Using Program Spectra to Improve the Effectiveness of Regression Testing

Christopher Hayden

cmhayden@cs.umd.edu

Department of Computer Science
University of Maryland, College Park

Student Presentation
CMSC737, Fundamentals of Software Testing
November 24, 2009
Introduction
- Regression Testing
- The Basic Problem

Program Spectra
- Types of Program Spectra
- Relationships Among Program Spectra Types

Some Empirical Studies
- Program Spectra Safety and Imprecision [1]
- Value Spectra and Deviation-Root Localization [2]
1 Introduction
   - Regression Testing
   - The Basic Problem

2 Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3 Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
What is Regression Testing?

**Definition**

*Regression Testing* is the retesting of a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

The process:

1. Generate a test suite.
2. Modify the source code.
3. Execute the test cases using modified version.
4. Search for deviations in the output.
**Definition**

Regression Testing is the retesting of a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

The process:

1. Generate a test suite.
2. Modify the source code.
3. Execute the test cases using modified version.
4. Search for deviations in the output.
Definition

**Regression Testing** is the retesting of a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

The process:

1. Generate a test suite.
2. Modify the source code.
3. Execute the test cases using modified version.
4. Search for deviations in the output.
What is Regression Testing?

**Definition**

*Regression Testing* is the retesting of a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

The process:

1. Generate a test suite.
2. Modify the source code.
3. Execute the test cases using modified version.
4. Search for deviations in the output.
What is Regression Testing?

**Definition**

Regression Testing is the retesting of a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

The process:

1. Generate a test suite.
2. Modify the source code.
3. Execute the test cases using modified version.
4. Search for deviations in the output.
Outline

1. Introduction
   - Regression Testing
   - The Basic Problem

2. Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3. Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
Regression Testing is not Perfect

Regression Testing has a few shortcomings . . .

- Expensive to rerun the entire test suite.
- Assumes deviations propagate to output.
- Difficult to find the cause of a deviation.
Regression Testing is not Perfect

Regression Testing has a few shortcomings . . .

- Expensive to rerun the entire test suite.
- Assumes deviations propagate to output.
- Difficult to find the cause of a deviation.
Regression Testing is not Perfect

Regression Testing has a few shortcomings . . .

- Expensive to rerun the entire test suite.
- Assumes deviations propagate to output.
- Difficult to find the cause of a deviation.
Regression Testing is not Perfect

Regression Testing has a few shortcomings . . .

- Expensive to rerun the entire test suite.
- Assumes deviations propagate to output.
- Difficult to find the cause of a deviation.
There is Another Way . . .

What if we selectively sample the program state during test execution?

The extra information recovered could help address our problems.

This is exactly the idea behind program spectra.
There is Another Way . . .

What if we selectively sample the program state during test execution?

The extra information recovered could help address our problems.

This is exactly the idea behind program spectra.
There is Another Way . . .

What if we selectively sample the program state during test execution?

The extra information recovered could help address our problems.

This is exactly the idea behind program spectra.
Outline

1. Introduction
   - Regression Testing
   - The Basic Problem

2. Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3. Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
What are Program Spectra?

Definition

A program spectrum is a distribution, or signature, of some aspect of a program’s run-time behavior.

Example

The distribution of paths traversed by the executions of a program is called a path spectrum.
Types of Program Spectra

Branch/Path Spectra
  Record loop-free intraprocedural paths executed.

Data-Dependence Spectra
  Record the set of definition-use pairs exercised.

Output Spectra
  Record the output produced.

Execution Trace Spectra
  Record the sequence of statements executed.
Classifications of Program Spectra

Definition
- **Hit spectra** track whether or not a given instance occurs.
- **Count spectra** record the number of times an instance occurs.
- **Trace spectra** track the order in which instances occur.

Definition
- **Syntactic spectra** are defined by the structural elements.
- **Semantic spectra** are spectra defined by program states.
**Definition**

**Hit spectra** track whether or not a given instance occurs.
**Count spectra** record the number of times an instance occurs.
**Trace spectra** track the order in which instances occur.

