An Empirical Investigation of Program Spectra*

Mary Jean Harrold  Gregg Rothermel  Rui Wu  Liu Yi
harrold@cis.ohio-state.edu  grother@cs.orst.edu  rwu@cis.ohio-state.edu  liuyi@cs.orst.edu

Abstract

A variety of expensive software maintenance and testing tasks require a comparison of the behaviors of program versions. Program spectra have recently been proposed as a heuristic for use in performing such comparisons. To assess the potential usefulness of spectra in this context, we conducted an experiment that examined the relationship between program spectra and program behavior, and empirically compared several types of spectra. This paper reports the results of that experiment.

1 Introduction

A variety of software testing and maintenance tasks require us to compare the behaviors of multiple program versions. For example, when we modify a program, we use regression testing to compare the behavior of the modified version to the behavior of its previous version, in the hope of detecting faults caused by the modifications. Similarly, when programs exhibit “regression failures” (behavioral failures that did not occur in preceding versions), we compare the behaviors of versions in the hope of pinpointing the cause of those failures. Tasks such as these constitute a significant percentage of the costs of software testing and maintenance; techniques that reduce these costs are valuable.

Path spectra were recently proposed as a heuristic for understanding the magnitude of the behavioral changes between program versions [10]. A path spectrum is a distribution of paths derived from an execution of a program using program profiling. A path-spectra-comparison technique compares path spectra to gain insight into program behavior. Such a technique may aid in addressing testing and maintenance tasks that require such understanding. For example, constructing expected outputs for programs can be costly; the presence of spectra differences may serve as an indicator of cases in which that construction is unnecessary. Alternatively, spectra comparison [10] may help programmers locate points of divergence in computations, that may guide them in fault localization.

For path spectra to be useful in these contexts, however, they must provide meaningful behavior signatures. An assessment of the potential usefulness of path-spectra comparisons requires an understanding of the correlation between spectra differences and program behavior.

Program behavior can be measured in many ways; however, one measure — important to uses of spectra such as those described above — considers whether particular inputs cause a program to fail. Reference [10] hypothesizes a strong correlation between spectra differences and faults, at least in one direction, stating that given a faulty program and corrected version, one would expect differences between spectra on an input that produces the bug in the original program. One goal of this work is to empirically investigate this claim.

If path spectra prove useful then other spectra such as branch spectra or complete-path spectra may also be useful, and may provide a range of techniques, varying in cost and effectiveness, for examining program behavior. Reference [10] conjectures, however, that edge and node spectra will not be as useful as path spectra for distinguishing program behavior. There is some empirical

---

*This work was supported in part by a grant from Microsoft Inc., by NSF under NYI Award CCR-9696157 to Ohio State University, ESS Award CCR-9707792 to Ohio State University and Oregon State University, and CAREER Award CCR-9703108 to Oregon State University.

The primary use of spectra investigated in [10] addresses the “Year 2000 problem,” and involves comparing spectra from two runs of the same program on input data that differs only with respect to date. The intuition is that spectra differences may help programmers locate date-dependent computations. The alternative use of spectra that we investigate here, in which spectra are collected from runs of a program and a slightly different version of the program on the same data, is briefly described in [10], but not pursued in detail. The goal of this work is to empirically investigate this alternative suggestion.
data [1] to support this conjecture; this data indicates that path profiling data may be superior to edge profiling data for certain applications. Another recent study [4], however, suggests the contrary. These studies have not directly investigated program spectra. A second goal of this work is to perform such an investigation.

## 2 Program Spectra

A program spectrum characterizes, or provides a signature of, a program’s behavior [10]. Path spectra use path profiling [1, 3] to track the execution of loop-free intraprocedural paths in a program. Path spectra can track the frequency of a path occurrence, or ignore frequency and track whether or not the path occurred. Spectra can also be constructed based on node or edge profiling data.

In addition to path, node, and edge spectra, many additional signatures of program behavior can be treated as spectra. To obtain a broader empirical view of the relationships among spectra, we consider nine distinct types of spectra. We next describe these spectra (Table 1), and use an example to illustrate them. In the following, let $P$ be a program.

