5J-09

LLVM IR用スライシングツールとそのGUIインタフェースの実装

齋藤　崇雅†　　栗田　大樹†　　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]

本報告では，多様なプログラミング言語に対応したプログラムスライスを求めるため LLVM IR を用いたツールの実装，またユーザの直感的な操作を可能にする GUI インタフェースの実装について述べる．

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，特定行の指定した変数に影響を与える可能性のある行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．

Program slicing tool for LLVM IR and its GUI Interface Implementation
Takamasa Saito†, Hiroki Awata†, Takaaki Masuhara† and Hirohide haga†
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

本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 部分に分けて開発を行っている．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

スライシングエンジンのアルゴリズムの概要を以下に示す．

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 に変換したコード，スライシングの基準となる行と変数名を入力する．

4.2. 制御・データ依存関係の取得

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 4.1 で入力された対象行と変数名に対応する命令後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制御・データ依存関係を取得する．

4.3. ソースコードの依存関係に変換

作成した対応表を用いて，LLVM IR の依存関係を元のソースコード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. GUI

6.1. GUI の効果

コマンドラインベース（CUI）では，操作性の観点でツールの取り扱いが容易ではなく，元のソースコードとスライスされたコードの比較が難しい．GUI を実装することにより，ユーザの直感的操作が可能になり，スライスを効率よく得られる．

6.2. GUI の実装

実装には，コンパイル不要かつ多くのマシンで実行可能な Python を使用している．また，GUI の実装には Python の GUI ライブラリである PyQt を使用している．

6.3. GUI の機能

スライシングツールの直感的操作の実現のため実装した機能を以下に示す．

- ウィンドウの生成
- スライスを求めるボタンの配置
- ソースファイル保存を行うボタンの配置
- XML 保存を行うボタンの配置
- スライスされたコードのハイライト表示

7. プログラムスライシングツール

今回実装したプログラムスライシングツールの使用方法を説明する． 図 1 は，スライシング結果表示のためのインタフェースである．

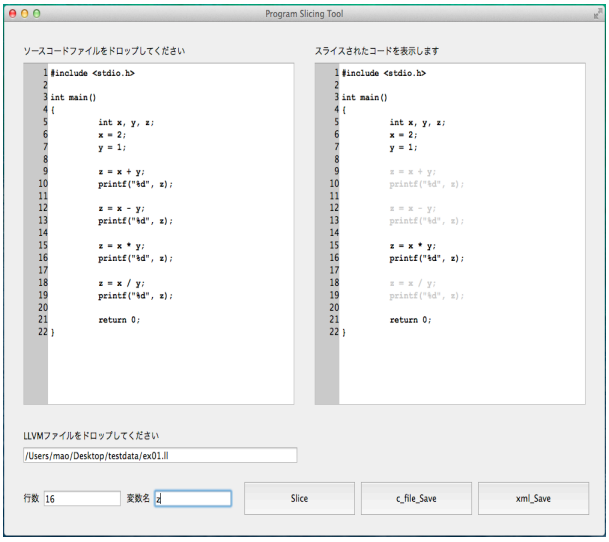

図 1 スライスされたソースコードの表示

図 1 では，左側のテキストボックスにソースコードのファイルと中間言語である LLVM ファイルをドロップする．スライスしたい行数と変数名を入力する．”Slice”と書かれたボタンを押すと，右のテキストボックスにスライスされたソースコードが表示される．例えば図 1 では，16 行目の変数名 z に対応しているソースコードを抽出し表示すると，そのスライスが表示される．図 2 はスライス結果を保存するためのインタフェースである．

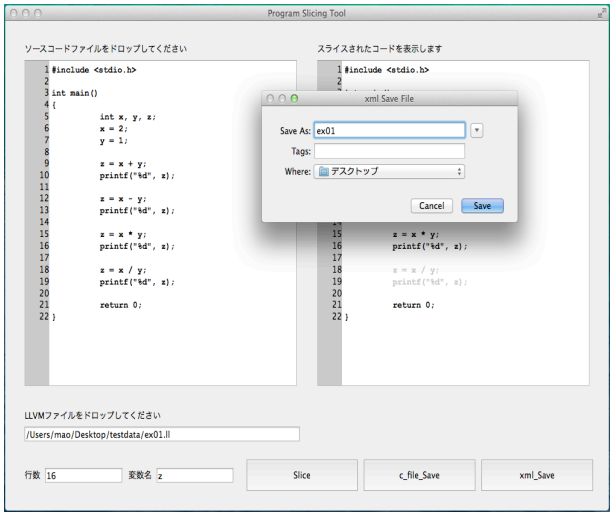

図 2 XML ファイルで保存

図 2 では”C_File_Save”と書かれたボタンを押すと，スライスされたソースコードを C のソースファイルとして保存する．”XML_Save”と書かれたボタンを押すと，スライスされたソースコードを XML ファイルと保存される．

8. まとめと今後の課題

LLVM IR を利用したスライシングツールの実装を行った．これまでコマンドラインベースで行われていたプログラムスライスが GUI インタフェースの実装を行うことにより，ユーザビリティが向上された．

今後の課題として，大規模なソースコードのスライスが得られるか評価実験を行うことが挙げられる．また，そのためにどのようなソースコードを準備するべきか検討する．

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).