6G－2

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

直井 邦彰　高橋 直久
NTT ソフトウェア研究所

1 はじめに

どのような入力を与えても決して実行されることのない計算経路は実行不可能経路 (Infeasible Path : IP)[1] と呼ばれる．IP 検出により，テストデータの効率的な作成や，プログラムの修正による影響波及領域の，より厳密な検出が可能となる[2]．我々は，先に，経路依存フローグラフ (PDFG) を用いた IP 検出法を提案するとともに[1]，プロトタイプにより計算時間を評価した[3]．本手法では，各計算経路に対して経路条件 (PC) と呼ぶ式を作成した後，各 PC が充足可能か判定する．そして，経路 P の PC が充足可能でないとき，P を IP と判定する．しかし，本手法では，分岐文の数が多くなると，すべての経路に対する充足判定に必要な計算を実用的な時間内に行えないという問題がある．このため，本稿では，以下の計算量削減法を提案する．

(1) プログラム・スライシング技法[4] を用いて，PC を作成すべき経路の数を削減する．
(2) 変数に着目して PC を長さの短い式に分割し，各式について充足可能か判定することにより，充足判定の計算量を削減する．
(3) 索表計算 (tabulation)[5] を用いて，同一の式に対する充足判定の計算を重複させない．

2 PDFG を用いた IP 検出法

PDFG を用いた IP 検出法では，まず，プログラムを PDFG により表現する．ここで，PDFG は，プログラムにおけるデータ，経路，制御の 3 つの依存関係を表現した有向グラフである．図 1 のプログラムに対する PDFG の例を，図 2 に示す．ただし，図 1 において，b_1 から b_4 は，分岐文の識別子を表す．次に，PDFG の各ノードに IP 検出のための意味規則[1),3)] を与え，一般データフロー計算モデル[6] に基づき PDFG を解釈実行させる．この実行では，各経路に対する PC を計算する．ここで，経路 P に対する PC は，P を実行させるために入力変数が満たすべき条件を表す．更に，プレスブルガー文 (P 文) の真偽判定機構[7] を用いて，PC が充足不可能かを判定する．例えば，図 1 のプログラムにおいて，b_4 の判定式のみ真となる経路 P_{fftf} の PC C_{fftf} は，$\neg(a < 0) \wedge \neg(b < 0) \wedge \neg(c > 0) \wedge (a+1 < 0)$ と求まる．また，P 文 $\forall a \forall b \forall c[\neg C_{fftf}]$ は真と判定されるため，C_{fftf} は充足不可能，すなわち，P_{fftf} は IP と判定される．

Computation reduction in infeasible path detection that uses a path dependence flow graph
Kuniaki NAOI and Naohisa TAKAHASHI
NTT Software Laboratories

本手法では，ある経路に対する PC の長さを l とすると，PC の充足判定には $O(2^l)$ 以上の計算量が必要となる[7]．また，分岐文の数を n とすると，最大 2^n 本の経路に対する PC の充足判定が必要となる．更に，n が増加すると，l も増加する．

```
    int sample(a,b,c)
    int a,b,c;
    {
        int p;
b1      if (a < 0) {p = a;} else {p = -a;}
b2      if (b < 0) {a = b;}
        a++;
b3      if (c > 0) {p += c;} else {p -= c;}
b4      if (a < 0) {p--;} else {p++;}
        return(p);
    }
```

図 1　サンプルプログラム

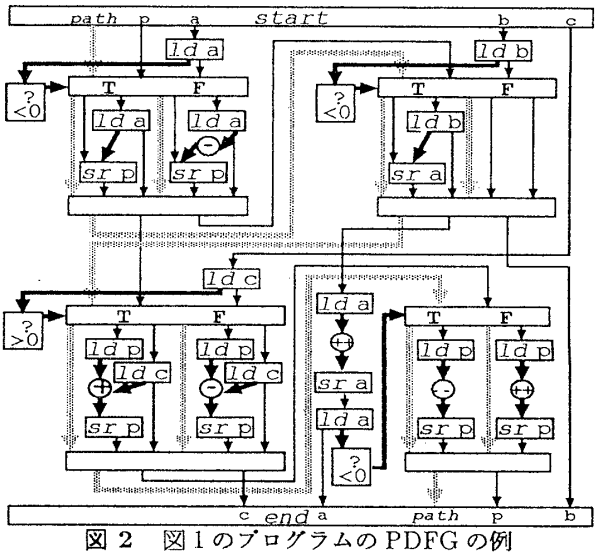

図 2　図 1 のプログラムの PDFG の例

3 IP 検出の計算量削減法

3.1 スライシング技法による経路の絞り込み

まず，各分岐文 b_i ごとに，b_i の条件式に出現する変数に対する逆方向静的スライス[4] $S(b_i)$ を求める．例えば，図 1 のプログラムにおける 4 つの分岐文に対してスライスを求めた例を，図 3 に示す．ここで，2 つのスライス $S(b_i)$，$S(b_j)$ がどちらも入力変数 u を持つとき，$S(b_i)$ と $S(b_j)$ とはスライス共有関係があると言う．そして，スライス共有関係の連鎖により互いに到達しあえるスライスの集合を求め，集合の各要素を 1 つのプログラムに統合する．図 3 の例では，スライス集合 $\{S(b_3)\}$ と $\{S(b_1), S(b_2), S(b_4)\}$ が求まり，後者について統合したプログラムを図 4 に示す．

