プログラムスライシング技術を用いたJavaソフトウェアのデバッグツールの試作

本橋　強志　　岩澤　京子
拓殖大学　工学部　情報工学科

1. はじめに

プログラミングにおいてバグを的確に検出したり，修正したりすることは，そのプログラムの完成度を向上することに直結する．デバッグは，プログラミングにおいて重要な工程であるが，大きな労力を要する工程でもある．これは，プログラムからバグの原因を特定することが困難であることに起因する．

ここで，私は，プログラムスライシング[1]という技術に注目した．この技術をデバッグに用いることで，狭い範囲にデバッグの対象を限定することができる．本研究では，プログラムスライシング技術を用いることで，デバッグにかかる労力を軽減するツールを試作することを目的とする．

本研究のデバッグツールは，ユーザが入力したプログラムに対して，プログラムにおける依存関係[2][3]を解析する．まず，注目する変数と対象とする範囲をユーザに指定させる．そして，この変数の値に影響を与えている可能性がある処理部分を，プログラムスライシングの結果として表示する．この結果によって，ユーザはプログラムの依存関係を知ることができる．

2. デバッグツールの構成と機能

図1にデバッグツールの全体構成図を示す．デバッグツールは，2つのデータと4つの処理機能を持っている．

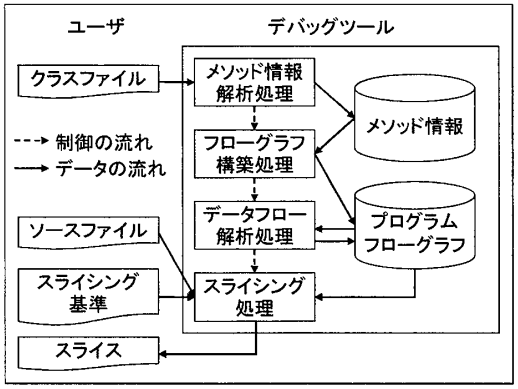

図1　デバッグツールの全体構成図

デバッグツールへの入力は，クラスファイル，Javaソースファイル，およびスライシング基準とする．スライシング基準とは，スライシングのときに注目する変数と対象とする範囲である．ここで，注目する変数は，変数の参照箇所をソースプログラムの行番号と変数名で指定させる．そして，対象とする範囲は，メソッド単位で指定させる．

デバッグツールは，クラスファイルのバイトコードに対して，スライシング基準に従ってプログラムの依存関係を解析する．そして特定の変数に影響を与える可能性のある処理部分のソースコードをユーザに出力する．この出力によって，ユーザはプログラムの依存関係を知ることができるので，デバッグにかかるコストを軽減することができる．このときのデバッグツールの入出力の例を図2に示す．

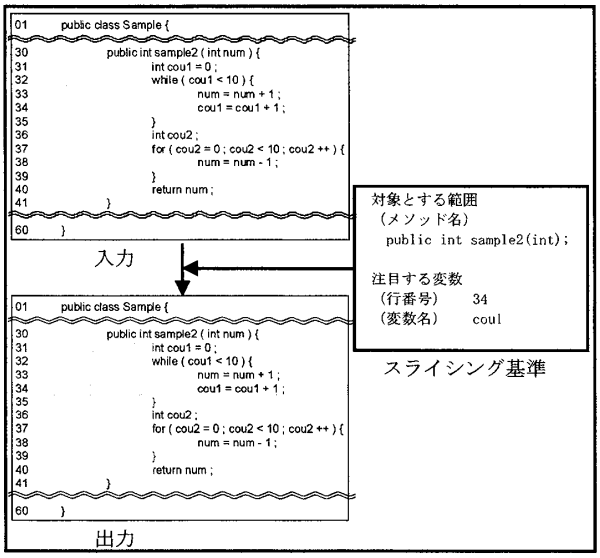

図2　プログラムスライシング処理の入出力例

3. プログラムフローグラフのデータ構造

プログラムフローグラフとは，プログラムにおける制御依存関係を表現するグラフデータである．基本ブロック，基本ブロック同士のリンク，命令情報，命令情報同士のリンク，先頭ブロックへのリンクといった情報によってグラフデータを定義している．

プログラムフローグラフのノードである基本ブロックとは，命令の集合によってプログラムの制御を表現するデータである．ノードは，1つ以上の命令情報，先行ブロックへのリンクの集合，後続ブロックへのリンクの集合の3つのデータを保持することで，プログラムの制御の構造を定義している．

データフローのノードである命令情報とは，バイトコードの命令文について，命令番号，命令の内容，命令の対象，先行命令へのリンクの集合，後続命令へのリンクの集合の5つのデータを保持することで，変数の参照と定義の順序を定義している．

Prototyping of Debugging Tool for Java Software by using Program Slicing Technique.
Tsuyoshi Motohashi, Computer Science Dept, Takushoku University.
Kyoko Iwasawa, Computer Science Dept, Takushoku University.

