Quality Assurance of Bioinformatics Software

Morteza Pourreza Shahri, Madhusudan Srinivasan, Upulee Kanewala, Indika Kahanda
Gianforte School of Computing, Montana State University

BIOINFORMATICS SOFTWARE

- Plays a very important role in making critical decisions within many areas including medicine and health care.
- Most of the research is directed toward developing tools.
- Little time and effort is spent on testing the software.
- The main challenge associated with testing bioinformatics software is the oracle problem.
  - Where a test oracle is not available or practically difficult to implement.

METAMORPHIC TESTING

- A technique used to test programs that face the oracle problem.
- Operates by checking whether the program under test behaves according to an expected set of properties known as metamorphic relations.
- Specifies how a particular change to the input of the program should change the output.
- MR is violated when the change in output differs from the definition.

METHOD

- We use MT to test LingPipe 4.1.2.
  - A tool for processing text using computational linguistics.
  - Often used in bioinformatics for bio-entity recognition.
- We choose two classes of LingPipe because they:
  - Do not have unit tests.
  - Provide important functionality for bio-entity recognition—identified via analyzing execution traces.
- PIT for generating faulty versions of the program (i.e. mutants).

```java
for (int j = 0; j < tokenCount; j++) {
    System.out.println("Split Output: " + tokens[j]);
} // Original
```

```java
for (int j = 0; j < tokenCount; j++) {
    System.out.println("Split Output: " + tokens[j]);
} // Mutant
```

- Identify the set of MRs for LingPipe bio-entity recognition.
- Identify classes to test based on the code coverage.
- Check the MRs on the original program.
- If MR is violated, fix the class.
- Generate the mutants for each class using PIT.
- Execute the initial and follow-up test cases on the mutants.
- Validate the results based on MRs.

RESULTS

<table>
<thead>
<tr>
<th>Class</th>
<th>Killing Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR1</td>
<td>36.4%</td>
</tr>
<tr>
<td>MR2</td>
<td>36.4%</td>
</tr>
<tr>
<td>MR3</td>
<td>25.0%</td>
</tr>
<tr>
<td>MR4</td>
<td>27.3%</td>
</tr>
<tr>
<td>MR5</td>
<td>25.0%</td>
</tr>
<tr>
<td>MR6</td>
<td>27.3%</td>
</tr>
<tr>
<td>MR7</td>
<td>7.1%</td>
</tr>
<tr>
<td>MR8</td>
<td>33.3%</td>
</tr>
<tr>
<td>MR9</td>
<td>33.3%</td>
</tr>
<tr>
<td>MR10</td>
<td>41.7%</td>
</tr>
</tbody>
</table>

CONCLUSIONS AND FUTURE WORK

- The results show that MT based automated testing can detect faulty versions of the program, which shows the utility of this testing technique for quality assurance.
- MRs based on shuffling the paragraphs of an article and removing a paragraph from an article provide the best killing rates for these classes.
- We will investigate the effectiveness of MT based unit testing by developing MRs for individual functions in the investigated classes.

REFERENCES