実行時情報を利用した要求駆動型部分冗長除去

植村拓凪†　　澄川靖信‡

拓殖大学†

1 はじめに

繰り返し実行される冗長な命令を除去するコード最適化器に部分冗長除去（partial redundancy elimination，以降 PRE と呼ぶ）[1]がある．一般的な PRE はプログラム全体を解析するデータフロー解析を利用するが，プログラムの一部のみを解析することで解析時間を短縮する要求駆動型 PRE（demand-driven PRE，以降 DDPRE と呼ぶ）[2~4]が提案されている．DDPRE の先行手法は，すべてのプログラムの実行回数は等しいと仮定し，制御フローグラフ（control flow graph，以降 CFG と呼ぶ）をトポロジカルソート順序で訪問しながら，式が出現するたびに冗長性を解析するクエリを伝播する．このクエリは同じ式を見つけたら true，冗長ではないと判断したときは false を返戻する．ある CFG 節において，true と false の両方が得られたとき，クエリを生成した式がその節で部分冗長といえるので，全冗長となるよう，false を返戻した先行節に式を挿入する．この結果を利用するように，クエリを生成した式を置換して冗長性を除去する．

本研究では，命令ごとに実行される回数は実際には異なることに着目した，実行時情報を利用した DDPRE（profile-guided DDPRE，以降 PDPRE と呼ぶ）を提案する．この情報を利用することによって，PDPRE は実行回数が特に多いプログラムのみに DDPRE を適用できる．

2 提案手法

PDRRE は，実行回数が多い順に CFG を訪問する．この順序によって，上位 k%のものに限定して冗長性を除去できる．しかし，既存の DDPRE のように，字面の一致性を解析するクエリを使用すると，冗長性を除去した後にコピー伝播を適用して明らかになる冗長性（以降，副次的効果と呼ぶ）を反映できなくなる．そこで，PDPRE は前処理として，大域値番号付け（global value numbering，以降 GVN と呼ぶ）を適用する．

GVN を効率よく実行するために，PDPRE では，プログラムは静的単一代入形式に変換されているものと仮定する．以降では，GVN とクエリ伝播の詳細について述べる．なお，PDPRE はあらかじめ各 CFG 節の実行回数は取得済みであると仮定する．

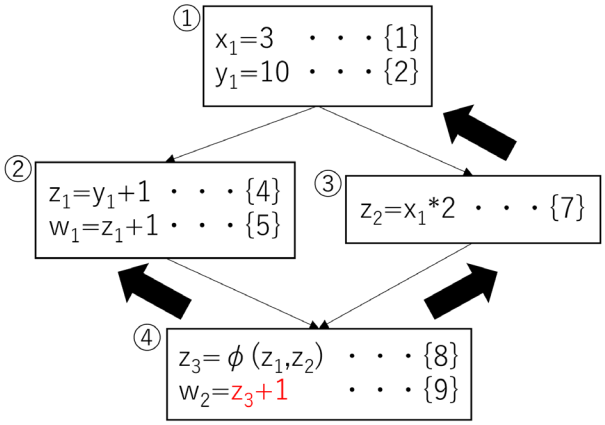

図1　値番号の付与例

2.1 大域値番号付け

GVN は，すべての式の値番号を表で管理する．値番号は，CFG をトポロジカルソート順で訪問し，すべての式に対して付与する．

図 1 に式とφ関数に対しての値番号を付与する例を示す．本稿では値番号を{}で括り，プログラム内の数字とは区別する．まず，図 1 の節①の文 x_1=3 の右辺に値番号を付与する．この例ではこれが初めて付与する値番号なので，3 をキー，値番号{1}を値として値番号表に格納する．次の文の右辺が 10 であり，初めて解析するため，新しい値番号として{2}を 10 に付与し，このエントリーを値番号表に格納する．四則演算の場合，まず，各項の値番号を求める．先ほどと同様に，未だ値番号表に格納していない，初めて解析するものであれば，その項に新しい値番号を付与して値番号表に格納する．もしこの項が値番号表に格納されているなら，対応する値番号を取得し，式の字面をその項と置き換える．この置換を終えた後の式が値番号表に格納されていれば，対応する値番号を取得する．さもなくば，新しい値番号を付与し，項を値番号に置換した状態の式と一緒に値番号表に格納する．例えば，図 1 では，式 y_1+1 に値番号を付けるとき，まずこの式を{2}+{3}に置換する．この式は初めて解

Profile-guided Demand-driven Partial Redundancy Elimination
†Takuna Uemura, Takushoku University
‡Yasunobu Sumikawa, Takushoku University

