変数依存グラフを用いたプログラムスライス計算アルゴリズム*

6F－9

太田　剛，渡辺　尚，水野忠則†
静岡大学　工学部‡

1　はじめに

プログラムスライス [1] は，逐次型プログラムのデバッグ支援を目的として提案されたが，近年，テスト，保守，理解等，広い範囲へ応用できる技術であることが認識されてきた [2]．スライスを求めるアルゴリズムとしては，Weiser[1], Leung[5], Ottenstein[4], Bergeretti[3], 直井 [6] 等が知られている．

本稿では，構造化プログラムを対象とした新たなスライス計算アルゴリズムを述べる．また，その計算量について考察し，既存アルゴリズムと比較検討する．

2　プログラムスライス

プログラムスライシングとは，プログラム内のある文の実行に影響を与える全ての文を抽出する技術であり，抽出された文の集合をスライスと呼ぶ [2]．スライスには，テキストを静的に解析して得られる静的スライスと，プログラムの実行履歴を解析して得られる動的スライスとがある．本稿では，静的スライスを対象とする．

静的スライスは，スライス基準 $C(i,V)$ が与えられたとき，文の間に定義される次の関係を辿って得られる，ある文の実行に影響を与える可能性を持つ全ての文の集合である．ただし i は文の番号，V は変数の集合を表す．

- データ依存関係 — 文 S_1 で値を定義された変数 v が，値を再定義されることなく，v の値を参照している別の文 S_2 に到達する場合．
- 制御依存関係 — 文 S_1 が分岐文あるいはループ文であり，その本体内に文 S_2 が存在する場合．

静的スライスの例を図 1に挙げる．なお，行番号に * を付した文が，スライス基準 $C(10,\{\mathtt{sum}\})$ に関して求めた静的スライスである．10 行めとのデータ依存関係によって 1, 5 行めが，また，5 行めの文がループ文の本体にあることから，制御依存関係によって 4 行めが得られる．さらに，4 行めで参照されている変数 rain とのデータ依存関係から 3, 8 行めが得られる．

3　変数依存グラフ

変数依存グラフとは，3 種類の辺を持つ有向グラフであり，次の性質を満たす．

- 各頂点は，プログラム中の各変数に対応する．ただし，どの変数とも対応しない頂点 T を持つ．
- データ依存辺は，プログラム中の各代入文に現れる両辺の変数の依存関係を示す．この辺は，文番号をラベルとして持つ．なお，定数を代入する文においては，T と依存関係を持つと定義する．
- ループ依存辺は，ループ文の条件節に現れる変数と，その本体内で代入される変数との依存関係を示す．この辺は，ループ文の開始文番号をラベルとして持ち，終了文番号を付加情報として持つ．
- 分岐依存辺は，分岐文の条件節に現れる変数と，その本体内で代入される変数との依存関係を示す．この辺は，分岐文の文番号をラベルとして持つ．

```
 1 *  sum = 0;
 2    rainy = 0;
 3 *  read(rain);
 4 *  while (rain != 99) {
 5 *    sum = sum + rain;
 6      if (rain != 0)
 7        rainy = rainy + 1;
 8 *    read(rain);
 9    }
10 *  write('sum = ', sum);
11    write('rainy = ', rainy);
```

図 1: スライス基準 $C(10,\{\mathtt{sum}\})$ による静的スライス (* を付した文)

図 1のプログラムに対応する変数依存グラフを図 2に示す．

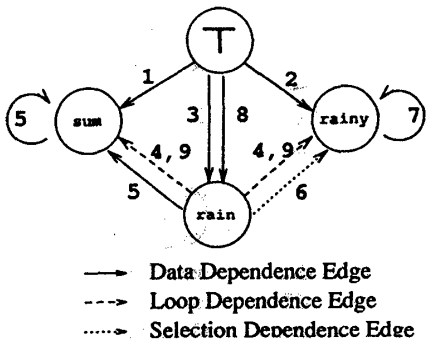

図 2: 図 1のプログラムの変数依存グラフ

4　スライス計算アルゴリズム

スライス基準 $C(i,V)$ に関するプログラムスライスを，変数依存グラフを用いて計算するアルゴリズムを示す．

$C(i,V)$ に関するスライス計算アルゴリズム

1. V の各要素 v_k について，ステップ 2. を行なう．
2. v_k に相当する頂点の入力辺のうち，i 以下のラベルが付いており，かつ，マークされていない各辺 e_l について，e_l にマークを付け，次の A,B,C のいずれかを行なう．もしそのような入力辺が存在しなければ 3. へ．

---

*A Program Slicing Algorithm using Variable Dependence Graph

†Tsuyoshi Ohta, Takashi Watanabe and Tadanori Mizuno
‡Shizuoka University

表 1: スライス計算アルゴリズムの比較 (プログラムサイズを n とする)

アルゴリズム	中間表現	テキストから中間表現		中間表現からスライス	
		最悪の場合	実質	最悪の場合	実質
Weiser	フローグラフ	$O(n)$	$O(n)$	$O(n^3 \log n)$	$O(n^2 \log n)$
Ottenstein	プログラム依存グラフ	$O(n^3)$	$O(n^2)$	$O(n^3)$	$O(n^2)$
Bergeretti	μ 関係行列	$O(n^4)$	$O(n^4)$	$O(n)$	$O(n)$
本稿	変数依存グラフ	$O(n^3)$	$O(n^2)$	$O(n^3)$	$O(n^2)$