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BHS</td>
<td>Branch-hit</td>
<td>conditional branches that were executed</td>
</tr>
<tr>
<td>BCS</td>
<td>Branch-count</td>
<td>number of times each conditional branch was executed</td>
</tr>
<tr>
<td>PHS</td>
<td>Path-hit</td>
<td>path [intraprocedural, loop-free] that was executed</td>
</tr>
<tr>
<td>PCS</td>
<td>Path-count</td>
<td>number of times each path [intraprocedural, loop-free] was executed</td>
</tr>
<tr>
<td>CPS</td>
<td>Complete-path</td>
<td>complete path that was executed</td>
</tr>
<tr>
<td>DSH</td>
<td>Data-dependence-hit</td>
<td>definition-use pairs that were executed</td>
</tr>
<tr>
<td>DCS</td>
<td>Data-dependence-count</td>
<td>number of times each definition-use pair was executed</td>
</tr>
<tr>
<td>OPS</td>
<td>Output</td>
<td>output that was produced</td>
</tr>
<tr>
<td>ETS</td>
<td>Execution-trace</td>
<td>execution trace that was produced</td>
</tr>
</tbody>
</table>

Table 1: A catalog of program spectra.

If, for each loop-free intraprocedural path in $P$, the spectrum indicates the number of times that path was executed, the spectrum is a path-count spectrum (PCS).

**Complete-path spectrum.** A complete path spectrum (CPS) records the complete path that is traversed as $P$ executes.

**Data-dependence spectra.** Data-dependence spectra record the set of definition-use pairs that are exercised as $P$ executes. If, for each definition-use pair in $P$, the spectrum merely indicates whether or not that definition-use pair was exercised, the spectrum is a data-dependence-hit spectrum (DHS). If, for each definition-use pair in $P$, the spectrum indicates the number of times that definition-use pair was exercised, the spectrum is a data-dependence-count spectrum (DCS).

**Output spectrum.** An output spectrum (OPS) records the output produced by $P$ as it executes.

**Execution-trace spectrum.** An execution trace spectrum (ETS) records the sequence of program statements traversed as $P$ executes.

At first glance, CPS and ETS may appear to be the same, because they both record the complete control flow path through $P$. They differ, however, in that CPS does not record the actual instructions executed along that path, whereas ETS does.

OPS and ETS are of interest in this context because of their relationship to regression testing. One important regression testing activity is the selection of a subset of the test suite that was originally used to test the program for use in testing the modified program; Reference [11] provides details. In brief, given a program $P$, test suite $T$ for $P$, and modified version $P'$, we want to identify the tests in $T$ that reveal faults in $P'$. For tests whose specified behavior has not changed, these “fault-revealing” tests are exactly the tests that produce different output spectra on $P$ and $P'$. In general, there is no algorithm to precisely identify these tests,
but under certain conditions, the tests that produce different execution trace spectra constitute a conservative (safe) approximation. Several regression test selection techniques [2, 5, 12] exploit this relationship to select safe subsets of $T$ for use in regression testing $P'$.

To illustrate these spectra, we present program Sums and its control flow graph (Figure 1), and the spectra for Sums on two executions (Table 2): execution 1 uses input 10 and has expected output 6, and execution 2 uses inputs 8, 2, and 4 and has expected output 6. Where applicable, the two types of spectra in each category – hit and count – are shown in columns on the right in the table; where the spectra category does not have these subtypes, “NA” is listed. Consider, for example, the branch spectra for Sums. There are three conditional branches in Sums, and the two executions exercise all of them: the BHS for execution 1 records Y for edges (1,2), (3,4), and (3,7); the BHS for execution 2 records Y for edges (1,2), (3,4), and (3,7); the BCS for execution 1 records that edges (1,2) and (3,7) were each exercised once; and the BCS for execution 2 records that edges (1,2) and (3,7) were exercised once, and edge (3,4) was exercised twice.

We can analytically compare types of spectra by determining a subsumption relationship between them. A spectra type $S_1$ subsumes spectra type $S_2$ if and only if, whenever the $S_2$ spectra for program $P$, version $P'$, and input $i$ differ, the $S_1$ spectra for $P$, $P'$, and $i$ differ. Reference [10] discusses the subsumption relationship that exists between path and edge (and thus branch) spectra; the following theorem establishes the subsumption relationship for all of the spectra we are considering.

**Theorem 1:** The spectra listed in Table 1 form the subsumption hierarchy shown in Figure 2, in which the spectra type that is the source of an edge directly subsumes the spectra type that is the target of that edge.

**Proof:** See Reference [7].

![Figure 1: Sums and its control flow graph.](image)

---

3 The Experiment

3.1 Objectives

The objectives of our experiment were to empirically investigate the following questions:

1. Given program $P$, faulty version $P'$, and universe of inputs $U$ for $P$, what correlation exists between inputs that cause $P$ and $P'$ to produce different spectra. More precisely:
   
   (a) How often does an input $i \in U$ that causes $P'$ to fail produce different spectra for $P$ and $P'$?
   
   (b) How often does an input $i \in U$ that produces different spectra for $P$ and $P'$ cause $P'$ to fail?

