2N－6

ソフトウェア意味構成管理モデルにおけるスライシング技法を用いたプログラム解析

直井 邦彰　高橋 直久
NTT ソフトウェア研究所

1 はじめに

プログラムの改造では，機能追加や変更，再利用により生じる影響の分析など，ソースコードに対する様々な意味的な解析が必要である[1),2)]．従来手法を用いてこれら解析を行うと，解析精度が低いという問題があった[1)]．我々は，精度よく自動的に解析を行うことを目指して，ソフトウェア意味構成管理(SCM)モデル[1)]の構築を進めている．本稿では，まず，SCMモデルでのプログラム解析機能について述べる．次に，これら解析を，プログラム・スライシング技法[3),4)]を用いて実現する手法について議論する．更に，計算経路に沿った解析に対して，動的スライスを適用することにより，解析精度を向上させる手法を提案する．

2 意味構成管理モデル

一般に，同じプログラムを複数の担当者が独立して編集した後，各修正結果を反映させる版を作成すると，プログラムが意図しない動作をすることがある[1)]．このような意味的誤りの発生を調べる技術を，複数修正間での相互干渉検出技術と呼ぶ[2)]．一方，プログラムの修正により影響を受ける文の集合を，影響波及領域と呼ぶ[1)]．

先に提案したSCMモデル[1)]では，影響波及領域の検出と相互干渉の検出を行う．本稿では，編集担当者のプログラム理解を支援することを目指し，SCMに対し，上述の検出機能に加え，以下のプログラム解析機能を保持させる．(a) 計算経路の決定に必要な文集合の解析．(b) 変数の利用目的の理解に必要な文集合の解析．

3 プログラム・スライシング

プログラムを改造する際，ある変数の計算に必要な文集合や，修正による影響波及領域を見つける[2)]．このように，ソースコードから，着目する性質を持つ文集合を抽出する技術をプログラム・スライシングと呼び，抽出したコードをスライスと呼ぶ[3),4)]．求めたい性質に応じて様々なスライシング技法が提案されており，静的/動的，逆方向/順方向，実行可/クロージャからなる互いに直交した概念である3つの属性の組合せとして，8通りのスライスに分類できる[4)]．例えば，静的順方向クロージャ・スライスは，影響波及領域の検出に用いられる．

我々は，これら8通りのスライスを，経路依存フローグラフ[5)]と呼ぶプログラムの有向グラフ表現形式を用いて，統一的な枠組の上で作成する手法を提案している[4),6)]．

Slicing-based program analysis for software semantic configuration management
Kuniaki NAOI and Naohisa TAKAHASHI
NTT Software Laboratories

4 スライシング技法を用いた解析の実現

本章では，2章の(a)，(b)の機能を，スライシング技法を用いて実現する手法について述べる．

4.1 計算経路の決定に必要な文集合の解析

経路の決定に必要な計算を調べる場合，各分岐文で分岐方向を決めるのに必要な文集合を解析すればよい．ところで，我々は，分岐文の条件式に出現する変数に対する静的逆方向クロージャ・スライスを用いて，Infeasible Path検出[5)]に必須な文集合を抽出する手法を提案している[7)]．ここで，本スライスは，分岐文の条件式の計算に必要な文集合を表す．従って，本スライスを作成すれば，分岐方向を決めるのに必要な文集合が求まる．

```
   #define F_VL    1
   #define F_FR    2
   #define R_OVF   -1
   #define MAXSIZE 100
   struct fq{char f_st; int f_id; int f_q;};
   struct df{int d_id; int d_val;};
   struct fq fqt[MAXSIZE];

   int     reassemble(pd)
   struct  df      *pd;
   {
       int i, firstfree;
l1     firstfree = -1;
l2     for(i=0;i<MAXSIZE;i++){
l3         struct fq *pfq = &fqt[i];
l4         if(pfq->f_st == F_FR){
l5             if(firstfree == -1)
l6                 firstfree = i;
l7             continue;
           }
l8         if(pfq->f_id != pd->d_id)
l9             continue;
l10        return(enque(pfq,pd));
       }
l11    if(firstfree < 0)
l12        return R_OVF;
l13    fqt[firstfree].f_st = F_VL;
l14    fqt[firstfree].f_id = pd->d_id;
l15    return(enque(&fqt[firstfree],pd));
   }
```

図1 サンプルプログラム

図1のプログラムのl_8の分岐を定めるのに必要な文集合を，静的逆方向クロージャ・スライスを用いて求めた例を，図2に示す．ここで，図1は，Internet Protocolにおいて，フラグメント化されたデータグラムを再構成するプログラムipreass[8)]の一部を取り出し，完結するように記述を補ったものである．

4.2 変数の利用目的理解のための解析

変数の利用目的を調べる場合，まず，着目する変数に依存する文集合，すなわち，着目する変数に対する静的順方向クロージャ・スライスを求める必要がある．しかし，本スライスは，一般に，経路を辿るために必要な条

```
     int      reassemble(pd)
     struct   df       *pd;
     {
         int i;
l1       for(i=0;i<MAXSIZE;i++){
l2           struct fq *pfq = &fqt[i];
l3           if(pfq->f_st == F_FR) continue;
l4           if(pfq->f_id != pd->d_id) continue;
         }
     }
```

