」に限定することと、自動変換でその構造の数学的性質を利用することにあると考え、オランダ国立研究所ＣＷＩと英国Oxford大学を中心に研究されてきた構成的アルゴリズム論(Constructive Algorithmics)が有効な理論的基礎を与えることを見出した。

中間データ構造を除去するプログラム変換としては、“Deforestation”が

良く知られている。この変換は任意の木構造データを除去できるものの、変換対象となるプログラムの形が Treeless Program と呼ばれる一階プログラムの特殊な場合に限られていることと、変換法そのものが大域的な解析を必要とするため効率が悪く実用的ではなかった。この欠点を補う実用的な変換法として“Shortcut Deforestation”が提案され、一部のコンパイラの最適化部で実装されているが、除去の対象となるデータ構造はリスト構造に限られていた。本研究で提案した上記2つの変換法は、このそれぞれを拡張したものと考えることができる。

すなわち、一般部分計算法は、Deforestation と同様の大域的な解析を必要とする Online 変換法であるが、分岐条件が不成立だったというような否定情報も解析に利用することで、より強力な変換を可能としている。例えば、素朴な文字列検索プログラムから KMP-like な効率の良いプログラムを導出できる。また、酸性雨定理は、Shortcut Deforestation の foldr-build 規則を構成的アルゴリズム論により完全に説明し、任意の代数データ型へ拡張している。それに基づく融合変換は、リストを含む任意の代数データ型の中間データ構造を除去可能であり、また、foldr-build 規則と数学的に双対な場合を含む拡張にもなっている。

さらに、本論文では、[文脈保存条件を用いた並列プログラムの導出法] を提案している。上記で導入した高階関数によって表現された逐次プログラムを、等価な並列プログラムへ系統的に変換する方法を示している。高階関数によって表される準同型写像のうち、どのような関数に本変換が適用可能であるかを、再帰呼出しが満足すべき文脈保存条件として定式化している。これにより、4種類の逐次プログラムスキームについて、等価な並列プログラムスキームを与えている。

以上のように、本論文で提案したプログラム変換法は、既存の実用的なプログラム変換法を大幅に改良したというだけでなく、構成的アルゴリズム論という理論分野とプログラム変換という実用的分野の関連性を初めて明らかにし、その有効性を示したものである。