2J-03

C ソースコード変更時の影響波及範囲の解析と可視化

高柳 寛樹 †　福原 和哉 ‡　猪股 俊光 †　杉野 栄二 †　新井 義和 †
今井 信太郎 †　成田 匡輝 †
† 岩手県立大学ソフトウェア情報学部　‡ 岩手県立大学共同研究員

1 はじめに

ソフトウェア開発では，既存のシステムに機能を追加したり一部に変更を加える改修作業がよく行われる．ソフトウェアの品質を保証しながら効率的な開発を行うためには，ソースコードを書き換えた影響が波及する範囲を特定する必要がある．影響範囲を明らかにすることができれば，テスト工数の削減，テスト漏れの防止が期待される．そこで，本研究室ではソフトウェア開発を支援するための静的解析技術について研究を進めており，変数の変更による影響波及範囲を解析するシステムを試作した [1]．しかし，解析対象が 1 つの関数内に限定されており，関数をまたいだ影響波及などを解析することができない．本研究では，関数呼び出しやグローバル変数を含むソースコードを対象とする影響波及範囲の解析ならびに可視化を試みた．

2 C ソースコードの中間・グラフ表現

本研究では影響波及範囲解析を行うにあたり，C ソースコードを中間表現に変換し，グラフを作成する．

2.1 C ソースコードから中間表現への変換

本研究が対象とする C ソースコードは `int` 型，`float` 型などの単純型からなるものとする．解析対象の C ソースコードは以下の手順で中間表現に変換しておく．

Step.1 CXML[2] へ変換

Step.2 グローバル変数のローカル変数への変換
グローバル変数は，出現している関数の仮引数に追加する．代入文の左辺に出現している場合，戻り値としても追加する．

Step.3 SSA 形式 (static single assignment form) へ変換

Step.4 `for` 文，`while` 文などの制御構造は `if` 文と `goto` 文を用いた形へ変換

図 1(a) に示す C ソースコードを Step.1〜Step.4 の手順で変換したコードを図 1(b) に示す．説明の都合上 CXML 形式ではなくコード形式で示している．また，@で始まる変数はグローバル変数に対応する．

```
int g = 0;
int func(int a){
   int b = 0;
   g = 2;
   b = a + g;
   return b;
}
int main(){
   int x = 1;
   int y = 0;
   y = func(x);
   if(y > g){
      x = 2;
   }
   g = y;
   return 0;
}
```

(a) ソースコード

```
1   func(a, @g){
2      %a_1 = a;
3      @@g_1 = @g;
4      b_1 = 0;
5      @@g_2 = 2;
6      b_2 = %a_1 + @@g_2;
7      return(b_2, @@g_2);
8   }
9   main(@g){
10     @@g_1 = @g;
11     $tmp_125_1 = Undefined;
12     x_1 = 1;
13     y_1 = 0;
14     ($tmp_125_2, @@g_2)
          = func(x_1, @@g_1);
15     y_2 = $tmp_125_2;
16     if(!(y_2 > @@g_2)) goto L1;
17     x_2 = 2;
18     x_4 = x_2;
19     goto L2;
20     L1:
21        x_4 = x_1;
22     L2:
23        @@g_3 = y_2;
24        return(0, @@g_3);
25  }
```

(b) 中間表現

図 1: ソースコードの変換例

2.2 影響波及範囲解析のためのグラフ

プログラムグラフ [3] をもとに影響波及範囲解析のための表現法を考案した．中間表現に含まれる変数，代入式，分岐条件式，補助ノードをノード集合 P とし，データ依存関係 (dd)，制御依存関係 (cd)，補助ノードを用いた関係 (call，return，merge) をアーク集合 R とする．以上より，影響波及範囲解析のためのグラフを $IG = (P, R)$ とする．

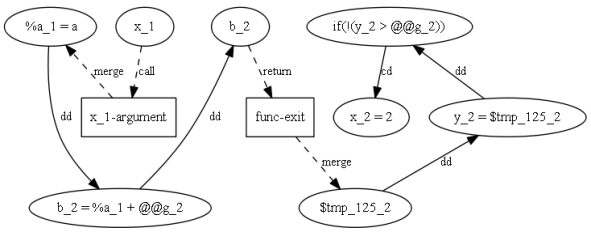

図 2: 図 1(b) の IG におけるサブグラフ

図 1(b) の IG における関数呼び出し関係は，図 2 のように表現できる．14 行目の関数 `func` の呼び出しに伴う引数 `x_1` と 2 行目の関数 `func` の定義 `%a_1 = a` との関係や，7 行目の戻り値 `b_2` と 14 行目の戻り値の代入先 `$tmp_125_2` との関係などからなる．中間表現に現れ

C Source Code Change Impact Analysis and Visualization
†Hiroki TAKAYANAGI, Toshimitsu INOMATA, Ezi SUGINO, Yoshikazu ARAI, Shintaro IMAI, Masaki NARITA
‡Kazuya HUKUHARA
Faculty of Software and Information Science, Iwate Prefectural University(†)
Researcher, Iwate Prefectural University(‡)

