B-003

COINS を用いた静的プログラムスライシングツールの実装
Implementation of Static Slicing Tool Using COINS Compiler Infrastructure

上野 智弘† 芳賀 博英†
Tomohiro Ueno Hirohide Haga

1. はじめに

テストセットの品質を評価する手法の 1 つに，ミューテーション解析がある．この手法では，プログラムに故意にバグを埋め込んだミュータントを生成し，そのミュータントをテストセットが検出できるかによってテストセットの品質を評価する．ミュータントは，ミュータントを生成する規則であるミューテーションオペレータを，ステートメントに適用することで生成できるが，すべてのステートメントに対してすべてのミューテーションオペレータを適用すると，膨大な数のミュータントが生成され，その結果ミュータントの生成・コンパイル・実行時間が膨大になるので，品質評価に影響がない範囲で，生成されるミュータントの数を減らす必要がある．

これは，あるステートメントが他のステートメントにどのくらいの影響度を持っているかを測り，その影響度を基に，ミューテーションオペレータを適用するステートメントを減らすことで解決できる可能性があり，その影響度を測ることができると考えられる手法の 1 つが，静的プログラムスライシングである．

本報告では，ミューテーション解析の問題解決に使用できる静的プログラムスライシングツールの実装が目的であり，コンパイラ・インフラストラクチャである COINS を用いて静的プログラムスライシングツールを実装した既存研究を参考にし，発展させた[1][2].

2. 静的プログラムスライシング

2.1 概要

あるステートメントの実行が影響を与える可能性があるすべてステートメントの集合，あるいは，あるステートメントの実行に影響を与える可能性があるすべてステートメントの集合を静的スライスといい，その静的スライスを求めることを，静的プログラムスライシングという[3]．静的プログラムスライシングは，スライシング基準という基準を基に，プログラム依存グラフという有向グラフを辿ることで，静的スライスを求めることである．あるステートメントの実行が影響を与える可能性があるすべてステートメントの集合である静的スライスを求めることをフォーワードスライシング，あるステートメントの実行に影響を与える可能性があるすべてステートメントの集合である静的スライスを求めることをバックワードスライシングという．図 1 は，スライシング基準を 6 行目でバックワードスライシングした例で，左側が静的スライス，右側がプログラム依存グラフである．図 1 左の 2 行目は，抽出されないステートメント，6 行目がスライシング基準のステートメントである．本報告でのバックワードスライシングは，プログラム依存グラフを逆方向に辿ることで求められるので，図 1 右のプログラム依存グラフを 6 行目のノードから逆方向へ辿ると，図 1 の静的スライスが抽出される．

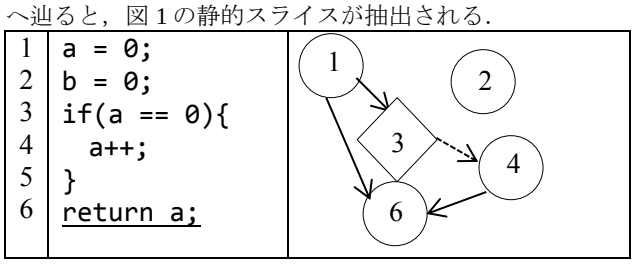

図 1 バックワードスライシングの例

静的プログラムスライシングの処理手順は，以下の通りである．

1. 制御依存グラフを作成する．
2. データ依存グラフを作成する．
3. プログラム依存グラフを作成する．
4. スライシング基準を基に，静的プログラムスライシングを行う[1].

2.2 静的スライス

スライシング基準は，どのステートメントに関して，あるいは，どのステートメントのどの変数に関して静的スライスを求めるかを決めるための基準であり，

「スライシング基準 C =（u，V）
（1）u はプログラム内のステートメント
（2）V はプログラム内の変数の部分集合」

で定義される[3].

静的スライスは，あるステートメントの実行が影響を与える可能性があるすべてステートメントの集合（フォーワードスライス），あるいは，あるステートメントの実行に影響を与える可能性があるすべてステートメントの集合（バックワードスライス）である．

2.3 プログラム依存グラフ(PDG)

プログラム依存グラフは，

「ソースプログラムの要素間の依存関係を有向グラフ化したものである．」[1]

で定義され，実装したツールでのプログラム依存グラフは，有向グラフのノードがステートメント，エッジが制御依存とデータ依存に相当し，制御依存グラフとデータ依存グラフの 2 つを統合して，作成できるものである．

制御依存グラフは，制御依存を示したグラフであり，制御依存は，

「X と Y ノードは制御フローグラフ上にあり， X からある辺を辿ると必ず Y を通るが，別の辺を辿ると Y を通らないとき，制御依存が X から Y にあるという．」[4]

で定義される．

データ依存グラフは，データ依存を示したグラフであり，データ依存は，

†同志社大学大学院 Doshisha University

「ステートメント X である変数を定義し，それが，その変数を参照するステートメント Y に到達する可能性があるとき，データ依存が X から Y にあるという．」[3]
で定義される．

3. COINS コンパイラ・インフラストラクチャ

3.1 概要

COINS コンパイラ・インフラストラクチャは，コンパイラを構成する基本機能のモジュールをすべて備え，それらの組み合わせを変えたり，一部のモジュールを新たに開発したりするだけで，新しいコンパイラを実現することができるコンパイラ基盤である[2].

