2V-02

局所的集約　よる ログラム依存グラフの効率的な構築法

横森励士　　大畑文明　　井上克郎

大阪大学大学院基礎工学研究科

1　まえがき

バッグ　おけるフォ　ルト位置特定を効率良く行うための手法としてプログラムスライスが提案されている. プログラムスライスとは, プログラム内のある文のある変数 (**スライス基準**) に影響を与えうる全ての文の集合であり, プログラムスライスを利用することでプログラムの参照範囲を限定できる. 現在では, デバッグ支援だけでなくテスト, 保守, プログラム合成, プログラム理解などにも利用されている.

プログラムスライスの計算にはプログラム文間の依存関係を表すプログラム依存グラフ (PDG) が用いられる. しかし, プログラミング言語の高級化に伴い依存関係解析アルゴリズムは複雑化し, 大規模ソフトウェアに対する PDG 構築には多くの時間コスト, 空間コストがかかるようになった. そこで, スライス (PDG) の正確性と依存関係解析に要するコストとのトレ　ドオフを考慮する事が必要になってきた [1]. ここでいう正確性とは, スライスに含むべきものは含まれているという前提において, スライスに含まれるべきでない文をどれだけ排除できるかの程度である. 正確性の向上によりスライスサイズは減少し, 正確性の低下によりスライスサイズは増加する.

本研究では, 解析中, 依存関係が文 (節点) 単位で保持され続ける従来手法に対し, 手続き間解析の際に対象となる節点を一時的に集約することにより, 手続き間解析を効率よく行う手法 を提案する.

2　プログラムスライス

スライス計算の手順は [2] に示されている. ここでは, 手続き型言語におけるスライス計算の過程を簡単に述べる.

Phase 1: 定義, 参照変数 抽出

プログラムの各文で定義, 参照される変数を抽出する.

Phase 2: 依存関係解析

プログラム文間のデ　タ依存関係, 制御依存関係を計算する. デ　タ依存関係とは, ある変数を定義した文とその変数を参照している文の間に存在する関係である. 制御依存関係とは, 分岐命令とその分岐文内に直接含まれている文や, ル　プ命令とそのル　プ文内に直接含まれている文の間に存在する関係である.

Phase 3: 手続き依存グラフ構築

依存関係解析結果を元に手続きごとに手続き依存グラフを構築する. 手続き依存グラフとは, 手続き内に存在する文を節点, 文間のデ　タ依存関係, 制御依存関係を辺で表現した有向グラフである.

Phase 4: 手続き間の依存関係解析

構築された手続き依存グラフを元に, 手続きの呼び出し文の実行前と実行後の間に存在する変数の依存関係を解析しプログラム依存グラフを完成させる. プログラム中の全ての手続き依存グラフに手続き間の依存情報を組み合わせたものを**プログラム依存グラフ (Program Dependence Graph, 略して PDG)** と呼ぶ.

Phase 5: スライス 抽出

スライスを抽出する. スライス基準に対応する PDG 節点から逆向きに制御依存辺およびデ　タ依存辺を経て推移的に到達可能な節点集合に対応する文がスライスとなる.

3　提案

3.1　従来手法の問題点

最近のプログラムの大規模化に伴い, スライスの計算に大量の時間・空間コストがかかるようになった. また手続き型言語においては手続きの呼び出しごとに手続き内部の再解析を行う必要があるため, 通常 Phase 4: 手続き間の依存関係解析で特に手間がかかる. 例えば, 約 28000 行の C プログラムの PDG 構築に約 2 時間を要し, その 9 割以上が手続き間の依存関係解析に費やされたという報告がされている [2].

この問題の解決法として従来は依存関係解析アルゴリズムを計算量の少ないものに変更することでコストを削減し, 正確性とのトレードオフを行う方法が考えられていた. しかし, 異なる依存関係解析アルゴリズムにより構築された PDG の相互利用は難しく, 実現できても PDG 構造が複雑になる事が多い.

Efficient Construction Method of Program Dependence Graph Using Node Integration
Reishi Yokomori, Fumiaki Ohata and Katsuro Inoue
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka, 560-8531 JAPAN.

