2M-2

効率的な要求駆動型部分冗長除去

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** [1](Partial Redundancy Elimination，以降 PRE と呼ぶ) は，共通部分式除去とループ不変式移動を同時に行う強力な手法である．PRE は，データフロー方程式を用いてプログラム全体を網羅的に解析し，全ての式の冗長性を同時に除去する．この際，ある冗長な式を除去すると，その式に依存する後続の式の冗長性が新たに明らかになる**副次的効果**（Second Order Effects）が生じる可能性がある．副次的効果を反映するためには PRE を複数回適用する必要があるので，解析コストが高くなる．近年，一度の適用で副次的効果を反映させ，多くのプログラムで解析コストを抑える手法として，各式の出現ごとに冗長性を除去する**要求駆動型**（Demand Driven) の手法が提案されている [2]．要求駆動型の手法は，「式 e は冗長か」というクエリをプログラムの開始点に向かって伝播させることによって，字面が等しい式が存在するかどうかを調べ，式の出現ごとに冗長性を除去することができる．このような要求駆動型の手法は，除去された式に依存する式の冗長性を効果的に除去できるだけでなく，プログラムの解析範囲を限定できる．しかしながら，従来法では，クエリがプログラム全体を伝播しなければいけない場合が存在し，網羅型の手法よりも解析時間が長くなることがあった．

本研究では，式の値に基づいた**値番号**を生成し，各プログラム点に到達可能な値番号を記録しておくことで，クエリの伝播範囲を限定できる要求駆動型の手法を提案する．本手法では式 e の冗長性を除去するために，値番号 v を生成し，「値番号 v は冗長か」というクエリを伝播する．冗長であるかどうかは各プログラム点に記録されている表を参照することで直ちに調べられるので，不要な伝播を防ぐことができる．また，本手法は値番号を基にした手法であるので，字面は異なるが同じ値を計算する式の冗長性も除去することができる．

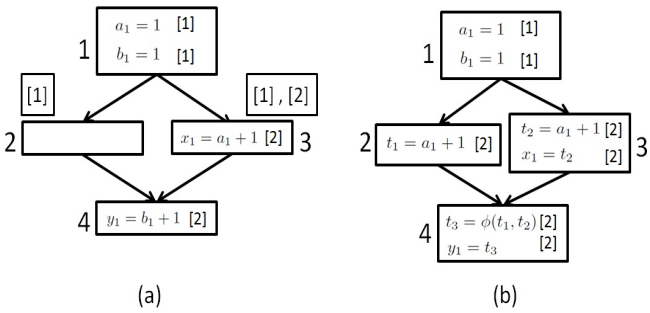

図 1　提案手法

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．本稿で使用する図では，値番号と定数を区別するために，値番号をカッコで囲むことにする．(a) の制御フローグラフ（Control Flow Graph，以降 CFG と呼ぶ）の節 4 の式 $b_1 + 1$ は，節 3 を含む実行経路上で，式 $a_1 + 1$ と同じ値を計算するので冗長である．一方，節 2 を含む実行経路上では冗長ではない．本手法では，$b_1 + 1$ に値番号 2 を生成し，節 2 と節 3 へクエリを伝播する．クエリが節 2 へ伝播したとき，節 2 の値番号表を調べると値番号 2 は存在しないので，節 2 へ到達する実行経路では冗長でないことがわかる．∎

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 節で従来の要求駆動型 PRE（以降 DDPRE と呼ぶ）を概説し，第 4 節で効率的な DDPRE を述べる．第 5 節で関連研究を述べ，第 6 節でまとめる．

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，節間の制御の様子を表す辺，特別な節である**開始節 s**，**終了節 e** からなる．

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

Effective Demand Driven Partial Redundancy Elimination
*Sumikawa Yasunobu, Tokyo University of Science
*Takimoto Munehiro, Tokyo University of Science

3　要求駆動型部分冗長除去

全ての経路で冗長な式は**全冗長**（Total Redundanct）と呼ばれ，以前の計算結果を再利用することによって除去できる．一部の経路で冗長な式は**部分冗長**（Partial Redundant）と呼ばれ，新たに式を挿入し，全冗長な式に変えることによって除去することができる．網羅型 PRE では，プログラム全体を解析し，式の挿入箇所を求め，すべての式の冗長性を除去する．