実装したツールでは，HIR，制御フロー情報，データフロー情報を COINS から取得し，使用した．HIR は，高水準中間表現，制御フロー情報は，基本ブロックの親や子などの情報，データフロー情報は，変数の定義の到達や参照などの情報である．

3.2 実装

実装したツールの構成は，図 2 の通りである．

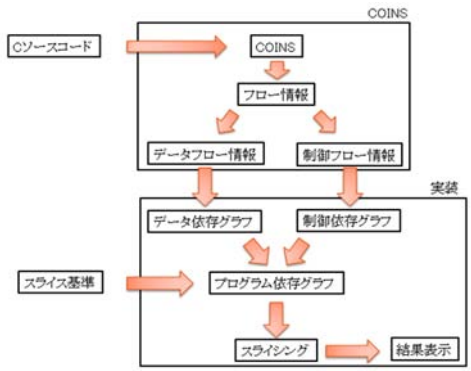

図 2　実装したツールの構成[1]

プログラム依存グラフは，2．3 節で説明した定義に基づいて作成する．　プログラム依存グラフの例を図 3 に示す．ただし，実線が制御依存，点線がデータ依存を表す．

```
1  int main(){
2  int a;
3  a = 0;
4  if(a == 0){
5    a++;
6  }
7  return a;
8  }
```

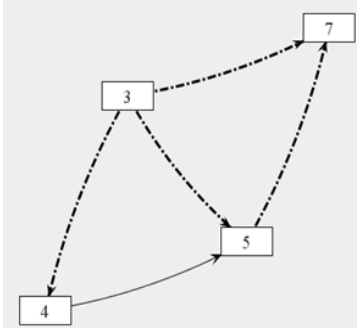

図 3　プログラム依存グラフの例

4. 評価実験

C 言語の基本的な機能である条件文，ループ文，配列，関数呼び出しを用いたプログラムに実装した静的プログラムスライシングツールを適用して，正しい結果を出力するかどうかを確かめる実験を行った．

5. 実験結果の検討

実験結果と手作業で定義に沿って静的スライシングを行った結果が一致したので，配列，if 文，for 文，while 文，switch 文，関数呼び出しを含むプログラムで，実装した静的プログラムスライシングツールは正しく動作することが確認できた．これは，COINS の生成する情報を適切に利用できたことを示している．

課題として，do-while 文，配列使用の一部，ポインタと大域変数，関数呼び出しを同じ行に 2 つ以上記述できない，COINS がソースプログラムを正しく解析できない，もしくは，解析しないことを含む場合に，実装したツールが正しく動作しないことがあげられる．do-while 文は，COINS が do-while 文の入口である do 部分の行番号を HIR 上に表示するが，条件判定の while 部分の行番号を HIR 上に表示しない．従って，制御依存の辺をソースプログラムのどの行から引けばいいのかわからないため，対応していない．配列使用の一部というのは，例えば，int a[2]で宣言した 1 次元配列を，F(a)のように，ただの変数として記述することである．これは通常の配列使用の HIR 上での表現と異なるため，対応していない．HIR 上での表現の変化に対応できれば，解決できると考えられるので，今後の課題である．ポインタと大域変数は，どの関数でも共通に定義・参照されるため，データ依存が複雑になる．従って，対応していない．関数呼び出しを同じ行に 2 つ以上記述できない理由は，スライシング部分で，どの関数の何番目の引数の変数からスライシングを行うかを判別できていないからである．これは，スライシング部分を改良すれば対応できると考えられるので，今後の課題である．COINS がソースプログラムを正しく解析できない，もしくは，解析しないことというのは，関数呼び出しの引数に使用した変数にその関数呼び出しの返り値を代入する，変数の宣言と同時に定義できない，同じ行に複数のステートメントを記述できないである．これらは，COINS がソースプログラムを正しく解析できない，もしくは，解析しないので対応していない．

6. おわりに

本報告では，ミューテーション解析の問題解決に使用できる静的プログラムスライシングツールの実装が目的であり，COINS を用いて静的プログラムスライシングツールを実装した既存研究を参考にし，発展させた[1]．具体的には，参考にした既存研究では対応していなかった配列と関数呼び出しに対応した．課題はいくつか存在するが，実装したツールは，C 言語の基本的な機能に対応できている．従って，本報告の目的である，ミューテーション解析の問題解決に使用できる静的プログラムスライシングツールとして，一定の評価は得られると考えられる．

参考文献

[1] 溝渕 裕司，中谷 俊晴，佐々 政孝，“コンパイラ・インフラストラクチャを用いた静的プログラムスライシングツール，”日本ソフトウェア科学会第 20 回大会（2003 年度）論文集，(2003).
[2] COINS コンパイラ・インフラストラクチャ，http://coins-compiler.sourceforge.jp/，参照日時（2014/02/03）
[3] 下村 隆夫，プログラムスライシング技術と応用，共立出版株式会社，(1995).
[4] 中田 育男，コンパイラの構成と最適化第 2 版，朝倉書店，(2009).