5H-03

プログラム理解のためのスライシング基準の特定と評価

増原　孝昭†　　芳賀　博英†
同志社大学大学院　理工学研究科†

1. 概要

ソフトウェア開発では，ソフトウェアの保守・拡張に多く時間が費やされている．中でもソースコードを読み理解する作業は多くの時間を要する[1]．そのため，ソースコードを読み理解する作業を支援し，その時間を削減する手法が求められている．

プログラム理解を効率的に行うための技術として，プログラムスライシングがある[2]．プログラムスライシングは，スライシング基準と呼ばれるプログラム内の特定のステートメントの変数を基準として，その変数に影響を与えるすべてのステートメントをスライスとして抽出する．これは，開発者が注目しているソースコードの断片の内容を理解するにあたって，変数の値がどのように計算されているかを調べるために用いられる[3]．プログラムスライシングによって，開発者はプログラム理解のために読むソースコードの量は少なくなり，理解にかかる時間を減らすことができる．特に文献[4]ではスライスによりプログラム理解にかかる時間が減少した例が紹介されている．また，文献[5]ではプログラムの特定の機能の実装方法とそのソースコード中の位置を開発者が理解することを目的としたソースコードの探索方法を考案している．しかし，スライスを抽出するために着目すべき変数は開発者が選択する必要があり，複雑なプログラムではどの変数に着目すべきかわかりにくい．

本報告では，プログラムスライシングより抽出されるスライスをもとに，どの変数から着目すべきか各変数を定量的に表現することで特定する．解析対象のプログラムのすべてのスライスについて，効率よく理解ができる順にスライスを開発者に提示できるように着目すべき変数の優先順位を求める．基本的なアルゴリズムのプログラムを対象に本手法を適用したところ，着目すべき変数を手作業で指定する場合と同じ変数を特定できる場合があることを確認した．

2. プログラムスライシング

プログラムスライシング（以下，スライシング）とは，プログラム内の，ある特定のステートメントの変数（スライシング基準）に影響を与えるステートメントの集合を抽出する技術である．このステートメントの集合をプログラムスライス（以下，スライス）と呼ぶ．スライシングの手順を以下に示す．

1. データ依存グラフを作成する．データ依存グラフとは，データ依存関係を表すグラフを指す．ある変数を定義するステートメントと，その変数を参照するステートメントとの関係をデータ依存関係と呼ぶ．
2. 制御依存グラフを作成する．制御依存グラフとは制御依存関係を表すグラフを指す．分岐があるステートメントと分岐によって実行するかどうかが決まるステートメントとの関係を制御依存関係と呼ぶ．
3. プログラム依存グラフを作成する．データ依存グラフと制御依存グラフを統合すると，図1の右のプログラム依存グラフができる．
4. スライシング基準より，プログラム依存グラフのエッジを逆向きに辿り， 到達可能なノードの集合を求める．そのノードの集合に対応したステートメントがスライスとなる．スライシング基準はスライスを求めるための基点となるステートメントと変数の組を指す．たとえば，図1の右のプログラム依存グラフにおいて，スライシング基準を〈7, d〉とした場合のスライスは{1, 3, 4, 5, 6, 7}となる．

```
   void func(int a, int b){
1      int v1 = a;
2      int v2 = b;
3      int c;
4      int d;
5      scanf("%d", &c);
6      if( c == 0 )
7          d = v1;
       else
8          d = v2 + 1;
9      printf("%d¥n", d);
   }
```

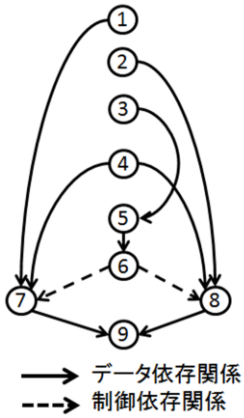

図1 スライシングの例

3. スライシング基準の順位付け

プログラムスライシングでは，スライスを抽出するために着目すべき変数は，開発者が選択する必要がある．しかし，複雑なプログラムではどの変数に着目すべきかわかりにくい．そこで，どの変数から着目すべきか優先順位を決定するために，以下の手順で，解析対象のプログラムの各変数のスライシング基準に選ぶべき優先度を定量的に表現する．

1. 解析対象のプログラムのすべての変数についてスライスを求める．このスライスの集合を$S = \{s_0, s_1, \cdots, s_n\}$とする．
2. x番目のスライス$s_x \in S$について，$s_y \subseteq s_x$を満たすスライスの集合$SL_x = \{s_y | s_y \subseteq s_x\}$を求める．

Identifying and evaluation slicing criteria for program comprehension
Masuhara Takaaki†, Haga Hirohide†
†Graduate School of Science and Engineering, Doshisha Universitv