**Definition**

**Syntactic spectra** are defined by the structural elements.
**Semantic spectra** are spectra defined by program states.
Outline

1. Introduction
   - Regression Testing
   - The Basic Problem

2. Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3. Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
Program Spectra Subsumption

**Definition**

Program spectra type $S_1$ **subsumes** program spectra type $S_2$ iff whenever the $S_2$ spectra for program $P$, version $P'$, and input $i$ differ, the $S_1$ spectra for program $P$, version $P'$, and input $i$ differ.

There is a correlation between subsumption and overhead.

Christopher Hayden  cmhayden@cs.umd.edu  Program Spectra and Regression Testing
Definition

Program spectra type $S_1$ subsumes program spectra type $S_2$ iff whenever the $S_2$ spectra for program $P$, version $P'$, and input $i$ differ, the $S_1$ spectra for program $P$, version $P'$, and input $i$ differ.

There is a correlation between subsumption and overhead.
Subsumption Hierarchy

- Execution Trace Spectra
  - Data Count Spectra
    - Data Hit Spectra
  - Complete Path Spectra
    - Path Count Spectra
    - Path Hit Spectra
  - Output Spectra
    - Branch Count Spectra
    - Branch Hit Spectra
Outline

1 Introduction
   - Regression Testing
   - The Basic Problem

2 Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3 Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
Given program $P$, faulty version $P'$, and input space $U$:

**Definition**
The **fault revealing** set of inputs $FR(P, P', U)$ is the set of inputs in $U$ that cause $P'$ to fail.

**Definition**
The **spectrum $S$ revealing** set of inputs $SR(P, P', U)$ is the set of inputs in $U$ that cause the spectra for $P$ and $P'$ of type $S$ to differ.
Given program $P$, faulty version $P'$, and input space $U$:

**Definition**

The **fault revealing** set of inputs $FR(P, P', U)$ is the set of inputs in $U$ that cause $P'$ to fail.

**Definition**

The **spectrum $S$ revealing** set of inputs $SR(P, P', U)$ is the set of inputs in $U$ that cause the spectra for $P$ and $P'$ of type $S$ to differ.
The Experiment

Objectives

- How often are inputs that cause a failure revealed in spectra differences?

- How often do inputs that cause spectra differences lead to program failure?
The Experiment
Objectives

- How often are inputs that cause a failure revealed in spectra differences?

- How often do inputs that cause spectra differences lead to program failure?
The degree of imprecision of spectrum type $S$ is given by

$$\frac{|SR(P, P', U) - FR(P, P', U)|}{|SR(P, P', U)|}.$$ 

The degree of unsafety of spectrum type $S$ is given by

$$\frac{|FR(P, P', U) - SR(P, P', U)|}{|FR(P, P', U)|}.$$
The degree of imprecision of spectrum type $S$ is given by
\[
\frac{|SR(P, P', U) - FR(P, P', U)|}{|SR(P, P', U)|}.
\]

The degree of unsafety of spectrum type $S$ is given by
\[
\frac{|FR(P, P', U) - SR(P, P', U)|}{|FR(P, P', U)|}.
\]
The Experiment

Design

- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program’s inputs.
- Every fault revealing input is also output spectra revealing.
- Threats to validity:
  - Results don’t generalize.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program’s inputs.
- Every fault revealing input is also output spectra revealing.
- Threats to validity:
  - Results don’t generalize.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program’s inputs.
- Every fault revealing input is also output spectra revealing.
- Threats to validity:
  - Results don’t generalize.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program’s inputs.
- Every fault revealing input is also output spectra revealing.
- Threats to validity:
  - Results don’t generalize.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program’s inputs.
- Every fault revealing input is also output spectra revealing.