2. What are the relationships between the various spectra types, both in terms of their correlation with program-failure behavior, and in terms of their correlation with one another?

3.2 Measures

To quantify 1(a) and 1(b), we utilize two measures. Given program $P$, faulty version $P'$, and universe of inputs $U$ for $P$, let $FR(P, P', U)$ be the set of inputs in $U$ that cause $P'$ to fail (i.e., $FR(P, P', U)$ are Fault Revealing for $P, P'$, and $U$). For each spectra variety $S$, let $SR(P, P', U)$ be the set of inputs in $U$ that produce spectra on $P$ and $P'$ that differ (i.e., $SR(P, P', U)$ are spectrum $S$ Revealing for $P, P'$, and $U$).

**Degree of Imprecision.** For spectra type $S$, any input $i$ in $U$ that is in $SR(P, P', U)$ but not in $FR(P, P', U)$ exhibits a spectra difference under $S$ that is not correlated with a failure. For such an input $i$, $S$-spectra-comparison is “imprecise.” The degree of imprecision of $S$ with respect to $P, P'$ and $U$ is given by the equation:

$$\left| \frac{|SR(P, P', U) - FR(P, P', U)|}{|SR(P, P', U)|} \right| \times 100$$

(1)
### Degree of Unsafety

For spectra type $S$, any input $i$ in $U$ that is in $FR(P, P', U)$ but not in $SR(P, P', U)$ exhibits a failure that is not correlated with a spectra difference of type $S$. For such an $i$, $S$-spectra-comparison is “unsafe”. The degree of unsafety of $S$ with respect to $P, P'$, and $U$ is given by the equation:

$$\frac{|FR(P, P', U) - SR(P, P', U)|}{|FR(P, P', U)|} \times 100 \quad (2)$$

Note that we can determine the degree of imprecision of $S$ even though $S$ is not safe. In this respect, our use of the term “imprecision” differs from its use in areas such as compiler optimization, where safe analyses are required. In the context of maintenance and testing, however, safe analyses are not always necessary – a technique that identifies a sufficiently large subset of some set of facts can be useful, even though it omits some members of that set of facts. However, we still wish to know, even of unsafe analyses, the degree to which they identify spurious results. Our use of “imprecision” supports this.

Note further that when $P$ is correct for input $i$, and when $P'$ is intended to have the same output for $i$ as $P$, then $i$ is in $FR(P, P', U)$ if and only if $P$ and $P'$ produce different output for $i$, or equivalently, if and only if $i$ is in $OPS\bar{R}(P, P', U)$.

### 3.3 Experimental Instrumentation

We used seven C programs as experimental subjects (Table 3). Each program has a variety of versions — each containing one fault-inducing modification — whose control flow graph structure is the same as that of the original program. Each subject program also has a large universe of inputs.\(^2\) By construction, for each of our experimental programs $P$ (with their universe $U$) and versions $P'$, $FR(P, P', U) = OPS\bar{R}(P, P', U)$.

#### Table 2: Spectra for program Sum of Figure 1.

<table>
<thead>
<tr>
<th>Spectrum</th>
<th>Profiled entities</th>
<th>Execution 1 (input is 10)</th>
<th>Execution 2 (input is 8, 2, 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>(1, 2), (3, 4), (3, 7)</td>
<td>Y 1</td>
<td>N 0</td>
</tr>
<tr>
<td>Path</td>
<td>(1, 2, 3, 7), (1, 3, 7), (1, 2, 3, 4, 5, 6, 7), (1, 3, 4, 5, 6, 7)</td>
<td>Y 1</td>
<td>N 0</td>
</tr>
<tr>
<td>Complete-path</td>
<td>(1, 2, 3, 7), (1, 2, 3, 4, 5, 6, 7)</td>
<td>Y NA</td>
<td>N NA</td>
</tr>
<tr>
<td>Data-dependence</td>
<td>(1, 3, 7), (1, 2, 3, 4, 5, 6, 7), (1, 3, 4, 5, 6, 7)</td>
<td>Y 1</td>
<td>N 0</td>
</tr>
<tr>
<td>Output</td>
<td>sum is 0</td>
<td>Y NA</td>
<td>N NA</td>
</tr>
<tr>
<td>Execution-trace</td>
<td>(S1, S2, S3, S7), (S1, S2, S3, S4, S5, S6, S8)</td>
<td>Y NA</td>
<td>N NA</td>
</tr>
</tbody>
</table>