る要素のノードは○で，関数呼び出しを示す補助ノードは□で表す．また，○どうしの関係を →，○と□の関係を ⇢ で示す．

3 影響波及範囲解析ツール

変更対象とする変数を基点ノードとして，R のもとで反射推移的閉包より基点ノードから可達なノードの集合を影響波及範囲として導出する．この方法によって，影響波及範囲の解析・可視化を行う静的解析ツールを実装した．本ツールが可視化するのは中間表現から生成される IG のうち，ノード集合 P と中間表現との対応を把握しやすくしたグラフとそのもとでの変更の影響波及範囲である．

本ツールへ図 1(b) のコードと変更対象の変数として 12 行目の x_1 を入力したときの解析結果を図 3 と図 4 に示す．グラフ・中間表現上で影響を受ける箇所が強調表示される．

```
func(a, @g){
%a_1 = a;
@@g_1 = @g;
b_1 = 0;
@@g_2 = 2;
b_2 = %a_1 + @@g_2;
return(b_2, @@g_2);
}

main(@g){
@@g_1 = @g;
$tmp_125_1 = Undefined;
x_1 = 1;
y_1 = 0;
( $tmp_125_2, @@g_2) = func( x_1, @@g_1);
```

図 3: 中間表現での可視化の一部

4 評価

本研究で開発したツールが正しい影響波及範囲を導出可能であるかについて，評価を行った．同じソースコード・変数の変更に対して本ツールと Frama-C[4] による静的解析，Visual Studio デバッガ [5] を用いた動的解析を行い，それぞれが導出した影響波及範囲の比較を行った．その結果，正しい影響波及範囲を導出していることを確かめた．

Frama-C，Visual Studio デバッガと比較して，本ツールは中間表現上で影響波及をワンステップごとに可視化できることや，グラフに影響波及関係を重ねて出力できることから影響波及範囲の追跡支援に効果的であると考えられる．

5 おわりに

本研究では，変数の変更による関数間の影響波及範囲を解析する手法を考案した．さらに，それにしたがって動作する静的解析ツールを実装し，評価を行った．その結果，関数間の影響波及を解析し，可視化することができた．このことから，本ツールを使うことでソフトウェアの品質を保証しながら効率的な開発を行えることが期待できる．

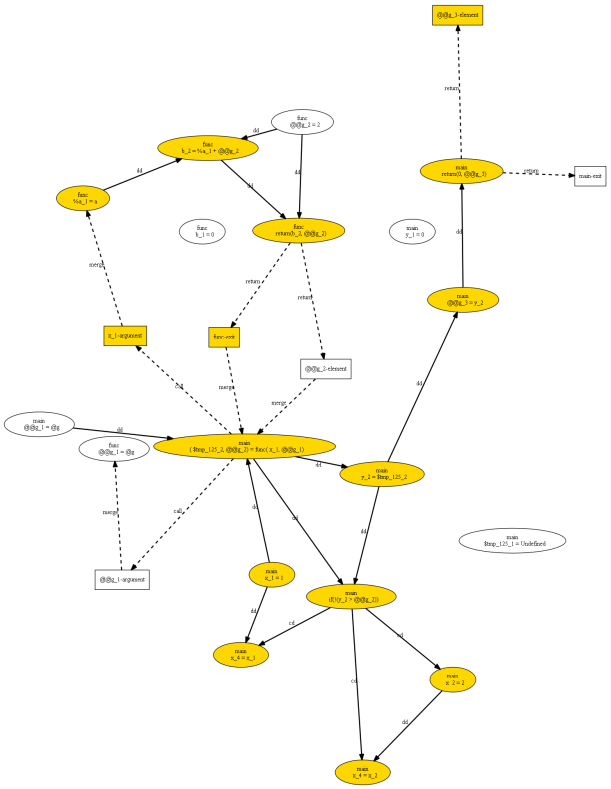

図 4: グラフでの可視化

参考文献

[1] 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，pp.51-56(2015).

[2] 岩間恵梨沙，福原和哉，猪股俊光，杉野栄二，今井信太郎，新井義和，成田匡輝：C ソースコード静的解析のための問い合わせ言語 CXmlPyQuery とその応用，信学技報，Vol.117，No.137，pp.125-131(2017).

[3] 高橋耶真人，福原和哉，猪股俊光，新井義和，今井信太郎：プログラムの関数・変数関係の一表現法，信学技報，Vol.113，No.269，pp.49-54(2013).

[4] CEA LIST：Frama-C，CEA LIST（オンライン），入手先<https://frama-c.com/download.html>（参照 2018-01-08）.

[5] Microsoft：Visual Studio Community - Visual Studio，Microsoft（オンライン），入手先<https://www.microsoft.com/ja-jp/dev/products/community.aspx>（参照 2018-01-08）.