Threats to validity:
- Results don’t generalize.
- Instrumentation effects resulting in bias.
# The Experiment

## Subjects

<table>
<thead>
<tr>
<th>Program</th>
<th>Lines of Code</th>
<th>Number of Versions</th>
<th>Input Space Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totinfo</td>
<td>431</td>
<td>22</td>
<td>1052</td>
<td>information measure</td>
</tr>
<tr>
<td>schedule1</td>
<td>416</td>
<td>7</td>
<td>2650</td>
<td>priority scheduler</td>
</tr>
<tr>
<td>schedule2</td>
<td>309</td>
<td>2</td>
<td>2710</td>
<td>priority scheduler</td>
</tr>
<tr>
<td>teas</td>
<td>238</td>
<td>26</td>
<td>1608</td>
<td>altitude separation</td>
</tr>
<tr>
<td>printtok1</td>
<td>584</td>
<td>4</td>
<td>4130</td>
<td>lexical analyzer</td>
</tr>
<tr>
<td>printtok2</td>
<td>513</td>
<td>7</td>
<td>4115</td>
<td>lexical analyzer</td>
</tr>
<tr>
<td>replace</td>
<td>569</td>
<td>20</td>
<td>5542</td>
<td>pattern replacement</td>
</tr>
</tbody>
</table>
The Results

![Graphs showing degree of unsafety and degree of imprecision for different spectra types.](image-url)
Outline

1. Introduction
   - Regression Testing
   - The Basic Problem

2. Program Spectra
   - Types of Program Spectra
   - Relationships Among Program Spectra Types

3. Some Empirical Studies
   - Program Spectra Safety and Imprecision [1]
   - Value Spectra and Deviation-Root Localization [2]
Value Spectra

Definition

A function-entry state $S^{entry}$ comprises argument values and global variable values at function entry.

A function-exit state $S^{exit}$ comprises argument values, global variable values, and return value at function exit.

A function execution $\langle S^{entry} , S^{exit} \rangle$ is a function-entry state–function-exit state pair.

Definition

A value spectrum is a distribution of function executions.
Value Spectra

**Definition**

A function-entry state $S^{entry}$ comprises argument values and global variable values at function entry.

A function-exit state $S^{exit}$ comprises argument values, global variable values, and return value at function exit.

A function execution $\langle S^{entry}, S^{exit} \rangle$ is a function-entry state–function-exit state pair.

**Definition**

A value spectrum is a distribution of function executions.
Path Spectra

Definition

A function-entry state $S_{\text{entry}}$ is the path taken from the beginning of the program to the point of function entry.
A function-exit state $S_{\text{exit}}$ is the path taken from the beginning of the program to the point of function exit.
A function execution $\langle S_{\text{entry}}, S_{\text{exit}} \rangle$ is a function-entry state–function-exit state pair.

Definition

A path spectrum is a distribution of function executions.
Path Spectra

Definition

A function-entry state $S_{entry}$ is the path taken from the beginning of the program to the point of function entry. A function-exit state $S_{exit}$ is the path taken from the beginning of the program to the point of function exit. A function execution $\langle S_{entry}, S_{exit} \rangle$ is a function-entry state–function-exit state pair.

Definition

A path spectrum is a distribution of function executions.
Deviations Propagation and Deviation Roots

**Definition**

Given \( f_{\text{new}} : \langle S_{\text{entry}}^{\text{new}}, S_{\text{exit}}^{\text{new}} \rangle \) and \( f_{\text{old}} : \langle S_{\text{entry}}^{\text{old}}, S_{\text{exit}}^{\text{old}} \rangle \), if \( f_{\text{new}} \neq f_{\text{old}} \) then \( f_{\text{new}} \) is a **deviated function execution**. If \( S_{\text{entry}}^{\text{new}} = S_{\text{entry}}^{\text{old}} \) but \( S_{\text{exit}}^{\text{new}} \neq S_{\text{exit}}^{\text{old}} \) then \( f_{\text{new}} \) is a **deviation container**. If \( S_{\text{entry}}^{\text{new}} \neq S_{\text{entry}}^{\text{old}} \) then \( f_{\text{new}} \) is a **deviation follower**.