滝本らは質問伝播と呼ばれる手法を用いた DDPRE を提案した [2]．滝本らの手法では，CFG を**トポロジカルソート順序**で訪問し，式の出現ごとにクエリを後向きへ伝播する．クエリは字面が等しい式が存在するなら解 $true$ を返し，そうでなければ $false$ を返す．もし式 e がある節 n において，$true$ と $false$ の両方を得られたならば，n へ到達する実行経路上で e は部分冗長なので，全冗長とするために $false$ を得た節の下向き安全性を調べた後に新しい式を挿入する．

4　効率的な要求駆動型部分冗長除去

本手法は，式の出現ごとに値番号を求め，値番号が部分冗長である可能性があれば，クエリを伝播し，下向き安全性を検査をしない**投機的**な式の挿入を行う．クエリを伝播する式は次の 3 つある．

1. 既に生成された値番号を持つ式
2. ループ構造の内の式
3. ループの外にある ϕ 関数に依存する式

1 つ目は，他の式と同じ値番号が生成された式は冗長であるのでクエリを生成する．2 つ目は，ループ不変式をループの外へ移動するためにクエリを生成する．本手法ではループを識別するために ϕ 関数にも次の規則で値番号を生成する．

1. ϕ 関数の引数の変数がすべて同じ値番号を持つのであれば，ϕ 関数の値番号を引数のものと同じにする
2. 引数の値番号と順序が同じ ϕ 関数が存在するのであれば，その ϕ 関数と同じものを生成する
3. いずれの場合でもなければ新しい値番号を生成する

このとき，ループの入口にある ϕ 関数の引数には，未だ値番号を生成していない変数を持つ．3 つ目は，ループの外にある ϕ 関数に依存する式である．ϕ 関数に依存する式は，図 2 の式 a_3+1 のように，字面が等しい式が先行節に無いが，同じ値を計算する式が存在する場合がある．本手法では，クエリが ϕ 関数を含む節へ伝播したとき，ϕ 関数の値番号を，引数の値番号で書き換え，対応する先行節へクエリを伝播する．

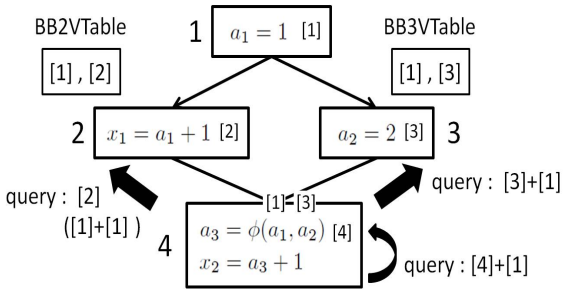

図 2　クエリ伝播

例：図 2 の節 4 の式 a_3+1 は，式の各項を値番号へ変換すると新しい値番号を生成するが，ϕ 関数に依存する式なのでクエリを伝播する．節 4 に a_3 を定義する ϕ 関数が存在するので，節 2 へ伝播するクエリは，値番号 4 を値番号 1 へ書き換える．同様に，節 3 へ伝播するときは値番号 3 で書きかえる．節 2 へ伝播するクエリは，値番号の変換の結果，値番号 2 となる．節 2 の値番号表を参照すると同じ値番号が存在するので解 $true$ を得る．一方，節 3 へ伝播するクエリは値番号を得られないので解 $false$ を得る．∎

5　関連研究

Knoop らは怠けたコード移動（Lazy Code Motion, 以降 LCM と呼ぶ）と呼ばれる字面が等価な式の冗長性を除去する網羅型手法を提案した [1]．LCM は最初に，冗長性を除去できる節の中で最も開始説に近い節を求める．次に，挿入する式の変数の生存範囲を短くするために，冗長除去可能な節のうち最も開始節から離れた節を求め，この節へ新しい式を挿入し，冗長除去を行う．

6　まとめ

本稿では，式の値を基に値番号を生成し，各プログラム点に到達可能な値番号を記録することによって，解析範囲を限定する手法を提案した．

参考文献

[1] Knoop, J., Ruthing, O. and Steffen, B.：Lazy Code Motion, *Proc. Programming Language Design and Implementation*, ACM, pp.224-234, 1992.

[2] 滝本宗宏：質問伝播に基づく投機的部分冗長除去，情報処理学会論文誌：プログラミング Vol.2, No.5, pp.15-27, 2009