3.2 提案する手法

今回我々は**節点集約**を用いたコスト削減手法を提案する. 節点集約とは, プログラム文と PDG 節点が 1 対 1 で対応していたものを多対 1 にするもので, 複数の文を 1 つの節点で表現する. この節点集約を依存関係解析前に行い, 各手続き内の PDG 節点数を減らしてから解析する. 解析後に集約した節点を分解し, PDG に置き換える.

この方法では, 依存関係解析時に各手続き内の PDG 節点数を減らすことができ, また手続き間の依存関係解析時にたどるべき節点数を減らすことができるため, 計算対象の削減による計算量 (時間コスト) 削減が可能となる. また節点集約を依存関係解析前に行うことで, 既存の依存関係解析アルゴリズムと独立させることができるため, 複数のアルゴリズムとの併用が容易になる.

4 計算の手順

今回提案する方法によるスライス計算の過程は次のようになる.

Phase 1: 定義, 参照変数 抽出
Phase 1.5: 節点集約
Phase 2: 依存関係解析
Phase 3: 手続き依存グラフ構築
Phase 4: 手続き間の依存関係解析
Phase 4.5a: 節点分解
Phase 4.5b: 集約節点内部の依存関係解析
Phase 5: スライス 抽出

4.1 節点集約

今回提案する方法では, 次の方針でプログラムの先頭から集約していく. ただし複数の条件を満たす場合は上の方針を優先する.

連続した文 S_1,S_2 において

- S_1 または S_2 が手続き呼び出し文である場合
  (1) S_1,S_2 は集約しない.
- S_1 または S_2 が分岐文や繰り返し文の場合
  (1) 分岐文や繰り返し文のブロック内の文の集約を行う.
  (2) ブロック内が一つの節点に集約できる場合はその分岐文や繰り返し文全体を一つの節点に集約する.
  (3) S_1,S_2 がどちらも一つの集約節点になるならば, S_1,S_2 を集約する. そうでなければ集約しない.
- S_1,S_2 のどちらも分岐文や繰り返し文または手続き呼び出し文でない場合
  (1) S_1,S_2 を集約する.

4.2 節点分解

手続き間の依存関係解析後に節点を分解し, 集約されていた文の依存関係解析を行なうことで集約節点を PDG の一部分に置き換える. しかし, 分解の際にプログラム文と PDG 節点が 1 対 1 対応になるまで分解してしまうと, 必要なメモリの量などの空間コストの削減は期待できない. そこで [3] で提案されている「依存関係の局所性」を用いた手法に基づいて分解を行ない, 一部の PDG 節点を複数の文に対応させる. その後, 集約されていた文の依存関係解析を行ない, 集約節点を PDG の一部分に置き換える. これにより得られるスライスの正確性の低下を抑えつつ, 分解時の節点数の増加を少なくすることができ, スライス抽出時の PDG 節点数を減らすことができるため, 空間コストの削減が期待できる.

4.3 依存関係の局所性

[3] で提案されている「依存関係の局所性」では, 連続している文において, それぞれの文が依存している変数の集合がある程度一致している場合は, それらの文を一つの節点に集約して依存関係の解析やスライスの抽出を行っても, 得られるスライスの正確性があまり低下しないことがしめされている. 例えば [3] での実験によると, スライスサイズの増加が 3 %以内であった一方で PDG の節点数が 20〜40 %, 解析時間は 20〜40 %削減できたという結果がしめされている.

5 まとめ

本研究では手続き間解析時の一時的な節点集約による, 手続き間解析を効率よく行う手法を提案した.

今後の課題としては,

- 本手法の実装
- 大規模プログラムに対する評価

を考えている.

参考文献

[1] Atkison, D. C. and Griswold, W. G. “The Design of Whole-Program Analysis Tools”. In *Proceedings of the 18th International Conference on Software Engineering*, Berlin, Germany, pages 16–27, 1996.

[2] The Wisconsin Program-Slicing Tool 1.0, Reference Manual. Computer Sciences Department, University of Wisconsin-Madison, 1997.

[3] 大畑, 西松, 井上 “依存関係の局所性を利用したプログラム依存グラフの効率的な構築法”, 情報処理学会研究報告, Vol.99, No.28, pages 17-24, 1999.