**Definition**

A deviation root is a change that originates a deviation.
Deviation Propagation and Deviation Roots

Definition

Given $f_{new} : (S_{entry\ new}, S_{exit\ new})$ and $f_{old} : (S_{entry\ old}, S_{exit\ old})$, if $f_{new} \neq f_{old}$ then $f_{new}$ is a deviated function execution. If $S_{entry\ new} = S_{entry\ old}$ but $S_{exit\ new} \neq S_{exit\ old}$ then $f_{new}$ is a deviation container. If $S_{entry\ new} \neq S_{entry\ old}$ then $f_{new}$ is a deviation follower.

Definition

A deviation root is a change that originates a deviation.
Deviation Root Localization

Heuristic 1

Given functions \( f \) and \( g \) such that:

- \( f \) is a deviation follower.
- \( g \) is the caller of \( f \) and a deviation container or non-deviated.
- All function executions between \( g \)'s entry and \( f \)'s call site are non-deviated.

Then deviation roots are likely among statements between \( g \)'s entry and \( f \)'s call site.
Deviation Root Localization

Heuristic 2

Given a function $f$ such that:

- $f$ is a deviation container.
- The callees of $f$ are non-deviated.

Then deviation roots are likely among the statements of $f$’s body.
The Experiment

Objectives

- How different are value spectra, path spectra, and output spectra in their ability to expose deviations?

- How accurately do the deviation-root localization heuristics work for value and path spectra?
The Experiment
Objectives

- How different are value spectra, path spectra, and output spectra in their ability to expose deviations?

- How accurately do the deviation-root localization heuristics work for value and path spectra?
The Experiment

Measures

Definition

The deviation exposure ratio is given by

\[
\frac{\text{deviated tests}}{\text{covering tests}} = \frac{|DT(S,P,P',CT)|}{|CT|}.
\]

Definition

The deviation-root localization ratio is given by

\[
\frac{\text{localized tests}}{\text{deviated tests}} = \frac{|LT(S,P,P',CT)|}{|DT(S,P,P',CT)|}.
\]
The Experiment

Measures

Definition

The **deviation exposure ratio** is given by

\[
\frac{\text{deviated tests}}{\text{covering tests}} = \frac{|DT(S,P,P',CT)|}{|CT|}.
\]

Definition

The **deviation-root localization ratio** is given by

\[
\frac{\text{localized tests}}{\text{deviated tests}} = \frac{|LT(S,P,P',CT)|}{|DT(S,P,P',CT)|}.
\]
The Experiment
Design

- Instrument 8 C programs, each with faulty versions and a test suite.
- Execute the test suite on the correct version.
- For each faulty version, execute those test cases that cover the faulty code.
- Threats to validity:
  - Subject programs are not representative of population.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 8 C programs, each with faulty versions and a test suite.
- **Execute the test suite on the correct version.**
- For each faulty version, execute those test cases that cover the faulty code.
- Threats to validity:
  - Subject programs are not representative of population.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 8 C programs, each with faulty versions and a test suite.
- Execute the test suite on the correct version.
- For each faulty version, execute those test cases that cover the faulty code.
- Threats to validity:
  - Subject programs are not representative of population.
  - Instrumentation effects resulting in bias.
The Experiment

Design

- Instrument 8 C programs, each with faulty versions and a test suite.
- Execute the test suite on the correct version.
- For each faulty version, execute those test cases that cover the faulty code.

