7M－4

プログラムスライスの抽出・実行機能を組み込んだデバッグ支援システムの試作

佐藤 慎一† 飯田 元† 井上 克郎† 鳥居 宏次†‡

†大阪大学 ‡奈良先端科学技術大学院大学

1 まえがき

近年のソフトウェアの大規模化に伴ってその複雑さも増加している．そのためプログラムの開発や保守の段階で，対象となるプログラム全体を把握しながら作業を行なうのは容易ではない．プログラムのある一部の文や変数にのみ着目し，それに関係する部分だけについてプログラムを読んだり，編集したり，または実行，デバッグし，他の関係のない部分については作業の対象外にすることができるならば，プログラマーの作業効率は向上するであろう．

本稿では，プログラムのある一部の文や変数にのみ着目して，それに関連するプログラムの断片（プログラムスライス）を表示したり，それを別の一つのプログラムとして抽出するデバッグ支援システムの試作を行なった．

2 プログラムスライス

プログラムスライス [4] 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．

我々が試作したシステムでは，入力データが不用で，抽出したプログラムの断片を一つのプログラムとして扱うことができるなどの特徴を持つ静的スライスを用いた．本稿ではこれ以降，静的スライスのことをスライスと呼ぶ．

スライスを計算するアルゴリズムは，[2, 3] に基づいている．プログラム中の各文について到達定義集合 (Reaching Definitions)[1] を計算して，それを基にプログラム依存グラフ (Program Dependence Graph，略して *PDG*)[2, 3] を作成する．*PDG* の各節点はプログラム中の各文もしくは式を表し，辺は節点間の依存関係を表す．*PDG* の中で，注目している節点から到達可能な節点の集合をスライスとする．[2, 3] のアルゴリズムでは関数の引数に値渡ししか許されなかったが，本システムでは変数渡しのものも許すように拡張した．

Debug support system with program slice

Shinichi Sato†, Hajimu Iida†, Katsuro Inoue† and Koji Torii†‡

†Osaka University

‡Nara Institute of Science and Technology

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，大阪大学基礎工学部情報工学科三年時のコンパイラ作成演習で使用する言語を基にしており，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．

このように，本システムで扱う言語は単純化されてはいるものの，プログラミング言語における基本的な文の構造は全て含んでいる．

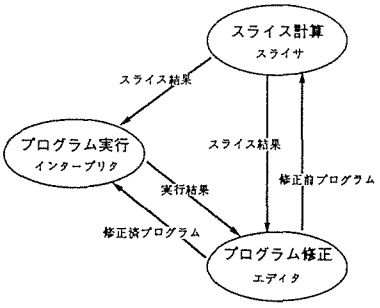

図 1: システム使用時の作業プロセス

3.2 作業プロセス

本システムを用いて開発および保守をする時の作業プロセスを 図 1 に示す．まず開発者は，コード化が終了したプログラムをシステムに入力し，実行させる．その結果構文エラーが発見されればコードを修正し，その修正したプログラムを実行させる．出力が誤っているなどの実行時エラーが発見された場合には，そのエラーに関係のある部分のみをスライスとして抽出し，それを一つのプログラムとみなしてデバッグの対象とする．このようにプログラムのソースコード全体を扱うのではなく，エラーに関係のある部分だけを扱うことにより，開発者は不必要な部分を参照する必要がなくなり，負担が軽減されるため，結果として開発効率の向上が期待される．

また，本システムによるスライス結果は元のプログラ

ムの一部の機能を実行するようなプログラムとして保存できるのでプログラムの部品化に役立てることができる．

4 システムの構成

システムの構成およびその中でのデータの流れは 図2のようになっている． 図2において，楕円はシステム内で処理を行なう部分を表し，四角形はそれらへの入力，もしくはそれらからの出力を表す．システムにプログラムを入力すると，パーザ部でその構文解析を行なうとともに，*PDG* の節点とプログラム中の各文との対応付けを行なう．それらのデータをもとにアナライザ部では，到達定義集合の計算および *PDG* の作成を行なったり，インタープリタ部での実行のための情報確保を行なう．これらの情報はスライサ部やインタープリタ部に伝えられる．

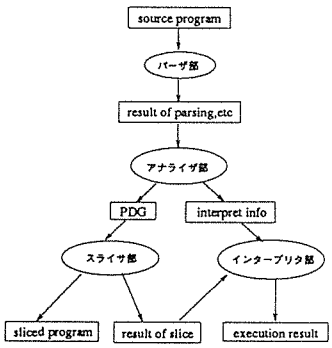

図 2: システムの構成

インタープリタ部はプログラムの実行を行なう．その際，通常の実行だけでなく，トレース実行やスライス結果だけの実行なども行なえる．

スライサ部はスライスを計算する．作業者はプログラム中の文とそれに含まれる変数を指定することによってそれらに関するスライスを得ることができる．その際，通常のスライスだけでなく，変数の定義が影響を与える可能性のある文の集合 (forward slice) も計算することができる．また，スライス結果からさらにスライスを計算することや，ある文に直接影響を与える文，さらにその文に直接影響を与える文というようにスライスを段階的に得ることもできる．

スライサ部ではスライスを一つのプログラムとして保存することもできる．その際スライスが文法的に正しくなるように，スライスを抽出した結果必要になった文の区切り記号を付加したり不必要になった手続きの宣言部を削除するなどの動作を行なう．こうすることによりプログラムの一部の機能を実現するプログラムを生成できる．

本システムを用いてスライスを抽出した例を 図3 に示す．このプログラムは配列に格納されたテキストデータの文字数，語数，行数を計算して出力するプログラムであるが，そのうちの行数を出力する文（下線文）における行数を格納する変数 `line` に関するスライスを抽出した例である．スライス部分は反転表示されている部分である．変数 `line` の出力値が正しくない場合にはこの部分にバグが存在することがわかる．このように，スライスを抽出してその部分のみを参照することにより，デバッグ作業の範囲を大幅に限定できる．

```
program wordcount(in,out);
var     in:array[0..1000] of char;
        i:integer;
        letter,word,line:integer;
        isinword:boolean;
        EOL,EOF:char;
begin
  [illegible]
  letter := 0;
  word := 0;
  [illegible]
  isinword := false;
  [illegible] begin
    letter := letter + 1;
      [illegible]
        [illegible]
      if (in[i] <> ' ') and (in[i] <> EOL) then
      begin
        if isinword = false then
        begin
          isinword := true;
          word := word +1
        end
      end else
        isinword := false;
    [illegible]
  end;
  writeln('letter =',letter);
  writeln('word   =',word);
  writeln('line   =',line)
end
```

図 3: スライスの例

5 あとがき

プログラムの一部をスライスとして抽出し，それを対象としてデバッグを行なったり，スライス結果を一つのプログラムとして保存できるようなシステムを試作した．本システムを使用することにより大規模なプログラムの開発やデバッグ，保守を容易に行なうことができるようになろう．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．

参考文献

[1] Aho, A.V., Sethi, S. and Ullman, J.D.:“Compilers : Principles, Techniques, and Tools”, Addison-Weseley, 1986.

[2] Horwitz, S. and Reps, T.:“The Use of Program Dependence Graphs in Software Engineering”, Proceedings of the 14th International Conference on Software Engineering, pp. 392–411(1992).

[3] 植田 良一，練 林，井上 克郎，鳥居 宏次:“再帰を含むプログラムの依存関係解析とそれに基づくプログラムスライシング”，信学技報，SS93–24, pp. 33–40(1993–09).

[4] Weiser, M.: “Program Slicing”, Proceedings of the Fifth International Conference on Software Engineering, pp. 439–449(1981).