5AE－4

依存関係解析に基づく開発支援システムへの動的スライス抽出機能の追加

西松顯　　井上克郎

大阪大学大学院基礎工学研究科

1　まえがき

大規模，複雑なプログラムではデバッグ時に誤りの原因の特定に時間がかかってしまい，結果としてプログラム開発の効率が悪くなる．このような場合に，誤りに関係があると思われる部分だけをプログラムから抽出するような機能があれば，開発者はその部分だけに注目することができる．その結果，プログラム中の誤りの位置を発見する負担を軽減することができ開発効率の向上が期待される．これまでにプログラム文の依存関係に基づいた開発支援システムを試作した [3].

本稿では，このシステムに動的スライスを抽出する機能を追加する事で，特定の入力におけるプログラムの誤り原因の特定の際に注目するべき文をさらに限定することができ，さらに開発効率の向上が期待できる．

2　プログラムスライス

プログラムスライスとは，直観的には，プログラム中のある文に対して，その文で参照しているある変数の内容に影響を及ぼしうる全ての文の集合のことである．プログラムスライスには，ソースプログラムを静的に解析して得られる静的スライス (Static Slice) と，ある入力データに対するプログラムの実行系列を解析して得られる動的スライス (Dynamic Slice) がある．

動的スライスを計算するアルゴリズムは文献 [1, 2] で提案されている。本稿では文献 [1] に基づき，依存関係としてデータ依存関係と制御依存関係だけを用いて動的スライスを計算する．依存関係を調べる対象は，文献 [1] ではDDG(Dynamic Dependence Graph) であるが，本稿では実行された文の列である実行系列を対象とする．実行系列中の注目している実行時点のある変数に直接/間接的に依存関係のある実行時点に対応する文の集合を動的スライスとする．ここでは命令同一関係 [2] を導入していないために，得られた動的スライスは必ずしも実行可能なものとはならない．

入力を持つプログラムが誤りを含む時，その誤りはどのような入力に対しても誤動作を起こす場合と，特定の入力に対してのみ誤動作を起こす場合がある．前者の場合は，静的スライスを用いる事でデバッグ対象となる文を限定できるが，後者の場合は，静的スライスでは任意の入力が対象となっているため，誤りに関係のない実行されなかった文もデバッグ対象となる可能性がある．それに比べ，動的スライスは実行されなかった文がスライスに含まれる様な事はないので，静的スライスを用いるよりさらにデバッグ対象となる文を限定できる．

3　システムの概要

3.1　対象とする言語の仕様

本システムが対象とする言語は，Pascal のサブセットである．その仕様は，文として条件文，ループ文，代入文，入出力文，手続き呼出文，複合文を持ち，手続き，関数は自己再帰呼び出しおよび相互再帰呼び出しもできる．また，その引数には値渡しと変数渡しの二種類がある．データ型はスカラー型のみを扱い，ポインタ型は扱わない．

3.2　システム構成

今回拡張したシステムは以下の 5 つの部分により構成されている．システム構成図を図 1 に示す．

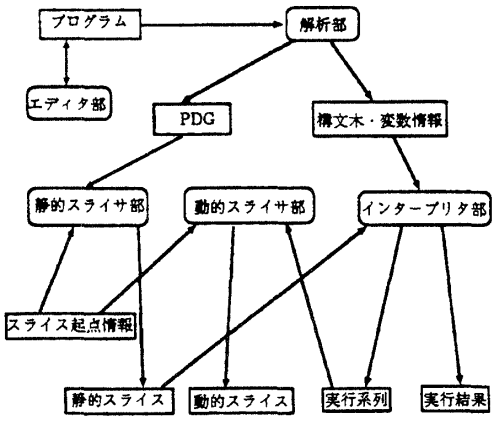

図 1: システム構成

Functional extension of the development support system Using dependence relations of program statements
Akira Nishimatsu and Katsuro Inoue
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka, 560 JAPAN.

- **エディタ部**　Pascal のサブセットで書かれたプログラムのロード，編集，セーブが行える．
- **解析部**　プログラムの意味解析，依存解析を行い PDG を作成する．
- **インタープリタ部**　プログラムの実行を行なう．動的依存解析に用いるために実行系列を保存する．実行では，ステップ実行やブレークポイントの指定がおこなえる．
- **静的スライス部**　作成された PDG を元に静的スライスを計算する．
- **動的スライス部**　インタープリタ部により保存された実行系列を元に動的スライスを計算する．

3.3 使用手順

本システムを用いて開発および保守をする時の使用手順を以下に述べる．

- コード化が終了したプログラムをシステムに入力しコンパイルする．このとき，構文エラーが発見されればコードを修正する．
- プログラムを実行する．出力が誤っているなどの誤動作が発見された場合には，その誤りが特定の入力の場合のみ起きるのか，全ての入力の場合に起きる場合かを確認する．
- 特定の入力のみの場合は動的スライスを抽出し，それをデバッグの対象とする．
- 全ての入力の場合は静的スライスを抽出し，それをデバッグの対象とする．

3.4 実行例

本システムを用いて静的スライス及び動的スライスを抽出した例を図2と図3に示す．

図 2と図 3は，5 個の整数を入力として受け取り，その最大値，最小値を出力するプログラムである．それぞれ，スライシング基準は静的スライスの場合は最小値 *min* を出力する文 writeln('MIN='min) であり，動的スライスの場合は，入力を a[1]=1,a[2]=2,a[3]=3,a[4]=4,a[5]=5 とした時の実行系列における実行時点 writeln('MIN='min) である．動的スライスは実行された文ののみがスライス計算の対象となる為，静的スライスよりもスライスに含まれる文がさらに限定できる．

4 あとがき

依存関係解析に基づく開発支援システムに動的スライス抽出機能を追加した．これによりプログラムの誤りにより動的スライス，静的スライスを使い分けることで開発効率の向上が期待できる．

今後の課題として，言語仕様の拡張やシステムの評価が挙げられる．

```
program max_min(input,output);
var i,max,min:integer;
    a:array[1..5] of integer;
begin
  i:=1;
  writeln('Input 5 numbers');
  while i<=5 do
  begin
    readln(a[i]);
    i:=i+1
  end;
  min:=a[1];
  max:=a[1];
  i:=2;
  while i<=5 do
  begin
    if min>a[i] then min:=a[i] ;
    if max<a[i] then max:=a[i] ;
    i:=i+1
  end;
  writeln('MAX=',max);
  writeln('MIN=',min)
end.
```

図 2: 静的スライス

```
program max_min(input,output);
var i,max,min:integer;
    a:array[1..5] of integer;
begin
  i:=1;
  writeln('Input 5 numbers');
  while i<=5 do
  begin
    readln(a[i]);
    i:=i+1
  end;
  min:=a[1];
  max:=a[1];
  i:=2;
  while i<=5 do
  begin
    if min>a[i] then min:=a[i] ;
    if max<a[i] then max:=a[i] ;
    i:=i+1
  end;
  writeln('MAX=',max);
  writeln('MIN=',min)
end.
```

図 3: 動的スライス

参考文献

[1] H.Argawal and J.Horgan. "Dynamic Program Slicing." *SIGPLAN Notices*, Vol. 25, No. 6, pp. 246–256, June 1990.

[2] B.Korel and J.Laski. "Dynamic Program Slicing". *Information Processing Letters*, Vol. 29, No. 10, pp. 155–163, 1988.

[3] 佐藤 慎一, 飯田 元, 井上 克郎. "プログラムの依存関係解析に基づくデバッグ支援システムの試作". 情報処理学会論文誌, Vol. 37, No. 4, pp. 536–545, April 1996.