B-001

Java に対する Daikon を用いたインバリアント自動生成のための汎用基盤ツール

An Infrastructure Tool for Automatic Generation of Invariants in Java with Daikon

宮本 敬三 †　堀 直哉 †　岡野 浩三 †　楠本 真二 †
Keizo MIYAMOTO　Naoya HORI　Kozo OKANO　Shinji KUSUMOTO

1　まえがき

Java プログラムより表明を自動生成するツールの一つに Daikon [1] がある．だが，Daikon で表明を生成できるポイントはプログラム中のメソッドの入口と出口に限られている．しかし，プログラムに対する検証の自動化やプログラムの正確性，保守性，可読性を高めるという観点から条件分岐部やループ部に対して事前条件，事後条件，ループ不変式を生成したいという要求がある [2]．そこで，本研究では Daikon を用いてループ部に対して事前条件，事後条件，不変式をまとめて生成する手法を考案し，そのツールを実装した．この手法ではループ部を機能が等価なメソッドに変換する．また，ループ不変式を生成するためにループ内に機能を持たない空のメソッドを挿入し Daikon を適用する．そして，これらのメソッドに対する表明をループ部の事前条件，事後条件，ループ不変式を表す表明とする．また，対象の Java プログラムソースを XML 形式 (JavaML) [3] に変換した後，生成された表明を新たなノードとして追加する．プログラムソースと表明を共に XML 形式で管理することで生成された表明をより柔軟に利用できるようにした．

2　準備

JML, Daikon，ESC/Java2, JavaML について簡単に述べる．

JML(Java Modeling Language) [4] は Java プログラムソース中にコメント形式で表明を記述するための形式的記述言語である．Java プログラムの文法を利用しており，一般のプログラマなどが記述しやすくなっている．また，ESC/Java2 などのツールが多く利用できる．

Daikon [1] は生成した表明を JML 形式など様々な形式で出力できる．また，生成された表明を JML 形式で対象プログラムに挿入する等，利便性の高い機能も有している．一般に，複数のプログラム言語に対応しているが，本研究では Java プログラム言語を対象とする．

ESC/Java2 は JML による表明付きの Java プログラムに対し，プログラムが表明を満たしているか否かを静的に検証するツールである．

JavaML [3] は Java プログラムソースを XML 形式に変換したものである．XML 形式にすることで計算機上でプログラムソースへの操作が容易に行える．XML 形式からテキスト形式への変換は XSLT スタイルシートを用いて容易に行える．

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻

3　提案手法

3.1　概要

提案手法では入力プログラムソースとそのテストプログラムに対し，Daikon を用いてプログラム中のループ部に対してループ部の直前で成り立つ条件 (事前条件)，ループ部の直後で成り立つ条件 (事後条件)，ループ内部で不変的に成り立つ条件 (ループ不変式) を生成する．Daikon ではプログラムポイント以外の部分に対して表明を生成することはできないので元のプログラムソースからループ部の前後がプログラムポイントとなるように新たなメソッドを持つプログラムソースを作成する．そして，このプログラムソースに対して Daikon による表明生成を行う．本手法を実装したツールの処理は以下のとおりである (図 1)．

1. Jikes [5] を用いて解析対象の Java プログラムソースを JavaML [3] 形式に変換する．
2. ループ部をメソッドに変換する．
3. 2 の処理によって得られたファイルを再び Java プログラムソースに変換する．
4. 3 の処理によって得られたプログラムソースに対して Daikon を適用する．
5. JavaML 形式の対象プログラムの Java プログラムソースに 4 の処理で得られた表明を新たなノードとして追加する．

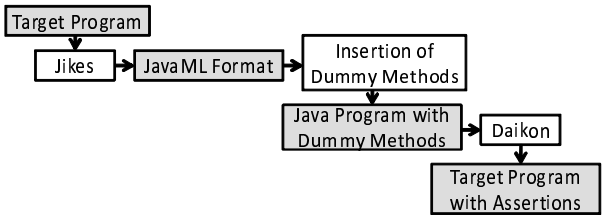

図 1　本ツールによる処理の概要

3.2　提案する拡張 JavaML 形式

3.2.1　拡張 JavaML 形式の構造

本手法によって生成された表明をプログラムソースと共に管理するための拡張 JavaML 形式について述べる．XML 形式の対象プログラムソース (JavaML) に対して生成された表明を新たなノードとして追加する．追加するノードはメソッドに関するものはメソッドノードの兄弟ノードとして，ループに関するものはループノードの兄弟ノードとして追加する．