Threats to validity:
- Subject programs are not representative of population.
- Instrumentation effects resulting in bias.
### The Experiment

#### Subjects

<table>
<thead>
<tr>
<th>program</th>
<th>funcs</th>
<th>loc</th>
<th>tests</th>
<th>vers</th>
<th>vsgen (sec/test)</th>
<th>vscomp (sec/test)</th>
<th>psgen (sec/test)</th>
<th>pscomp (sec/test)</th>
<th>vsize (kb/test)</th>
<th>psize (kb/test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>printtok</td>
<td>18</td>
<td>402</td>
<td>4130</td>
<td>7</td>
<td>0.76</td>
<td>0.18</td>
<td>0.14</td>
<td>0.04</td>
<td>6.51</td>
<td>0.92</td>
</tr>
<tr>
<td>printtok2</td>
<td>19</td>
<td>483</td>
<td>4115</td>
<td>10</td>
<td>0.48</td>
<td>0.08</td>
<td>0.19</td>
<td>0.05</td>
<td>1.72</td>
<td>1.19</td>
</tr>
<tr>
<td>replace</td>
<td>21</td>
<td>516</td>
<td>5542</td>
<td>32</td>
<td>0.49</td>
<td>0.08</td>
<td>0.18</td>
<td>0.02</td>
<td>2.1</td>
<td>0.85</td>
</tr>
<tr>
<td>schedule</td>
<td>18</td>
<td>299</td>
<td>2650</td>
<td>9</td>
<td>1.22</td>
<td>0.15</td>
<td>0.18</td>
<td>0.04</td>
<td>6.72</td>
<td>1.27</td>
</tr>
<tr>
<td>schedule2</td>
<td>16</td>
<td>297</td>
<td>2710</td>
<td>10</td>
<td>1.24</td>
<td>0.19</td>
<td>0.30</td>
<td>0.06</td>
<td>6.09</td>
<td>1.42</td>
</tr>
<tr>
<td>teas</td>
<td>9</td>
<td>138</td>
<td>1608</td>
<td>41</td>
<td>0.35</td>
<td>0.04</td>
<td>0.03</td>
<td>0.02</td>
<td>0.36</td>
<td>0.23</td>
</tr>
<tr>
<td>totinfo</td>
<td>7</td>
<td>346</td>
<td>1052</td>
<td>23</td>
<td>0.51</td>
<td>0.04</td>
<td>0.13</td>
<td>0.02</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>space</td>
<td>135</td>
<td>6218</td>
<td>13585</td>
<td>18</td>
<td>1.46</td>
<td>0.23</td>
<td>0.28</td>
<td>0.07</td>
<td>28.43</td>
<td>4.03</td>
</tr>
</tbody>
</table>
The Results
Deviation Exposure Ratios
The Results
Deviation-Root Localization
The Results

Cost of Analysis

- Time cost of generation higher than time cost of comparison.
- Time and space costs increase as program size increases.
- Time and space costs of higher for value spectra than for path spectra:

\[
V\text{Cost} = O(|vars| \times |userfuncs| \times |testsuite|)
\]
\[
P\text{Cost} = O(|branches| \times |userfuncs| \times |testsuite|)
\]
The Results
Cost of Analysis

- Time cost of generation higher than time cost of comparison.
- Time and space costs increase as program size increases.
- Time and space costs of higher for value spectra than for path spectra:

\[
VCost = O(|vars| \times |userfuncs| \times |testsuite|)
\]
\[
PCost = O(|branches| \times |userfuncs| \times |testsuite|)
\]
The Results
Cost of Analysis

- Time cost of generation higher than time cost of comparison.
- Time and space costs increase as program size increases.
- Time and space costs of higher for value spectra than for path spectra:

\[
\begin{align*}
VCost & = O(|vars| \times |userfuncs| \times |testsuite|) \\
PCost & = O(|branches| \times |userfuncs| \times |testsuite|)
\end{align*}
\]
Summary

Program spectra can improve regression testing by:
- Reducing the number of tests that need to be run (maybe).
- Revealing faults that do not propagate to output.
- Localizing the portion of code causing deviations.

Future work:
- Verify first conjecture above.
- Reduce the degree of imprecision.
- Create and examine new types of program spectra.
Program spectra can improve regression testing by:

- Reducing the number of tests that need to be run (maybe).
- Revealing faults that do not propagate to output.
- Localizing the portion of code causing deviations.

Future work:

- Verify first conjecture above.
- Reduce the degree of imprecision.
- Create and examine new types of program spectra.