Techniques for Testing Scientific Programs Without an Oracle

Upulee Kanewala and James M. Bieman

Computer Science Department
Colorado State University

May 18, 2013

This work is supported by the National Institute of General Medical Sciences of the N.I.H
Outline

1. Introduction
2. Metamorphic testing (MT)
3. Assertion checking
4. Oracles generated with Machine Learning (ML)
5. Discussion
Introduction

Non-testable programs
- An oracle does not exist or too hard to implement
- Many scientific programs are non-testable

Ad-hoc approaches used for implementing oracles
Test oracle precision

- Oracle precision: tolerance for errors in the oracle
  - **Null oracle**: detects whether program crashes or hangs
  - **Ideal oracle**: provides correct pass/fail judgment for any possible execution
  - **Pseudo-oracle**: independent implementation of the program under test

- Three methods fall between the null oracle and the ideal oracle
Oracle Properties

Completeness

If
the result of a test $t$ is correct according to specification $s$
Then
oracle $o$ will indicate that $t$ has passed

Soundness

If
oracle $o$ indicates that a test $t$ has passed
Then
program $p$ is correct w.r.t $t$ and specification $s$
Oracle Properties

Completeness

If
result of a test $t$ is correct according to specification $s$
Then
oracle $o$ will indicate that $t$ has passed

Soundness

If
oracle $o$ indicates that a test $t$ has passed
Then
program $p$ is correct w.r.t $t$ and specification $s$
Outline

1. Introduction
2. Metamorphic testing (MT)
3. Assertion checking
4. Oracles generated with Machine Learning (ML)
5. Discussion
Metamorphic testing (MT)

- MT checks whether the program behaves according to a set of *metamorphic relations* (MRs)
- MR specifies how the output changes according to a change made to the input
MT Process

- Identify MRs that need to be satisfied by the PUT
- Create a set of initial test cases
- Create follow-up test cases based on input transformations
- Execute test case pairs and check whether the MR holds

**Initial test case:**
\[a_1 \ a_2 \ a_3 \ a_4 \ a_5 = S\]

**Follow-up test case 1:**
\[a_4 \ a_3 \ a_1 \ a_5 \ a_2 = S\]

**Follow-up test case 2:**
\[a_1+k \ a_2+k \ a_3+k \ a_4+k \ a_5+k = S+5^k\]
MT Process

- Identify MRs that need to be satisfied by the PUT
- Create a set of initial test cases
- Create follow-up test cases based on input transformations
- Execute test case pairs and check whether the MR holds

Initial test case: $a_1 \ a_2 \ a_3 \ a_4 \ a_5 = S$

Permute

Follow-up test case 1: $a_4 \ a_3 \ a_1 \ a_5 \ a_2 = S$

Add a constant

Follow-up test case 2: $(a_1+k) \ (a_2+k) \ (a_3+k) \ (a_4+k) \ (a_5+k) = S+5k$
MT Process

- Identify MRs that need to be satisfied by the PUT
- Create a set of initial test cases
- Create follow-up test cases based on input transformations
- Execute test case pairs and check whether the MR holds

Initial test case: $a_1 \ a_2 \ a_3 \ a_4 \ a_5 = S$

Follow-up test case 1: $a_4 \ a_3 \ a_1 \ a_5 \ a_2 = S$

Follow-up test case 2: $(a_1+k) \ (a_2+k) \ (a_3+k) \ (a_4+k) \ (a_5+k) = S+5k$
MT Process

- Identify MRs that need to be satisfied by the PUT
- Create a set of initial test cases
- Create follow-up test cases based on input transformations
- Execute test case pairs and check whether the MR holds
Applications of MT

- Applications:
  - Testing ML classifiers: k-nearest neighbor (k-NN) and Naïve Bayes classifier (NBC) \(^1\)
  - Unit and integration testing of a jpeg2000 encoder\(^2\)

- MT oracles are complete and not sound

---


Assertion Checking

```cpp
long square_root(long x) 
in {
    assert(x >= 0);} 
out (result) {
    assert((result * result) <= x && (result+1) * (result+1) >= x);} 
body {
    return cast(long)std.math.sqrt(cast(real)x); }
```

- Assertions: expected range of output, variable relationships within the program
- Assertions can be more effective than the null oracle
- Oracle is complete but not sound

