5H-02

関数呼出しを含むLLVM IRプログラム用スライシングツールの開発

齋藤　崇雅†　　Nguyen HuuHoang†　　増原　孝昭†　芳賀　博英†
同志社大学理工学部†

1. はじめに

プログラムスライシングはプログラムの依存関係を解析する技術である．プログラムスライシングによりスライシングされたソースコードはプログラムスライスと呼ぶ．プログラムスライシングは Mark Weiser[1]により考案された手法である．プログラムスライシングはソフトウェアのテスト，デバッグ，リバースエンジニアリング，プログラム理解，コード最適化など，広範囲に応用されている．[2]

LLVM は任意のプログラム言語に対応可能なコンパイル基盤であり，LLVM IR は特定のプログラミング言語と CPU アーキテクチャに依存しない中間表現である．異なるプログラミング言語のソースコードをスライシングしたい場合は，その言語に対応したスライシングツールを言語毎に作る必要がある．そこで，LLVM IR のスライスを求めることで，任意のプログラミング言語に対応したスライシングツールが実現できると考えられる．[3]

これまでの研究成果では，LLVM を利用して C 言語のような手続き言語に対して，関数呼び出しをしないプログラムのスライシングを行っていたが，関数呼び出しをするプログラムのスライシングを行っていないという問題点がある．[4]

本報告では，スライシングツールの対応する文法を拡大するためのスライシングアルゴリズムの拡張，関数呼び出しをするプログラムをスライシングするツールの開発の報告を行う．

2. プログラムスライシング

2.1. プログラムスライシングの概要

プログラムスライシングとは，あるプログラムのソースコードにおいて，特定文の指定した変数に影響を与える可能性のある文のみを抽出する手法である．プログラムスライシングにはフォワードスライシングとバックワードスライシングがある．本報告では後者を対象とする．

2.2. LLVM との連携

LLVM は，コンパイラ開発のための基盤ソフトウェアである[4]．LLVM は C 言語をはじめとした複数のプログラムを LLVM IR と呼ばれる中間言語表現に変換し，LLVM IR を元に実行可能プログラムを作成する仕組みとなっている．
本報告では中間表現である LLVM IR を対象にスライシングを行うことで，一度に複数の言語に対応したプログラムスライシングツールを開発することとした．

Implementing program slicing tool for LLVM IR with function calls
Takamasa Saito†, Nguyen HuuHoang†, Masuhara Takaaki†, Haga Hirohide†
Faculty of Science and Engineering, Doshisha University, 1-3, Miyakodani, Tatara, Kyotanabe-shi, Kyoto, 610-0321

3. スライシングツールの構成

昨年度のスライシングツールは，スライシングを行うスライシングエンジン部分とユーザ・インタフェースである GUI 部分に分けられる．スライシングエンジンは LLVM IR を対象としたスライシングを実行する．また，スライシングの基準となる行と変数名のパラメータ入力や，スライシングの結果の出力，スライスと元のソースコードの比較を行うためのハイライト表示を GUI により実現する．

4. スライシングアルゴリズムの概要

4.1. 情報入力

スライシングツールに対象のソースコード，LLVM IR に変換したソースコード，対象の行番号，対象の変数名を入力として与える．

4.2. 依存関係の取得

ソースコードと LLVM IR の対応表を内部で作成し，LLVM IR の解析を行い対象命令に影響を与える可能性のある命令をすべて抽出することで依存関係を取得する．

4.3. ソースコード内の依存関係に変換

作成した対応表を用いて，LLVM IR の依存関係を元となるソースコード内の依存関係に変換する．

4.4. 結果出力

ソースコード内の依存関係からスライス結果となる行番号を出力する．

5. LLVM IR 解析のアルゴリズム

LLVM IR の解析処理は次の 4 段階から構成されている．

(1). Parse
(2). Analyze
(3). Build LLVM IR
(4). Build source code graph

以下にその概要を述べる．

(1) Parse

LLVM IR の各行の処理を解釈して，使用している変数名などを保存する．

(2) Analyze

変数名などを保存して集まった LLVM IR の要素をツリー構造に変換する．

(3) Build LLVM IR graph
ツリー構造に変換したものを LLVM IR の依存関係を有向グラフに変換する.
(4) Build source code graph
LLVM IR の有向グラフを元にソースコードの依存関係の有向グラフを作成する.