A. e_l がデータ依存辺である場合．この辺を逆向きに辿って得られた頂点に相当する変数を v'，この辺のラベルを i' としたとき，スライス基準 $C(i', \{v'\})$ として，本アルゴリズムを再帰的に適用する．

B. e_l が分岐依存辺である場合．この辺を逆向きに辿って得られた頂点に相当する変数を v'，この辺のラベルを i' としたとき，スライス基準 $C(i', \{v'\})$ として，本アルゴリズムを再帰的に適用する．

C. e_l がループ依存辺である場合．この辺を逆向きに辿って得られた頂点に相当する変数を v'，この辺の付加情報を a としたとき，スライス基準 $C(a, \{v'\})$ として，本アルゴリズムを再帰的に適用する．

3. マークされている辺のラベルに相当する文を集める．この和集合が求めるスライスである．

Weiser のアルゴリズムにおける $S^i_C \cup B^i_C$ の計算 (これが静的スライスとなる) は，本アルゴリズムにおいては，ループ依存辺または分岐依存辺をちょうど i 個と任意個数のデータ依存辺を辿ってラベルを得ることに相当する．

5 計算量

アルゴリズムの計算量に関して，プログラムからスライス計算のための内部構造を作成するまで，および，それからスライスを計算するまでの 2 段階に分けて議論する．なお，n をプログラムのサイズとする．

プログラムテキストから変数依存グラフを構成するための計算量は，頂点および辺の数に比例する．頂点の数は，変数の数 $|V|+1$ である．また，辺の数は，代入文の数 $|A|$，制御文の数 $|C|$，制御文本体内の代入文の数 $|B|$ とすると，最悪の場合データ依存辺が $|A| \times |V|$，分岐依存辺とループ依存辺の和が $|C| \times |B| \times |V|$ となる．したがって，変数依存グラフを構成するためには $|V|+1+|A||V|+|C||B||V|$ に比例した時間を要することになる．最悪の場合 $|V| = |A| = |C| = |B| = n$ であるから，これは $O(n^3)$ の計算量となる．ただし，実質上，制御文の条件節に現れる変数の数は，ある小さい定数 k 以下とみなせるので，$O(n^2)$ の計算量であると見て良い．

変数依存グラフからスライスを計算するための計算量は，グラフの探索問題における計算量と同じものとなり，$O(|V| + 2 \cdot |C||B||V|) = O(n^3)$ となるが，ここでも前述の議論と同様 $O(n^2)$ であると見て良い．

6 議論

表 1に，Weiser(Leung の訂正版), Ottenstein[1], Bergeretti のアルゴリズムとの比較を示す [1, 5, 4, 3].

[1]文献 [4] には，スライス計算時間は線形だとあるが，本稿のアルゴリズムと同様にグラフの探索問題に帰着してあるので，これは誤りであろう．

Weiser のアルゴリズムは，テキストから中間表現への計算量と比較して，中間表現からスライスへの計算量が格段に多く，プログラム修正を伴うデバッグ作業に向いているとは言えない．Bergeretti のアルゴリズムでは，一旦 μ 関係行列を計算してしまえばスライスの計算にかかる時間は短いため，プログラム変更の必要がない保守時のプログラム理解や影響調査に向いていると言える．これらと比較して，Ottenstein や本稿のアルゴリズムは，プログラム修正を伴うデバッグ作業に比較的向いていると言える．ただし，プログラム依存グラフはスライス計算だけを目的として提案された表現法ではないため，本稿の変数依存グラフと比較して，格段に頂点数，辺数共に多く，それだけ計算量は多くなる．

7 おわりに

本稿では，変数依存グラフを用いたスライス計算アルゴリズムを述べ，その計算量について他のアルゴリズムと比較検討した．今後は，これらのアルゴリズムを実装し，実測実験による比較をする予定である．また，デバッグを支援するために，プログラム変更に柔軟に対応できる，インクリメンタルなスライス計算アルゴリズムについて考察する予定である．

謝辞 本研究の一部は，住友財団の基礎科学研究助成を受けて行なわれた．

参考文献

[1] M.Weiser: "Program Slicing," *IEEE Trans. Softw. Eng.*, Vol.SE–10, No.4, pp.352–357 (1984).

[2] 下村: "Program Slicing 技術とテスト，デバッグ，保守への応用," 情報処理, Vol.33, No.9, pp.1078–1086 (1992).

[3] J.-F.Bergeretti, B.A.Carré: "Information-Flow and Data-Flow Analysis of while-Programs," *ACM Trans. Program. Lang. & Syst.*, Vol.7, No.1, pp.37–61 (1985).

[4] K.J.Ottenstein, L.M.Ottenstein: "The Program Dependence Graph in a Software Development Environment," *ACM SIGPLAN Notice*, Vol.19, No.5, pp.177–184 (1984).

[5] H.K.N.Leung, H.K.Reghbati: "Comments on Program Slicing," *IEEE Trans. Softw. Eng.*, Vol.SE–13, No.12, pp.1370–1371 (1987).

[6] 直井, 高橋: "経路依存フローグラフを用いたプログラム・スライシング," 情処研報, Vol.93, No.84, 93–SE–94, pp.41–48 (1993).