5J-03

ソースコード変更時の影響波及範囲解析ツールの実装

梅門　創† 佐々木 穣‡ 菅原 拓海‡ 大久保 建男† 福原 和哉§ 猪股 俊光‡
新井 義和‡ 今井 信太郎‡
†岩手県立大学大学院ソフトウェア情報学研究科 ‡岩手県立大学ソフトウェア情報学部
§いわてものづくり・ソフトウェア融合テクノロジーセンター

1 はじめに

組込みソフトウェア開発では，既存のソースコードに改良や修正を加える場面が多々存在するが，変更後の製品の品質を保証するためには，変更を加えたソースコードと変更の影響が波及した可能性のあるすべてのソースコードに対してテストが必要となる．しかし，実際には影響が及んでいないソースコードへのテストは本来必要ではないため，開発の負担となってしまう．このとき，加えた変更が他のソースコードのどの部分に影響を波及させるのか，その範囲が明らかになれば，テストの負担を最小限にすることができる．

本研究室では，組込みソフトウェア開発で多く用いられている C 言語を対象とし，ソースコード全体に影響を与える可能性が高いと考えられる構文要素の 1 つである変数に着目した，変数への変更が及ぼす影響波及範囲の解析について研究を進めている [1] [2]．本研究では，今までの手法では解析を行えなかった大域変数と，関数間に及ぶ影響に対して新たな解析手法を提案する．

2 影響波及範囲解析ツール

2.1 概要

本研究室では，対象とする C 言語に対してあらかじめ SSA 形式 [3] への変換を行ったソースコードを入力とし，変数の変更による影響波及範囲解析を行うツールを実装している [4]．このツールでは，変数変更時の影響が 1 つの関数内でどれだけ波及するのか解析することが可能である．

しかし，現在ツールに実装している解析手法は，解析の範囲を 1 つの関数のみに絞っているため，関数呼び出しによる複数の関数同士の関係の表現や，引数・戻り値に対する影響波及解析を行うことができない．そのため，関数の呼び出し先など，プログラム内の任意の場所から操作が可能な大域変数に対しても，現在の解析手法では影響波及範囲解析をすることができないという問題がある．

2.2 プログラムスライス

本研究では，関数間に及ぶ影響波及範囲解析と，大域変数に対応した影響波及範囲解析を実現するため，プログラムスライス [5] 技術を用いた新しい解析手法を考案する．

プログラムスライスは，プログラム内の命令文間の依存関係を明らかにする技術であり，明らかになった依存関係を，命令文はノード，命令文の間の依存関係はアークとして表現することで，プログラム依存関係グラフを構築できる．このとき，ある命令文中の変数の定義が他の命令文で使用されている場合，変数を定義している命令ノードから変数を使用している命令ノードに対して実線でデータ依存関係アークが引かれる．また，if 文などの分岐命令によって命令文の実行が制御されている場合，分岐命令ノードから実行が制御されている命令ノードに対して破線の制御依存関係アークが引かれる．

3 大域変数への対応

3.1 大域変数の局所変数化

大域変数を扱うために，以下の (i)〜(iii) にしたがって，大域変数を関数の仮引数と戻り値に局所変数として定義し，解析を行えるようにソースコードの変換を行う．

(i) 関数 `main()` 以外の関数の仮引数と戻り値に大域変数を局所変数として追加し，複数の戻り値があることをタプル式で表す．
(ii) 関数 `main()` では大域変数の定義を局所変数として関数内に定義する．
(iii) 呼び出し側では局所変数化した大域変数に対して呼び出した関数の処理が終了した後の値を受け取るための関数呼び出し命令を新たに追加する．

図 1(a) の大域変数を局所化した例を図 1(b) に示す．(i) より，関数 `f()` の仮引数に変数 g を追加するとともに，return 文をタプル式 `(c_1,g_2)` とする．(ii) より，関数 `main()` では，局所変数 g として定義し，(iii) より，変数 g に対して関数 `f()` の呼び出し関数終了後の値を受け取るための呼び出し関数命令を追加する．

Implementation of Source Code Change Impact Analysis tools
† So UMEKADO, Jo SASAKI , Takumi SUGAWARA, Tateo OOKUBO, Kazuya FUKUHARA, Toshimitsu INOMATA, Yosikazu ARAI, Shintaro IMAI
Iwate Prefectural University Graduate School（†）
Iwate Prefectural University（‡）
Iwate Monozukuri and Software Integration Technology Center（§）

```
 1: g = 0;
 2: int f(x){
 3:   g = g + 2;
 4:   c = x;
 5:   return c;
 6: }
 7: void main(){
 8:   a = 2;
 9:   a = f(a);
10:   b = g;
11: }
```

