依存グラフの変形に基づくコード最適化の統一的枠組み

滝本宗宏

論文の内容の要旨

近年、計算機ハードウェアの高性能化とともに、その上で動作するソフトウェアは大規模化と複雑化の一途をたどっている。そのようなソフトウェアを効率良く実行するために、プログラミング言語処理系においては、ソースプログラムから目的コードへの翻訳過程でコード最適化と呼ばれる処理が不可欠の存在になっている。実行効率の良いコードの生成には、複雑な解析を必要とするので、従来手法の集成では、コンパイル時間の増大を招き、プログラムの生産性を低下させる一因となる。

本研究では、最適化の効果及びアルゴリズムの効率において、従来技法よりも優れたコード移動に基づく最適化手法を提案する。本手法では、制御フローグラフによるプログラム表現以外に、変数間の値の依存関係を表す依存グラフを用いる。コード移動に伴う式の等価性の保存は、このグラフに対する変形規則の適用によって保証される。コード移動によって明らかになる等価式、不要な代入、定数式など、最適化の対象になる式は、依存グラフの変形によって見つけることができる。これらの情報は、グラフ構造を利用して高速で精密な計算が可能であり、従来手法では検出できなかった冗長な式および不要な式の検出を可能にする。

依存グラフへの変形規則の適用は、あるプログラム点における式の計算を別のプログラム点に移した場合の、移動先での等価式の依存構造を求めることに等しい。コード移動を行う際の上述の式の解析は、各プログラム点への式の移動に対応する依存グラフを制御フローグラフに関連付けることによって、データフロー解析として定式化することができる。この定式化は、コード移動に関して、理論的な基盤に基づく一般的な枠組を与え、高速で簡潔な最適化の実現を保証する。

コード移動においては、依存関係にある式の間で計算順序を変更しないことが前提となる。制御フロー上で式を前方または後方に移動させる場合、その移動は、それぞれ依存先または依存元に当たる式によってブロックされ、1回のコード移動では、移動できる範囲が限定される。その範囲を表すプログラム点は、簡単なデータフロー解析によって求めることができる。しかし、この方法では、ブロックをしていた式自身も移動することがあるので、すべての可能な移動を求めるには、データフロー解析を繰り返し適用する必要があり、非効率的である。

この他に、依存グラフを用いて、依存性を保存したまま各式を順次移動させていく方法が提案されている。依存グラフは、各変数に対する定義と使用との関係を明示的に表現するので、依存関係を保存しながらコード移動を行うことによって、可能な移動を1回の処理で行うことができる。依存グラフは、複数の式の間での計算の構造を表現しているので、変数名によらず、計算構造から等

価な式を検出できるという性質ももつ。ただし、依存グラフを用いる場合には、制御フローの合流点における依存構造の等価変換を考慮しないと、コード移動の範囲が限定され、最適化の効果が低減する可能性がある。

本論文では、まず依存グラフに対する等価変換の変形規則を導入することによって、上述の結合点に関する制約が除去できることを示す。それらの変形規則は、依存グラフと静的単一代入形式のプログラム表現とが対応する性質を利用して、結合点の前後における等価式の表現の差異から導くことができる。さらに、依存グラフとその変形規則を用いることによって、従来から行われてきた等価式の検出と定数畳込みが精密にかつ効率良く行えることを示す。次に、コード移動に基づく代表的な最適化手法である部分冗長除去と部分不要コード除去について、最適化効果の向上と最適化アルゴリズムの高速化を目的として、新たなプログラム表現を提案し、それに基づくデータフロー解析法を述べる。依存グラフを利用する方法では、コード移動による式の等価性を維持するために、変形前後のグラフを等価グラフとして保持しておく必要がある。そのグラフは、依存グラフの変形前後における等価な計算の対応関係を1つの節で表現することによって構成できる。このグラフを拡張値グラフと呼ぶ。拡張値グラフは、各プログラム点における計算の依存構造を表現すると同時に、その構造と等価な依存構造を対応付けているので、制御フローグラフの各節に存在するスロットを拡張値グラフの各節に関連付けることによって、拡張値グラフの依存辺を通して移動をブロックする情報を依存先あるいは依存元に伝播させることができる。このことから、従来、保守的な評価という立場から、元の式あるいは文の出現位置で与えなければならかった計算順序制約をデータフロー方程式に組み込むことができる。この特徴は、可能な移動を求めるために従来、繰返し適用してきたデータフロー解析が1回で済むことを意味する。さらに、拡張値グラフを作成する過程では、等価式の検出と定数畳込みの両方を同時に行うことができる。

本手法のコード最適化結果に対する有効性を確かめるために、ここに提案した手法を用いたコード最適化を2種類実装したCコンパイラを作成し、10個の評価プログラムを用いて、生成されるコードの実行効率を評価した。その結果、従来法に比べ、実行効率の良いコードを生成できることが示された。

以上