6.課題

既存のスライシングツールでの課題として条件分岐を含む文のスライス，関数呼び出しを含む文のスライス，構造体を含む文のスライスが不可である点がある．一番大きな課題は関数呼び出しを含む文でのスライスの不可である.

6.1. 成果

既存のスライシングツールでは，プログラムの依存グラフを見ることが出来ず，スライシングの結果にミスが発生しても特定できなかった.そのため，プログラムの流れを理解できるように，プログラムの依存グラフを生成し可視化を行った.

既存のスライシングツールのアルゴリズムでは関数を含む文のプログラムを下の図１のようなグラフを生成した.

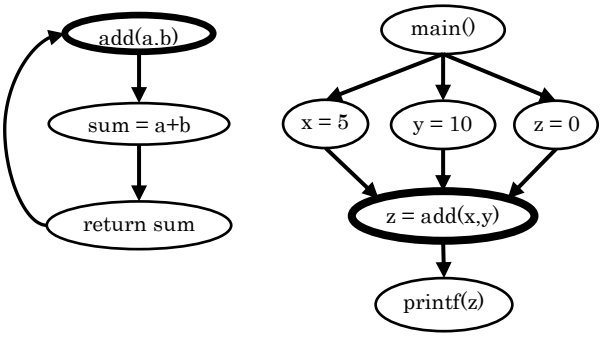

図１　関数呼び出しを含むプログラム依存グラフ

printf(z)ノードの変数 z をスライス基準にスラインシングを行ったときに，図１のグラフより矢印を逆に辿る．しかし，図 1 では z = add(x,y)で add(a,b)関数を呼び出しているが，依存グラフではこの２つのノードの間に矢印が結ばれていないため，スライシングを行うことができないということがわかった．ここで今後の解決手法の手がかりが得られた.

また，プログラムの依存グラフを生成し可視化を行った結果として，条件分岐を含む文のスライスが可能となった.

6.2. 関数呼出しのスライシング

今回関数呼び出しを含む文でのスライスの不可を解決する手法として，プログラムの依存グラフより関数定義のノードから関数を呼び出すノードに矢印を向けることである．これにより関数呼び出しを行う場合でも矢印を逆に辿っていくことでスライスを求められる.

まず関数を呼び出すノードを認識しなければならない．次にその関数を定義しているノードを見つけなければならない．これらの処理は LLVM IR の特徴を利用し，正規表現を使って求められる.

２つのノードを求められると，一方のノードから他方のノードへ矢印を向けると，複数の関数の依存グラフが求められる．具体的に図１から下の図２を求められる.

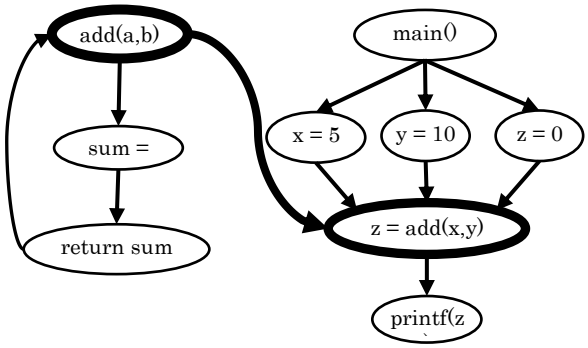

図２　関数呼び出しを含むプログラム依存グラフ

図２のようなグラフを生成することにより，関数呼び出しを含むプログラムのスライスが可能となった.

7.　まとめと今後の課題

スライシングを行う上で，プログラムの処理をグラフの生成により可視化を行なった．生成したことにより，細かいミスを修正することにより，条件分岐を含むプログラムのスライスを可能となった．また関数呼び出しを含むプログラムのスライシングを可能とするための解決手法を考察し，スライスを可能とした.

今後の課題としては，スライシングエンジン部分の開発では，構造体や配列を含んだプログラムのスライシング，インターフェースの開発では，複数のソースコードを跨いだ大規模なプログラムのスライシングの実装である.

参考文献

[1] Mark Weiser. Program Slicing. IEEE Trans. Software Engineering, Vol. 10, No. 4, pp. 352-357, July 1984.
[2] 大崎博基.山本晋一郎.阿草清滋.プログラム理解のための依存関係表示ツール.1996.
[3] 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.2003
[4] Chris Lattner and Vikram Adve, "LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation," Proc. CGO (2004).
[5]Biswaranjan Panda. Sagardeep Mahapatra. Ved Prakash.Slicing of Object-Oriented Software. 2010