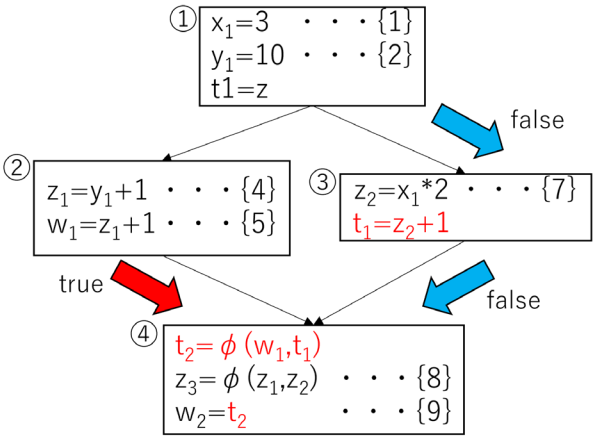

図2　実行例

表 1　各節の実行回数

CFG節	実行回数
①	50
②	30
③	20
④	50

析するので、新しい値番号として{4}を付与する．

ϕ 関数への値番号付けは，引数を使用する．図 1 の $\phi(z_1,z_2)$では左の引数の値番号は{4}であり，右の値番号は{7}なので，実行経路によって変数 z の値が変わる．このとき，新しい値番号をこの ϕ 関数に付与する．もしすべての引数の値番号が同じなら，その ϕ 関数にもその値番号を付与する．

2.2 CFG 節の訪問とクエリ伝播

PDRRE では実行回数が多い順に CFG 節を worklist に記録し，worklist から節を 1 つずつ取り出すことで各節を訪問する．訪問している節の中に式 e が出現するごとに，その式の値番号を取得し，「式 e の値番号 v は冗長であるか」を調べるクエリを逆向きに伝播する．以降では，図 1 に示す CFG の節④の式 z_3+1 を使用して PDPRE のクエリ伝播の様子を示す．また，図 2 にクエリ伝播と式変形の結果，表 1 にこの CFG の各節の実行回数をそれぞれ示す．

クエリを先行節に伝播するとき，訪問中の節にクエリの式で使用している変数を定義する ϕ 関数が存在するか調べる．もし存在するなら，伝播先の先行節に対応する変数を ϕ 関数の引数から取り出し，クエリの式変形と値番号の再取得を行う．例えば，節④の式 z_3+1 を{8}+{3}に変形した後，z_3 を定義している ϕ 関数 $\phi(z_1,z_2)$に注目し，先行節②にクエリを伝播するなら式の{8}を{4}，先行節③に伝播するなら{8}を{7}に置き換える．この後，クエリは新しい式の値番号に一致するものがあるかどうかを解析する．

クエリの解は，v と同じ値番号が出現したことを意味する true，伝播した経路上に v が出現しなかったことを意味する false，同じ値番号を解析するクエリが同じ節に 2 回訪問したことを意味する ⊤ のいずれか 1 つである．1 つの節で複数の解を得たとき，クエリ解を定義する半束上で交わり演算子によって最終的な解を求める．

1 つの節で true と false の両方が返戻されたとき，v はその節で部分冗長である．全冗長とするために，false を返戻した先行節に到達する経路上に新しく式を挿入する．PDPRE では，単に false を返戻した先行節を挿入節とするのではなく，クエリが false を返戻した節の中から実行回数が最小のものを挿入節とする．

3 まとめ

本稿では，実行時情報を利用する DDPRE である PDPRE を提案した．PDPRE は，CFG を実行回数が多い順に訪問し，各式の値番号が冗長かどうかを解析する．また，部分冗長な命令を全冗長に変形するための式の挿入点を，クエリが false を返戻した節の中から最も実行頻度が低い箇所とする．

今後の課題としては，PRE の枠組みでは除去できない冗長性を除去するために要求駆動型解析を利用する手法に対して，実行時情報を利用できるよう拡張することが考えられる．

参考文献

[1]: E. Morel and C. Renvoise, Global optimization by suppression of partial redundancies", Commun. ACM, 22, 2, p. 96 – 103 (1979).
[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 (PRO), 2, 5, pp. 15–27 (2009).
[3]: Yuya Yanase and Yasunobu Sumikawa, Lazy Demand-driven Partial Redundancy Elimination, Journal of Information Processing, Vol. 31, pp. 459–468, 2023.
[4]: Yasunobu Sumikawa and Munehiro Takimoto, Effective Demand-driven Partial Redundancy Elimination, IPSJ Trans. Programming, Vol. 6, No. 2, pp. 33–44, 2013.