本手法では，IP 判定に必要なプログラムを抽出している．そのため，一般に，プログラムに出現する分岐文の数 n より考慮すべき分岐文の数 m は小さいので，判定すべき経路の数が 2^n から 2^m に減少する．

```
int sample(a) int a; {if (a < 0) ;}
  (a) 文b1の条件式に対するスライスS(b1)
int sample(b) int b; {if (b < 0) ;}
  (b) 文b2の条件式に対するスライスS(b2)
int sample(c) int c; {if (c > 0) ;}
  (c) 文b3の条件式に対するスライスS(b3)
int sample(a,b) int a,b; {
  if (b < 0) {a = b;} a++; if (a < 0) ;
}
  (d) 文b4の条件式に対するスライスS(b4)
```

図 3　図 1 のプログラムの各条件式に対するスライス

```
 int sample(a,b) int a,b; {
b1   if (a < 0) ;
b2   if (b < 0) {a = b;}
     a++;
b4   if (a < 0) ;
}
```

図 4　$S(b_1), S(b_2), S(b_4)$ を統合したプログラム

3.2　PC の分割による計算量の削減

3.1節で抽出したプログラムの各経路に対して，PC を作成する．このとき，PC の長さを短くできれば，充足判定のための計算量が削減できる．ここでは，まず，3.1節におけるスライス共有関係と同様の手法により PC を複数の式に分割する．次に，各式に対してそれぞれ充足判定を行う．ここで，PC の項 t_i と t_j に変数 u が出現する場合，t_i と t_j は項共有関係があると言う．そして，経路 P の PC を，項共有関係の連鎖により互いに到達できる項のみから構成される式 F の論理積により表現する．更に，P 文の真偽判定機構を用いて，各 F が充足可能か否かを判定する．もし，ある論理式が充足不可能であれば，PC は充足不可能となり，P は IP と判定される．例えば，図 3 のプログラムにおいて，b_4 の判定式のみ真となる経路 P_{fft} の PC $\neg(a<0) \wedge \neg(b<0) \wedge (a+1<0)$ は，$F(a) = \neg(a<0) \wedge (a+1<0)$，$G(b) = \neg(b<0)$ とおいた時，$F(a) \wedge G(b)$ と表せる．ここで，$F(a)$，$G(b)$ とも充足可能であるため，PC は充足可能，すなわち，P_{fft} は IP でないと判定される．

本手法では，項共有関係に着目して PC を変形し，長さの短い式に対する充足判定を複数回行うことにより，充足判定の計算量を削減する．

3.3　索表計算による重複した計算の削除

3.2節に示す手法では，複数の経路に対して同一の式に対する充足判定を行う場合が発生する．そのため，充足判定に索表計算を適用することにより，複数回，同一の式に対する判定を行うことを防ぐ．すなわち，各式の判定の際，まず，テーブルを参照して既に判定したか調べる．もし，そうであれば，テーブルに記録されている値を利用し，そうでなければ P 文の真偽判定機構を用いて充足判定を行う．更に，判定終了後には，式と判定結果をテーブルに記録する．例えば，図 4 のプログラムでは，すべての分岐文が真となる経路 P_{ttt} と，b_1 のみ偽となる経路 P_{ftt} では，どちらも，式 $(b<0) \wedge (b+1<0)$ に対する充足判定が必要である．しかし，索表計算を適用すると，P_{ttt} と P_{ftt} のいずれか先に行う判定 1 回だけでよくなる．

4　考察

IP 検出の計算量は，2章に示した通り，分岐文の数 n および PC の長さ l の指数オーダ以上となる．ここで，3.1節と 3.2節の手法を用いると，n と l は小さくなる．また，3.3節の手法を適用すると，n を小さくするのと同様な効果が得られる．ところで，3.1節のスライス共有関係の連鎖により到達できる集合は，多項式時間で検出できる．例えば，ノードが入力変数を表し，同じスライスに出現する入力変数間で到達しあえるアークを持つグラフ $\mathcal{G}$ を考え，$\mathcal{G}$ の連結関係を調べればよい．3.2節の項共有関係についても同様である．更に，3.1節におけるスライスの作成や統合，3.3節における表検索も，多項式時間で行える．すなわち，提案手法実現のために増加する計算量は多項式オーダとなる．従って，IP 検出の計算量の削減が達成される．

5　おわりに

PDFG を用いた IP 検出法における計算量削減法について報告した．本削減法では，スライシング技法の適用により，PC を作成すべき経路の数を削減する．更に，PC の分割と索表計算の適用により，PC の充足判定の計算量を削減する．今後，計算量についてより厳密に評価するとともに，提案手法の有効性を確認するために，プロトタイプシステム[3]に提案手法を組み込んで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，ソフトウェア基礎技術研究部後藤滋樹部長，伊藤正樹リーダはじめソフトウェア基礎技術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：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).
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) 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).
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) Bird,R.S. : Tabulation Techniques for Recursive Programs, *ACM Computing Surveys*, Vol.12, No.4, pp.403 − 417 (Dec. 1980).
6) 高橋直久, 鈴木英明, 直井邦彰, 福田晴元：データフロー計算の一般化 — ソフトウェア・リエンジニアリングにおける複雑な情報の構造化と理解を目指して —, ソフトウェア科学会第 10 回全国大会 (Jun.1993).
7) 東野輝夫, 北道淳司, 谷口健一：整数上の線形制約の処理と応用, コンピュータソフトウェア, Vol. 9, No.6, pp. 31 − 39 (Jun. 1992).