---

Oracles generated with Machine Learning (ML)

**Training phase**

$$i_j \rightarrow P \rightarrow o_j$$

$$1 \leq j \leq m$$
Oracles generated with Machine Learning (ML)

Training phase

1 ≤ j ≤ m

<table>
<thead>
<tr>
<th>i_j</th>
<th>P</th>
<th>o_j</th>
</tr>
</thead>
</table>

Training set

<table>
<thead>
<tr>
<th></th>
<th>f_1</th>
<th>f_2</th>
<th>...</th>
<th>f_k</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_1</td>
<td>v_{11}</td>
<td>v_{12}</td>
<td>...</td>
<td>v_{1k}</td>
<td>p</td>
</tr>
<tr>
<td>i_2</td>
<td>v_{21}</td>
<td>v_{22}</td>
<td>...</td>
<td>v_{2k}</td>
<td>f</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>i_m</td>
<td>v_{m1}</td>
<td>v_{m2}</td>
<td>...</td>
<td>v_{mk}</td>
<td>p</td>
</tr>
</tbody>
</table>
Oracles generated with Machine Learning (ML)

Training phase

1 \leq j \leq m

Training set

Decision Tree
Oracles generated with Machine Learning (ML)

Training phase

1 ≤ j ≤ m

Testing phase
Oracles generated with Machine Learning (ML)

Introduction

Metamorphic testing (MT)

Assertion checking

Techniques for Testing Scientific Programs Without an Oracle

Upulee Kanewala and James M. Bieman

Colorado State University
Oracles generated with Machine Learning (ML)

Training phase

1 ≤ j ≤ m

Testing phase

Upulee Kanewala and James M. Bieman
Colorado State University

Techniques for Testing Scientific Programs Without an Oracle
Applications of oracles generated with ML

- **Applications:**
  - Mesh simplification program \(^4\)
    - Used another similar program to create training data
  - Image segmentation program \(^5\)
    - Used previous versions of the same program to create training data

- Oracles are not complete and not sound

---


Fault finding effectiveness

- Two studies compared MT vs assertion checking \(^6,7\)
  - Determined MT is more effective but used different test sets
- Oracles developed using ML were not compared
- Adding more test cases alone will not improve the effectiveness

\(^6\) Z. Zhang, W. K. Chan, T. H. Tse, and P. Hu, “Experimental study to compare the use of metamorphic testing and assertion checking,” *Journal of Software*, 2009

Domain knowledge

- Domain expert is required to develop MRs/assertions
- ML oracles:
  - ML knowledge
  - Selecting suitable features
Automation

- **MT**: techniques for automatically identifying likely MRs
- **Assertion checking**: automatic invariant detection tools
- **Disadvantage**: Can make these oracles incomplete
Unsolved Problems

- Metamorphic testing:
  - Develop techniques to automatically detect likely MRs
  - Develop techniques to identify effective MRs

- Assertion checking:
  - Minimize spurious invariants detected by automatic invariant detection methods

- Oracles generated with ML:
  - Investigate applicability of ML based oracles in other domains
Unsolved Problems

- **Metamorphic testing:**
  - Develop techniques to automatically detect likely MRs
  - Develop techniques to identify effective MRs

- **Assertion checking:**
  - Minimize spurious invariants detected by automatic invariant detection methods

- **Oracles generated with ML:**
  - Investigate applicability of ML based oracles in other domains
Unsolved Problems

- **Metamorphic testing:**
  - Develop techniques to automatically detect likely MRs
  - Develop techniques to identify effective MRs

- **Assertion checking:**
  - Minimize spurious invariants detected by automatic invariant detection methods

- **Oracles generated with ML:**
  - Investigate applicability of ML based oracles in other domains
Summary

- Many scientific programs are non-testable

- Three techniques for testing programs that do not have oracles

<table>
<thead>
<tr>
<th>Technique</th>
<th>MT</th>
<th>Assertion Checking</th>
<th>ML based Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>ML Classifiers</td>
<td>JDK 1.4.2 classes</td>
<td>Mesh simplification</td>
</tr>
<tr>
<td></td>
<td>jpeg encoder</td>
<td></td>
<td>Image segmentation</td>
</tr>
<tr>
<td>Complete</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Sound</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- Fault finding ability, automation and domain knowledge differs
Thank you!