5L-1

スピルコストを考慮した部分冗長除去

澄川 靖信 *　滝本宗宏 *
東京理科大学 *

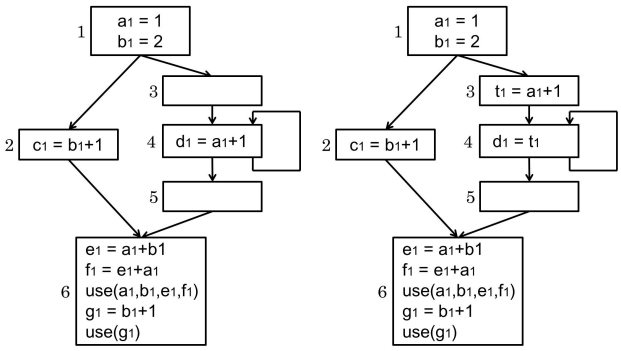
1　はじめに

コンパイラのコード最適化の 1 つである**部分冗長除去**（Partial Redundancy Elimination，以降 PRE と呼ぶ）は，部分冗長な式の除去と，ループ不変式のループ外への移動を同時に行う強力な手法である．PRE は，一部の実行経路で冗長な式を，すべての実行経路で冗長な式となるように，新しく式を挿入し冗長性を除去する．式の挿入は，元のプログラムの実行点よりも早く実行されるプログラム点で行われるので，変数の生存期間を拡張し，**レジスタスピル** [1] が生じる機会を増加させる可能性がある．レジスタスピルが生じると，変数の定義の直後にストア命令，使用の前にロード命令が挿入されるので，実行経路の長さを増加させ，実行効率を低減させる可能性がある．

本研究では，ループ不変式ではない，演算コストの低い式の冗長性を除去しないことによって，レジスタスピルを生じる回数を低減させ，実行効率の低減を防ぐ手法を提案する．

例：図 1(a) のプログラムに本手法を適用した結果が図 1(b) である．(a) の**制御フローグラフ**（Control Flow Graph，以降 CFG と呼ぶ）の節 4 の式 a_1+1 はループ不変式なので，PRE を適用することによって，(b) のプログラムのように，ループ外へ移動できる．また，節 6 の式 b_1+1 は部分冗長な式なので，新しい式 t_2= b_1+1 を節 5 の出口に挿入することによって，全冗長な式となり，除去できる．しかしながら，新しく式を挿入した結果，節 6 の use 文の直前で生きている変数の数が増加し，5 つとなるので，もし使用できるレジスタが 4 つであった場合，レジスタスピルが生じる．

本手法では，ループ不変式ではない加算や減算，ビットシフト演算といった比較的演算コストが低い式の冗長性は除去しない．b_1+1 はループ不変式ではないので，t_2= b_1+1 を挿入せず，冗長性を除去しない．結果として，use 文の直前で生きている変数の数は 4 つなので，上記のレジスタスピルを防ぐことができる．∎



(a) 元のプログラム　　(b) 本手法を適用したプログラム

図 1　本稿で使用するプログラム例

しかしながら，本手法のように冗長性を除去しない場合，**副次的効果**（second order effects）として知られている，ある式の冗長性を除去した後にコピー伝播を適用し，その式に依存する後続の式の字面を変更することによって明らかになる冗長性を検出できなくなる問題がある．

著者らは, 大域値番号付け（Global Value Numbering, 以降 GVN と呼ぶ）をあらかじめ適用し，字面に依存しない解析を可能とする手法として，**効率的な要求駆動型 PRE**（Effective Demand Driven PRE, 以降 EDDPRE と呼ぶ）を提案した [3]．EDDPRE は，GVN によって生成された，式の値ごとに生成される値番号を利用し，各式の値番号の一致性によって，冗長性を除去する．

副次的効果は，字面に基づいた解析を行う場合はすべての冗長性を除去しなければならないので，本手法は EDDPRE を拡張し，字面に依存しない解析を行うことによって，冗長性を除去せずに副次的効果を反映する．

以降の構成は次のとおりである．第 2 節で本稿で述べるプログラムの表現形式を定義し，第 3 節で EDDPRE を概説し，第 4 節で本手法について述べる．第 5 節で関連研究について述べ，第 6 節でまとめる．

Partial Redundancy Elimination Considering Spill Costs
*Sumikawa Yasunobu, Tokyo University of Science
*Takimoto Munehiro, Tokyo University of Science