図 2　図 1 のプログラムの l_8 に対するスライス

件文が含まれるとは限らない．このため，スライスを実行させるために必要な文をすべて含む実行可スライス[4]を求める必要がある．すなわち，着目する変数に対する静的順方向実行可スライスを求めれば，着目する変数に依存する文だけでなく，経路を辿るために必要な文を必ず含む文集合が得られる．

```
     int      reassemble(pd)
     struct   df       *pd;
     {
         int i, firstfree;
l1       firstfree = -1;
l2       for(i=0;i<MAXSIZE;i++){
l3           struct fq *pfq = &fqt[i];
l4           if(pfq->f_st == F_FR){
l5               if(firstfree == -1)
l6                   firstfree = i;
l7               continue;
             }
         }
l8       if(firstfree < 0)
l9           return R_OVF;
l10      fqt[firstfree].f_st = F_VL;
l11      fqt[firstfree].f_id = pd->d_id;
l12      return(enque(&fqt[firstfree],pd));
     }
```

図 3　図 1 の firstfree に対するスライス

図 1 のプログラムに出現する変数 firstfree の利用目的を調べるために必要な文集合を，静的順方向実行可スライスとして求めた例を，図 3 に示す．ここで，図 3 のプログラムにおいて，l_2，l_3，l_4，l_7 は，クロージャ・スライスには含まれない文である．

5　特定経路の解析に対する精度向上

影響波及解析や 4 章の解析では，静的スライスを作成する．ところで，ある入力を与えて実行させた経路に対する解析において，静的スライスを用いた場合 (静的手法)，その経路を実行しない文も解析結果に含むため，精度が低下する．そこで，実行した文集合と静的スライスとの積集合を用いると (実行文参照手法)，特定経路に対応した文集合だけが求まると期待できる．しかし，この手法でも，経路に沿って解析した場合には検出されない文を結果に含むことがある．ところで，動的スライスは，実行時に発生する依存関係を用いて解析した結果求まる文集合である[4]．従って，動的スライスを用いれば (動的手法)，経路に沿って解析した際には検出されない文の除去が可能となり，検出精度が向上する．以下に，具体例を示す．

5.1　修正による影響波及領域の精度向上

例えば，図 1 のプログラムの先頭に図 4 の d を，図 1 の l_9 と l_{10} の間に図 4 の l_a と l_b をそれぞれ挿入した修正プログラムを考える．このとき，静的手法では，l_3 から l_{15} までのすべての文に影響が及ぶと判定される．そして，l_4 の条件式が常に真となる入力を与えて修正プログラムを実行した場合，実行文参照手法では，l_{13} を始めとする文が影響を受けると判定される．しかし，ここで，動的手法を用いると，実行した文集合には修正部分を含まないため，修正による影響を受けないと判定される．すなわち，検出精度の向上が達成される．

```
d  #define F_FL    3
la     if(pfq->f_st == F_FL) continue;
lb     if(pd->d_val < 0) pfq->f_st = F_FL;
```

図 4　図 1 のプログラムに対する修正行

5.2　計算経路の決定に必要な文集合解析の精度向上

4.1 節の例において，l_4 の条件式が常に偽となる入力を与えた場合を考える．実行文参照手法による解析結果では図 1 の l_4 が含まれるが，上述の入力では l_8 の条件式の真偽に l_4 は依存しない．5.1 節と同様に，l_8 に対する動的逆方向実行可スライスを求めることにより l_4 を除去でき，検出精度の向上が図れる．

6　おわりに

意味構成管理モデルにおけるプログラム解析機能と，スライシング技法による実現法について述べた．更に，特定経路に対する解析を行う際，動的スライスを用いて解析精度を向上させる手法について述べた．今後，実現機能の詳細について検討を進め，更に，提案手法の有効性を確認するために，構築中のプログラム解析プロトタイプシステム[6]を用いて実験的に評価を進める予定である．最後に，日ごろ御指導御討論頂く，広域コンピューティング研究部後藤滋樹部長，伊藤正樹リーダはじめ広域コンピューティング研究部の皆様に感謝致します．

参　考　文　献

1) 直井邦彰, 高橋直久：意味構成管理システムを用いた修正プログラムにおける相互干渉の検出, 情報処理学会第 45 回全国大会, 6T − 1 (Oct. 1992).
2) 直井邦彰, 高橋直久: 経路依存フローグラフを用いたプログラム解析, NTT R&D,Vol.42,No.8,pp.1007–1016(Aug.1993).
3) Weiser,M. : Program Slicing,*IEEE Trans.Software Engineering*, Vol.SE-10, No.4, pp.352–357 (Jul. 1984).
4) 直井邦彰, 高橋直久：経路依存フローグラフを用いたプログラム・スライシング, 電子情報通信学会ソフトウェアサイエンス研究会 (Sep.1993).
5) 直井邦彰, 高橋直久：経路依存フローグラフを用いた Infeasible Path 検出法, 電子情報通信学会論文誌, Vol. J76 − D − I, No. 8, pp. 429 − 439 (Aug. 1993).
6) 直井邦彰, 高橋直久：経路依存フローグラフを用いたプログラム解析システム, 情報処理学会第 47 回全国大会, 5D − 9 (Oct. 1993).
7) 直井邦彰, 高橋直久：経路依存フローグラフを用いた Infeasible Path 検出における計算量削減法, 情報処理学会第 48 回全国大会, 6G − 2 (Mar. 1994).
8) Comer,D.E. and Stevens, D.L. : Internetworking with TCP/IP, Vol.II, Prentice-Hall, Inc. (1991).