4. プログラムフローグラフ構築処理

メソッド情報に対して，バイトコードの命令文をメソッドごとに基本ブロック単位に分割する．そして，プログラムにおける制御依存関係に注目して基本ブロックをポインタで接続することによって，プログラムフローグラフを構築する．このときのプログラムフローグラフの例を図3に示す．

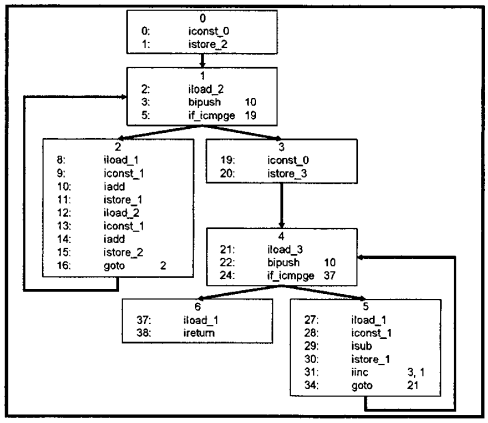

図3　プログラムフローグラフの例1

5. データフロー解析処理

プログラムフローグラフ構築処理で作成したプログラムフローグラフに対して，プログラムのデータの流れに注目して，変数の定義と参照の順序を解析する．そして，プログラムフローグラフにデータフローの情報を追加して，プログラムフローグラフを完成させる．基本ブロックの外部に及んでいるデータフローについては，図4のデータフロー方程式に従って解析する．このときのプログラムフローグラフの例を図5に示す．

```
in [ B ] = ∪ out [ P ]
out [ B ] = gen [ B ] ∪ ( in [ B ] − kill [ B ] )
＊B は現在の基本ブロック，P は先行ブロック
＊in [ B ] = φ，out [ B ] = gen [ B ]を初期値とする
```

図4　データフロー方程式

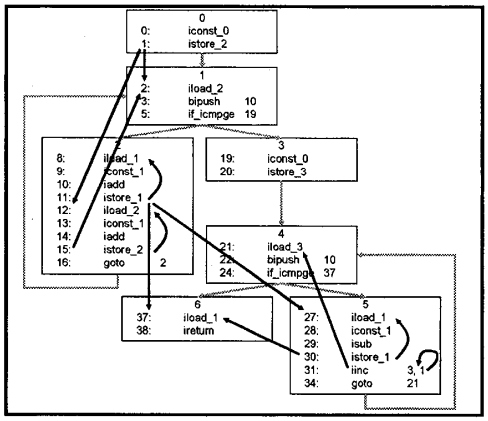

図5　プログラムフローグラフの例2

6. プログラムスライシング処理

Java ソースファイルとスライシング基準を取得する．まず，スライシング基準について，Java ソースファイルのソースコードとクラスファイルのバイトコードの対応を取る．そして，データフロー解析処理で拡張したプログラムフローグラフに対して，スライシング基準に従ってプログラムフローグラフを逆方向に探索することによって，プログラムにおける依存関係を解析する．この探索によって取得したプログラムの処理部分について，バイトコードとソースコードの対応を取って，該当する部分のソースコードをユーザに示す．この様子を図6に示す．

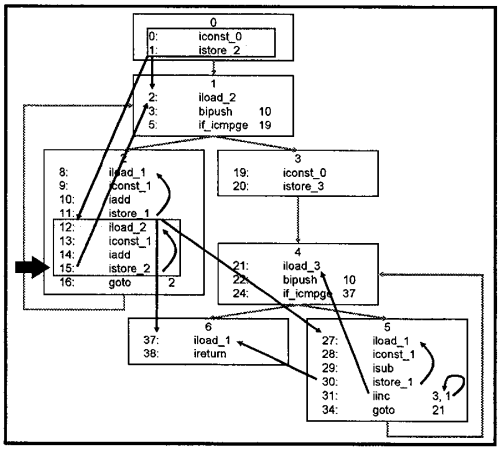

図6　プログラムスライシング処理の例

7. おわりに

ユーザが入力したプログラムから制御とデータの依存関係を解析して，特定の変数の値に影響を与えている可能性がある処理部分を出力するデバッグツールを作成した．デバッグツールは，使用する変数に影響を与えるすべての実行文を表示することができる。また，着目している変数の使用に直接届く定義文だけを表示することもできる．

デバッグツールは，プログラムの制御依存関係について，ソースプログラムをデータ依存関係とともに表示することによって，ユーザに示すことにしている．このように示すことによって，特定の変数の定義だけが強調表示されるので，ユーザの理解が容易になると判断した．

また，try-catch による例外処理などに対応できていないので，対象となるソースプログラムの記述が制限されている．対応できる記述の制約を減らすことができれば，デバッグツールに汎用性を持たせることができると考えている．

参考文献

[1]下村隆夫：プログラムスライシング技術と応用（共立出版，1995）
[2]A. V. エイホ，R. セシィ，J. D. ウルマン：コンパイラ―原理・技法・ツールII（サイエンス社，1990）
[3]菅田謙二，大畑文明，井上克郎：Java バイトコードにおけるデータ依存解析手法の提案と実現（日本ソフトウェア科学会，Vol. 18，No. 3，pp. 40-44，2001）