2 入力プログラム

CFG は，途中に分岐や合流が無い命令列である**基本ブロック**を表す節の集合 **N**，節間の制御の様子を表す辺の集合 **E**，特別な節である開始節 **s**，終了節 **e** からなる．

本手法は，コード移動に基づいた手法なので，コード移動を阻害する可能性がある辺として知られている**クリティカル辺**（critical edge）はあらかじめ除去されているものとする．

本手法が仮定するプログラムは，静的単一代入形式の 3 番地コードに変換されているものとする．

3 効率的な要求駆動型部分冗長除去

EDDPRE は，あらかじめ GVN を適用し，各式の値番号を利用する質問伝播と呼ばれる手法を用いて冗長性を除去する．質問伝播は，CFG を**トポロジカルソート順序**で訪問し，値番号 val である式の出現ごとに「val は冗長か」というクエリを制御とは逆向きに伝播する．クエリは，同じ値番号を持つ式が出現したときに解 $true$ を返し，実行経路上に同じ値番号が存在しなければ $false$ を返す．すなわち，節 n で複数の解が得られ，$true$ と $false$ の両方が含まれるとき，クエリの式は n で部分冗長である．$false$ を得た節が**安全性**を満たすならば新しく式を挿入し，冗長性を除去する．また，クエリが，クエリを生成した節に到達したとき，**自己生成解**が $true$ となる．自己生成解が $true$ であり，かつクエリの解 $false$ を得たとき，$false$ を得た節の安全性を検査せずに式を挿入することで，ループ不変式だけを投機的にループ外へ移動させる．

4 スピルコストを考慮した部分冗長除去

本節で提案手法である**スピルコストを考慮した PRE** について説明する．本手法は，比較的演算コストが低い式の冗長性を除去しないことによって，レジスタスピルを生じる回数を低減し，実行効率の低減を防ぐ．本手法では，EDDPRE を適用した後，クエリの解が $true$ であり，かつ，自己生成解が $true$，または式の演算子が，次の演算子ではないときに冗長性を除去する．

1. 加算，減算
2. 右ビットシフト，左ビットシフト
3. ロード命令，ストア命令を除いた一項演算子

自己生成解の $true$ が得られたとき，クエリを生成した式はループ不変式であることがわかる．一般的に，ループの実行回数は大きいと考えられるので，ループ不変式はループ外へ移動させる．

上記の演算子に関する条件を満たす式の中には除去されない冗長な式が存在するが，コンパイラの後の処理である命令選択フェーズにおいて，冗長な式が他の式の一部となるように合成され，合成後の命令が 1 サイクルで実行可能となることや，無用コード除去を適用することによって除去される可能性がある．すなわち，上記の条件に該当する式の冗長性も，他の最適化を組み合わせることによって，レジスタ圧力を増加させることなく，除去できる可能性がある．

5 関連研究

Knoop らは，式を挿入する節の中で最も生存期間が短い節を求める手法を提案した [2]．しかしながら，すべての式の冗長性を除去する手法なので，レジスタスピルを生じる回数が増加する傾向がある．

6 まとめ

本稿では，比較的演算コストが低い式の冗長性を除去しないことによって，レジスタスピルを生じる回数を抑制し，PRE の適用によって実行効率の低減を防ぐ手法を提案した．本手法は，GVN と要求駆動型 PRE を組み合わせた手法を拡張し，副次的効果を反映しながらレジスタ圧力を低減する．今後の課題としては，実際に利用できるレジスタ数の考慮と，プロファイリングを利用し，各演算子の実際のコストを考慮し，冗長性除去の効果が高い式だけを除去する拡張が考えられる．

参考文献

[1] G. J. Chaitin. Register allocation & spilling via graph coloring. In *Proceedings of the 1982 SIGPLAN symposium on Compiler construction*, SIGPLAN '82, pp. 98–105, New York, NY, USA, 1982. ACM.

[2] J. Knoop, O. Ruthing, and B. Steffen. Lazy code motion. In *Proceedings of the ACM SIGPLAN 1992 conference on Programming language design and implementation*, PLDI '92, pp. 224–234, New York, NY, USA, 1992. ACM.

[3] 澄川靖信, 滝本宗宏. 効率的な要求駆動型部分冗長除去. 情報処理学会第 74 回全国大会講演論文集第 1 分冊, 第 74 巻, pp. 427–428. 情報処理学会, 2012.