(a) 通常形式

```
 1: [int,int] f(x_1,g_1){
 2:   g_2 = g_1 + 2;
 3:   c_1 = x_1;
 4:  return (c_1,g_2);
 5: }
 6: void main(){
 7:   g_1 = 0;
 8:   a_1 = 2;
 9:   a_2 = α (f(a_1,g_1));
10:   g_2 = β (f(a_1,g_1));
11:   b_1 = g_2;
12: }
```

(b) 変換後

図 1: プログラムの変換例

3.2 関数間に及ぶプログラム依存関係グラフの構築

SSA 逆変換を行ったソースコードから，関数間における依存関係を表現したプログラム依存関係グラフを構築するために，関数間の関係を表す次の特殊ノードを導入する．

- **entry ノード**
  関数とその関数によって制御されている命令の関係を表す
- **parameter ノード**
  関数呼び出し命令の実引数と関数の仮引数の関係を表す
- **exit ノード**
  関数の戻り値と関数を呼び出した命令の関係を表す

また，局所変数化した大域変数に対して，呼び出した関数終了後の値を受け取るために追加された関数呼び出し命令と通常の関数呼び出し命令はそれぞれ別々に展開する．

図 1(b) の例のプログラムから構築されたプログラム依存関係グラフを図 2 に示す．例えば，図 1(b) の 9，10 行目の関数呼び出し命令のノードでは，変数 a_2 に関数 f() の戻り値を受け取るための関数呼び出しと，変数 g_2 に関数 f() の戻り値を受け取るための関数呼び出しをそれぞれ展開している．

3.3 影響波及範囲の解析

解析を行う命令文とその中の変数をスライシング基準と呼ぶ．スライシング基準に基づき，前向きスライシング [5] を適用して，変更によって影響を受ける可能性のある命令を求める．スライシング基準となる命令から，データ依存関係と制御依存関係を辿っていくことで，影響を及ぼす可能性のある命令の集合を得ることができる．

例えば，図 1(b) のプログラムに対してスライシング基準を (7,{g_1}) として前向きスライシングを行うと，命令の集合 { 2,4,7,9,10,11 } が得られる．

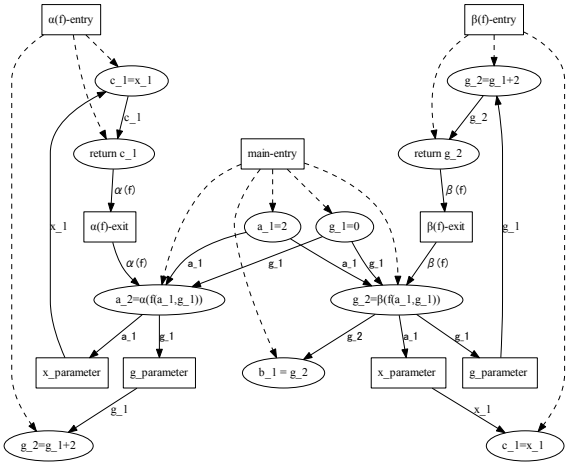

図 2: プログラム依存関係グラフ

4 おわりに

本研究では，大域変数と関数間に及ぶ変数変更時の影響範囲の解析を，プログラムスライシングを用いて実現する手法について提案した．今後の課題として，提案した手法の実装と，現在未対応としているポインタや構造体といった構文に対応することが挙げられる．

参考文献

[1] 大久保建男，猪股俊光，新井義和，今井信太郎 : ソースコード変更時の影響波及解析のための一表現手法，岩手県立大学ソフトウェア情報学部卒業論文 (2015)

[2] 晴澤陽太，猪股俊光，新井義和，今井信太郎 : ソースコード変更時の影響範囲の可視化，情報処理学会東北支部研究報告会（2015）

[3] 中田育男，渡邊坦，佐々政孝，滝本宗宏 : コンパイラの基盤技術と実践 コンパイラ・インフラストラクチャ COINS を用いて，朝倉書店 (2008)

[4] 大久保建男，福原和哉，晴澤陽太，猪股俊光，新井義和，今井信太郎 : ソースコード変更時の影響波及解析を目的とした表現法と影響の可視化法の提案，信学技報，Vol. 115, no.153, SS 2015-19, pp.51–56.(2015)

[5] 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)