3. x番目のスライスを得るためのスライシング基準を選択する優先度p_xを以下の式(1)より求める．

$$p_x = \frac{1}{|SL_x|}\sum_{s_y \in SL_x} |s_y| \quad (1)$$

4. すべてのスライスについて，p_xの値が小さいものから順にスライスを開発者に提示する．

4. 評価実験

3 節で述べた手法について，開発者がスライシング基準を選択する場合と同様の順にスライスを抽出することが可能か実験を行った．

4.1 手順

1. 基本的なアルゴリズムのプログラムを用意する．ここでは，(A)Gauss-Legendre の二次収束公式により円周率を求めるプログラム，(B)配列の和，積，平均値を求めるプログラム，(C)シンプソンの公式で積分値を求めるプログラムを解析対象とした．各プログラムについてどの変数をスライシング基準に設定すればプログラムの理解の時間を短縮可能なスライスが抽出できるのか，その理想的な順序を各変数について手動で設定する．ただし，理想的な順序とはサイズが小さいスライスから特定の変数の計算を順方向に追って理解する方法[5]によるものとする．また，宣言と初期化のみが行われているステートメントの変数は順位付けを除外し，順位付けの判断が難しい複数の変数についてはそれらを同順位とする．
2. 3 節に示した手法で各プログラムのスライシング基準として選択する優先度を解析対象のプログラムそれぞれについて求める．
3. 2.で求めた各変数の順位が，1.で求めた順位とどれだけ一致しているかを，解析対象のプログラムごとに比較する．

4.2 結果

4.1 節に示したプログラム(A)(B)(C)に対して 3 節の手法による優先順位の算出結果を以下の表 1，表 2，表 3 に示す．

表 1 プログラム(A)のスライシング基準の順位付け

スライシング基準	順位(手動)	順位(本手法)	優先度
〈18, res〉	5	5	4.23
〈17, y〉	4	3	3.29
〈15, x〉	1	1	2.00
〈14, t〉	3	4	3.50
〈11, aryB〉	2	2	3.00
〈10, aryA〉	2	2	3.00

表 2 プログラム(B)のスライシング基準の順位付け

スライシング基準	順位(手動)	順位(本手法)	優先度
〈13, ave〉	6	6	3.23
〈12, mul〉	2	4	2.75
〈11, sum〉	4	4	2.75
〈10, ave〉	5	3	2.60
〈8, mul〉	1	1	2.00
〈7, sum〉	3	1	2.00

表 3 プログラム(c)のスライシング基準の順位付け

スライシング基準	順位(手動)	順位(本手法)	優先度
〈15, res〉	6	6	5.08
〈14, res〉	5	5	4.36
〈13, A〉	4	4	3.88
〈11, x〉	1	1	2.00
〈10, A〉	3	3	3.13
〈9, dA〉	2	2	2.50

4.3 考察

(A)の結果に関して，〈17, y〉と〈14, t〉については本手法と手動による順位付けでは逆の順位となった．これは，ソースコード中の変数の位置について〈11, aryB〉は〈17, y〉より〈14, t〉の方が距離が短いためであると考えられる．したがって，スライスのサイズだけではなく，各変数のソースコード中の位置を考慮してスライスを提示する必要がある．また，(C)の結果では順位が手動の場合とすべて一致しているが，(B)については半数が一致という結果になった．これは，積を求める処理と平均を求める処理が別の機能として混在しているためであり，機能ごとに分類してからスライシング基準の順位付けを行うと手動の場合と順位が一致するようになる．このため，機能ごとに分類してから順位付けを行う必要があると考えられる．

5. まとめ

本報告では，プログラムスライシングにより複雑なプログラムを効率よく理解するために，スライシング基準としてどの変数を選ぶべきか優先度を定量的に表現することで特定した．基本的なアルゴリズムのプログラムについて本手法を適用したところ，着目すべき変数を手作業で指定する場合と同じ変数を特定できる場合があることを確認した．今後の課題として，より巨大なプログラムに対して本手法を適用した場合に着目すべき変数の優先順位が手動の場合と一致するのか実験を行う必要がある．また，一致する精度を高めるためには複数の機能が混在したプログラムにおいてどのように優先度を設定すべきか考慮する必要がある．

参考文献

[1] T.A. Corbi, “Program Understanding,” Challenge for the 1990's,*IBM Systems Journal*, Vol. 28, No.2, 294-306 (1989).
[2] M. Weiser, “Program Slicing,” In *Proc. of ICSE*, 439-449 (1981).
[3] T.D. LaToza and B.A. Myers, “Developers Ask Reachability Questions,” *Proc. 32nd IEEE/ACM International Conference on Software Engineering*, Vol. 1, 185-194 (2010).
[4] M. Sridharan, S. J. Fink and R. Bodik, “Thin Slicing,” *Proc. 2007 ACM SIGPLAN Conference on Programming Langurage Design and Implementation*, 112-122 (2007).
[5] K. Chen and V. Rajlich, “Case Study of Feature Location Using Dependence Graph,” *Proc. 8th IEEE International Workshop on Program Comprehension,* 241-237 (2000).