# 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

# 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]

# 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 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.

# 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.

# 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.

# 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.

# 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 $S1$ subsumes program spectra type $S2$ iff whenever the $S2$ spectra for program $P$, version $P'$, and input $i$ differ, the $S1$ spectra for program $P$, version $P'$, and input $i$ differ.

There is a correlation between subsumption and overhead.

# Program Spectra Subsumption

**Definition**

Program spectra type $S1$ subsumes program spectra type $S2$ iff whenever the $S2$ spectra for program $P$, version $P'$, and input $i$ differ, the $S1$ spectra for program $P$, version $P'$, and input $i$ differ.

There is a correlation between subsumption and overhead.

# Subsumption Hierarchy

$$
\begin{tikzcd}
 & & \text{Execution Trace Spectra} \arrow[dll] \arrow[d] \arrow[drr] & & \\
\text{Data Count Spectra} \arrow[d] & & \text{Complete Path Spectra} \arrow[d] & & \text{Output Spectra} \\
\text{Data Hit Spectra} & & \text{Path Count Spectra} \arrow[dl] \arrow[dr] & & \\
 & \text{Branch Count Spectra} \arrow[dr] & & \text{Path Hit Spectra} \arrow[dl] & \\
 & & \text{Branch Hit Spectra} & &
\end{tikzcd}
$$

# 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]

# Defintions

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.

# Defintions

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 causes a failure revealed in spectra differences?
- How often do inputs that causes spectra differences lead to program failure?

# The Experiment

## Objectives

- How often are inputs that causes a failure revealed in spectra differences?
- How often do inputs that causes spectra differences lead to program failure?

# The Experiment
## Measures

### Definition

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

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

### Definition

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
## Measures

### Definition

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

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

### Definition

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

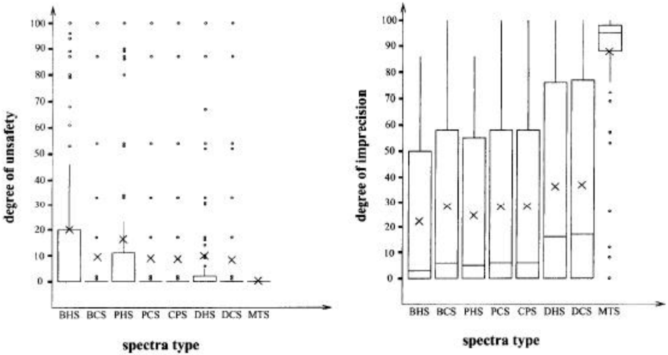
- Instrument 7 C programs, each with faulty versions.
- A single independent variable: the spectra type.
- Apply each spectra calculation to each version of each program for each of that program's inputs.
- Every fault revealing input is also output spectra revealing.
- Threats to validity:
  - Results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

# The Experiment

## Subjects

| Program | Lines of Code | Number of Versions | Input Space Size | Description |
|---|---|---|---|---|
| totinfo | 431 | 22 | 1052 | information measure |
| schedule1 | 416 | 7 | 2650 | priority scheduler |
| schedule2 | 309 | 2 | 2710 | priority scheduler |
| teas | 238 | 26 | 1608 | altitude separation |
| printtok1 | 584 | 4 | 4130 | lexical analyzer |
| printtok2 | 513 | 7 | 4115 | lexical analyzer |
| replace | 569 | 20 | 5542 | pattern replacement |

# The Results


degree of unsafety
0
10
20
30
40
50
60
70
80
90
100
BHS BCS PHS PCS CPS DHS DCS MTS
spectra type
degree of imprecision
0
10
20
30
40
50
60
70
80
90
100
BHS BCS PHS PCS CPS DHS DCS MTS
spectra type

# 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 $S^{entry}$ 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.

# 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.

# Deviation Propagation and Deviation Roots

## Definition

Given $f_{new} : \langle S^{entry}_{new}, S^{exit}_{new} \rangle$ and $f_{old} : \langle S^{entry}_{old}, S^{exit}_{old} \rangle$, 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 Propagation and Deviation Roots

## Definition

Given $f_{new} : \langle S^{entry}_{new}, S^{exit}_{new} \rangle$ and $f_{old} : \langle S^{entry}_{old}, S^{exit}_{old} \rangle$, 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.

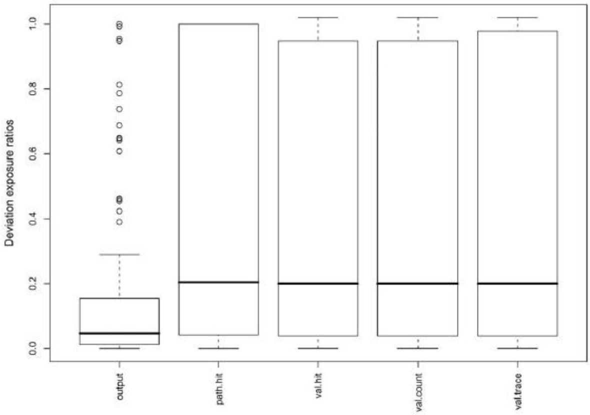
# The Experiment

## Subjects

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

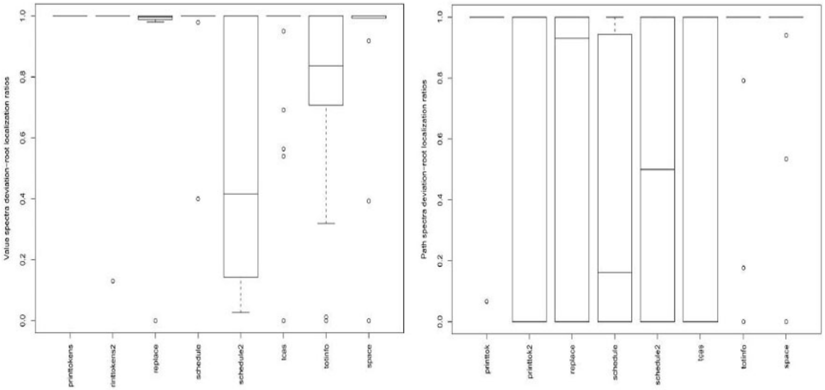
# The Results

## 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:

$$\begin{aligned} VCost &= O(|vars| \times |userfuncs| \times |testsuite|) \\ PCost &= O(|branches| \times |userfuncs| \times |testsuite|) \end{aligned}$$

# 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:

$$
\begin{aligned}
VCost &= O(|vars| \times |userfuncs| \times |testsuite|) \\
PCost &= O(|branches| \times |userfuncs| \times |testsuite|)
\end{aligned}
$$

# 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:

$$\begin{aligned} VCost &= O(|vars| \times |userfuncs| \times |testsuite|) \\ PCost &= O(|branches| \times |userfuncs| \times |testsuite|) \end{aligned}$$

# 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.

# 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.

- M.J. Harrold, G. Rothermel, K. Sayre, R. Wu, and L. Yi, "An Empirical Investigation of the Relationship between Spectra Differences and Regression Faults," *J.Software Testing, Verification and Reliability*, vol. 10, no. 3, pp. 171-194, 2000.
- Tao Xie and David Notkin, "Checking Inside the Black Box: Regression Testing by Comparing Value Spectra," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 869-883, 2005.