7B－9

拡張値グラフに基づくコード最適化 — 定数畳込みと強さの軽減*

滝本宗宏†　　原田賢一‡

慶應義塾大学理工学部§

email: mune@hara.cs.keio.ac.jp

1 はじめに

コンパイラのコード最適化において，近年コード移動による手法が多く研究されてきた．その多くは移動によって，同じ字面の式を発見し，冗長なものを取り除くという手法であり，代表的なものに**部分冗長除去** [1] がある．これに対し，以前に提案を行った**拡張値グラフ** [2] は，各計算点に式を移動した際の構造をグラフ表現を用いて保持することによって，部分冗長を効率的かつ効果的に字面によらず，その依存構造から発見し，取り除くことを可能にした．

本発表では，拡張値グラフが移動先の式の構造を保持することを用いて，従来の**定数畳込みと強さの軽減**を拡張する方法を提案する．この手法によって，従来，原始プログラムのままの式の依存構造に対して適用した定数畳込みや強さの軽減は，新たな定数畳込みや強さ軽減の候補を生成しながら適用できるようになる．

以降で，本研究の基礎となるデータ構造である拡張値グラフの概観を述べ，拡張値グラフを使った部分冗長除去について説明する．その後，応用として，定数畳込みと強さの軽減の適用について述べる．

2 拡張値グラフ

式の等価関係を解析する場合，その字面に基づいて解析を単純に行うと，多くの等価な式を見落す可能性がある．それは，式のオペランドが変数である場合，変数名が違っていても同じ値を保持していることがあるからである．字面の影響を受けずに等価関係を発見するためには，式の依存グラフが必要になる．このような依存グラフの中で，大域的に依存を表現するものに**値グラフ** [3] がある．値グラフは，原始プログラムにおける式の依存を表現するのには優れているが，式を移動させた場合に変化する依存構造を保持することができない．この点で，値グラフを構造の変化を保持できるように拡張したものが拡張値グラフである．

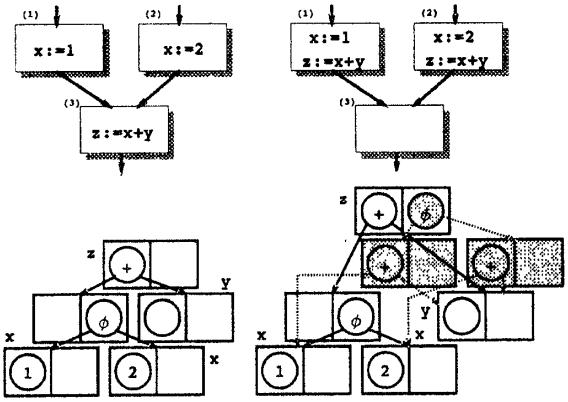

図 1: 拡張値グラフ

拡張値グラフは最大で２つまでの副節をもつ主節と，副節から主節へ出る有向辺によって構成される．左側の副節は演算子と定数をラベルにもち，右の副節は複数のパスから届く定義を結合する働きをする仮想関数の ϕ- 関数 [4] をラベルにもつ．拡張値グラフは，原始プログラムの依存構造から初期グラフを作成し，式の移動によって起こる依存構造変化によって変形を行なう．**図1**の左側は，原始プログラムと，原始プログラムから作成した初期グラフである．右側は基本ブロック (3) の式を２つの先行節に移動した場合のコードとその拡張値グラフである．移動前後で，依存構造が異なる２つの副節は，同じ値を表現するものとして同じ主節で表現される．この拡張値グラフの各主節にその節の依存構造で移動できる基本ブロックを記録しておくことによって，各計算点での式構造を統合的に表現できる．

この拡張値グラフに対して，オペランドに相当する節と演算子ラベルの組を使ってハッシュ法による等価発見

*Code Optmization based on Extended Value Graph — Constant Folding and Strength Reduction

†Munehiro Takimoto

‡Kenichi Harada

§Faculty of Science and Technology, Keio University

をボトムアップに適用すると，等価な節が1つになった最終的な拡張値グラフが得られる．

3 拡張値グラフに基づく部分冗長除去

拡張値グラフは，左の副節から出る辺が式の依存関係を表現し，右の副節は条件によって等価になり得る式（条件等価な式）をつなぎ合わせた等価関係を表している．ここで，条件によらず等価になる式（無条件等価）は同じ節になる．各拡張値グラフ節で表現される基本ブロックごとにデータフロー用の節を用意して制御フローグラフに沿ってつなぎ合わせる．この操作によって無条件等価な式をつないだ無条件等価グラフが作成できる．このグラフを，右の副節から出る辺の等価情報を利用して，辺でつながった節が表現する無条件等価グラフどうしをつなぐと，各値の流れを意味する**値流れグラフ**ができる．

この値流れグラフ上で部分冗長除去のデータフロー解析を行うことによって，効果的な部分冗長除去が可能になる．

4 定数畳込み

図2の左側の基本ブロック (3) にある式はそのままでは定数畳込みを適用することができない．しかし，**図2**の右側のように，この式を2つの先行節に移動すると，畳込みが可能になる．このような状況に対して，拡張値グラフを次のように利用することができる．

拡張値グラフは各計算点の依存構造をもっているので，移動によって新たに畳込みが可能になる部分は畳み込んでおくことができる．この変形を行った拡張値グラフを基に値流れグラフを作成し，その上でデータフロー解析を行う．コストの低い構造になる部分で式の挿入が起こるように，データフロー方程式に対して条件を加えておくことによって，新たに畳込みが可能になる計算点に確実に定数の形で式を挿入することができる．

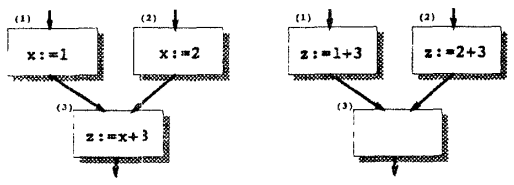

図 2: 移動によって可能になる定数畳込み

5 強さの軽減

拡張値グラフによる各計算点の依存構造と部分冗長除去の過程において，すべての式を最も初期の位置に移動することから，移動によってある式が利用可能になるかどうかを知ることができる．

この情報を用いると，大域的に演算子の強さの軽減を行うことができる．例えば，依存構造で $a*(b+c)$ となる計算は，代数的変換によって $a*b+a*c$ とすることができる．ここで，$a*b$ と $a*c$ が利用可能式であれば，$*$ の演算子を $+$ の演算子に軽減することができる．

また，特に代数的変換のことだけとってみても，本手法の定数畳込みとの組合せで大きな効果が得られる可能性を含んでいる．

6 まとめ

拡張値グラフの各計算点の依存構造を保持する性質を用いて，部分冗長除去に定数畳込みと強さの軽減を効果的に組み合わせる方法を示した．

参考文献

[1] Alpern, B., Wegman, M. N. and Zadeck, F. K.: Detecting Equality of Variables in Programs, *POPL*, ACM, pp. 1–11 (1988).

[2] Morel, E. and Renvoise, C.: Global Optimization by Supperssion of Partial Redundancies, *Comm.ACM*, Vol. 22, No. 2, pp. 96–103 (1979).

[3] 滝本宗宏, 原田賢一: ϕ-関数移動による効率的な部分冗長計算除去, プログラミング－言語・基礎・実線－研究会報告, No. 20-3, pp. 21–30 (1995).

[4] Sreedhar, V. C. and Gao, G. R.: A Linear Time Algorithm for Placing ϕ-Nodes, *POPL*, ACM, pp. 62–73 (1995).