3.2.2　拡張 JavaML 形式への変換

本ツールでは解析対象の Java プログラムのループ部に対してプログラムポイントを作成するための処理を

```
class sum extends Object {
  private int sum = 0;
  public  int summation(int i) {
    $d_summation_i  = i;
    $d_summation_sum = sum;
    $d_method0();
    sum = $d_summation_sum;
    i  = $d_summation_i;
    return (sum);
    }
  public static void main(String[] args) {
    int x = summation(10);
    System.out.println(x);
    }
  private  void $d_method0() {
    while  ($d_summation_i > 0) {
      $d_summation_sum  = $d_summation_sum + $d_summation_i;
      $d_summation_i--;
      dummy0();
      }
    }
  private  int $d_summation_i;
  private  int $d_summation_sum;
  private  void dummy0() {
    return ;
    }
  }
```

図 2　ループメソッドへの変換例

行う際に Java プログラムを JavaML 形式に変換する．JavaML 形式への変換は Jikes という Java コンパイラを用いた．JavaML 形式から Java プログラムソースへの変換は XSLT の一つである SAXON と，JavaML から Java プログラムソースの変換規則を定義したスタイルシートを用いた．

3.3　ループ部への Daikon の適用

3.3.1　ループ部のメソッド化

既述のように Daikon が事前条件，事後条件．不変式を生成できるのはメソッドの入口と出口に対してのみである．そのためループ部に対して表明を生成するためにループ部を抽出し機能等価なメソッド (ループメソッド) に変換する．ループ内部で変数の値を変更している場合，ループメソッド呼出し後に続く処理に値の変更を反映させる必要がある．そのためループ内部で使用される変数 (ループ内部で宣言される変数以外) を全てフィールド変数 (代理変数) として別名で宣言する．そしてループメソッドの呼び出し前後でこの代理変数と値の受け渡しを行うことで値の変更を反映させる．また，ループ不変式生成のために空のメソッド (ダミーメソッド) をループ内に挿入する．ループが入れ子になっている場合は各ループをループメソッドに変換する．図 2 にループメソッドへの変換例を示す．

3.3.2　ループラベルの挿入

ループ部に対して表明を生成し，拡張 JavaML のノードとして追加するためには，ループメソッドと元のプログラム中のループの対応関係を知る必要がある．そこでループ部に対してループメソッドの ID と同じ ID を持つラベル (ループラベル) を Java コード内のループ部

```
class sum extends Object {
   int sum = 0;
  public int summation(int i){
    $$_loopLabel_s  = 0;
    while  (i > 0) {
      sum  = sum + i;
      i--;
      }
    $$_loopLabel_e  = 0;
    return (sum);
    }
  private int $$_loopLabel_s;
  private int $$_loopLabel_e;
  }
```

図 3　ループラベルの挿入例

に埋め込む．表明生成後ループラベルの ID に対応するループメソッドの表明を出力ファイルから取得し新たにノードを作成し追加する．ループ部の開始と終了を示すラベルを，それぞれフィールド変数$$_loopLabel_s ,$$_loopLabel_e として定義する．そしてそれらの変数へのループの ID を表す数の代入式をループラベルとしてループ部の開始部と終了部に挿入する．ループが入れ子になっている場合はそれぞれのループに対してループラベルを付ける．ループラベル挿入例を図 3 に示す．

4　まとめ

Daikon を用いてメソッドだけでなく，ループ部に対しても事前条件，事後条件，ループ不変式を生成するための手法の提案と実装を行った．また，生成された表明をより汎用的に利用するために，拡張した JavaML 形式を提案した．

今後の課題として，ラベルつき break, continue への対応をしたい．また，研究グループで研究している静的解析と動的解析を組み合わせて表明を自動生成する手法にこの基盤ツールを利用し，種々の評価実験を行いたい．この際，拡張 JavaML ファイルを中間ファイルとして利用することを考えている．

参考文献

[1] M. D. Ernst, J. H. Perkins, P. J. Guo, S. Mccamant, C. Pacheco, M. S. Tschantz, and C. Xiao. The daikon system for dynamic detection of likely invariants. *Science of Computer Programming*, Vol. 69, No. 1-3, pp. 35–45, 2007.

[2] B. Meyer. Object-Oriented Software Construction 2nd Ed. *Prentice-Hall*, 2000.

[3] G. J. Badros. JavaML: a markup language for Java source code. *in Computer(IEEE)*, Vol. 25, No. 10, pp. 40–51, 1999.

[4] G. T. Leavens, A. L. Baker, and C. Ruby. Preliminary design of JML: a behavioral interface specification language for java. *ACM SIGSOFT*, Vol. 31, No. 3, pp. 1–35, 2006.

[5] Jikes's home. *http://jikes.sourceforge.net/*.