#### Table 3: Subject programs.

We used a variety of tools and techniques to compute and record the various types of spectra. For the output spectra (OPS), we ran $P$ and the $P'$s on the inputs in $U$. For the execution trace spectra (ETS), we used our test selection tool, DejaVu [12], to identify the inputs in $U$ that traverse modified statements in the $P'$s. For the branch-hit spectra (BHS), the branch-count spectra (BCS), the path-hit spectra (PHS), the path-count spectra (PCS) the data-dependence-hit spectra (DHS), the data-dependence-count spectra (DCS), and the complete-path spectra (CPS) we used the various coverage tools from the Aristotle analysis system [6] and the FATE data-flow testing system [9] to record the enti-

\(^2\) These programs, versions, and inputs were assembled by researchers at Siemens Corporate Research for a study of the fault-detection abilities of control- and data-flow coverage criteria [8].

\(^3\) In general this approach may identify a superset of the inputs that produce different execution traces; however, in practice we can determine when the approach incurs imprecision, and we know that in all the cases examined for our experimentation, the algorithm identified precisely the inputs that produced different execution traces.
ties (i.e., branches, paths, definition-use pairs, complete paths) executed in P and the P's.

3.4 Experimental Design

Variables. The experiment manipulated a single independent variable, namely, the spectra: OPS, ETS, BHS, BCS, CPS, PHS, PCS, DHS, and DCS. On each run (with P, P', and U), we measured a single dependent variable, namely, the set of inputs in U that revealed spectra differences between P and P'. We utilized this data to examine the degree of imprecision and unsafety of the various spectra, and to compare spectra, using an analysis strategy described later in this section.

Design. This experiment uses a within-subjects design: we applied each spectra calculation to each (base program, modified version) pair, for each input in the universe for that base program. We then examined the spectra with respect to the OPS results obtained on that base program, version and universe, to measure the values of dependent variables.

Threats to Validity. The primary threats to validity for this experiment are external: these are conditions that limit our ability to generalize the results of our study to a larger population of subjects. First, the subject programs are not large, and we cannot claim that they represent a random selection over the population of programs as a whole. Second, the faulty program versions all involve simple, one- or two-line faults, manually seeded, with the intent of simulating “real” faults, but with no data to indicate that they represent a random selection over the population of faults as a whole. These threats can be reduced only by repeated application of the experiment on wider classes of subjects.

A second source of threats to validity for this study are internal: these are influences that can affect the dependent variables without the researchers’ knowledge. Our greatest concerns here involve instrumentation effects, which can bias our results. To control for such effects, we utilized two types of cross-checks: (1) we obtained certain results independently using different instruments at different sites and examined them for correlation; (2) we examined results for adherence to the analytically determined spectra hierarchy relationship. We did not, however, control for the structure of source programs or for the locality of program changes.

Analysis Strategy. We utilized the data gathered from our experimental runs as follows. First, to examine the correlation between inputs that cause P' to fail and inputs that cause P to produce different spectra than P' (objective 1), for each program P and version P', with respect to universe U, and for each spectra type S, we calculated:

1. The degree of imprecision of S with respect to P, P', and U.

2. The degree of unsafety of S with respect to P, P', and U.

Second, to compare spectra (objective 2), for each program P and version P', with respect to universe U, and for each pair of spectra types S_1 and S_2, we calculated:

1. The number of inputs in U that cause spectra differences of type S_1.

2. The number of inputs in U that cause spectra differences of type S_2.

3. The number of inputs in U that cause spectra differences of type S_1 but not of type S_2.

4. The number of inputs in U that cause spectra differences of type S_2 but not of type S_1.

We also summarized this data over the entire set of (program, modified version) pairs, considering each for the entire universe U of inputs (245,087 inputs). The next section presents and analyzes this data.

4 Data and Analysis

Figure 3 uses boxplots to present the degrees of unsafety and imprecision calculated for each spectra over the 88 different (program, modified program) pairs. (The caption provides an explanation of boxplots.) The data reveals several things relative to our first objective. First, the unsafety data (left) supports the conjecture that we can expect to see spectra differences on inputs that elicit faults. Every variety of spectra demonstrates a median degree of unsafety of 0%. Furthermore, four varieties of spectra (CPS, PCS, BCS, and DCS) demonstrate a 0% degree of unsafety over their entire first, second, and third quartiles; in other words, over three quarters of the (program, modified program) pairs, these spectra demonstrated differences on every input that elicited a fault. For the BHS and PHS spectra, the degrees of unsafety demonstrate a greater interquartile range, reflecting greater diversity in results. Finally, all spectra other than ETS displayed occasional extreme results, represented by the outliers, and further reflected in the fact that their means are noticeably higher than their medians. Significantly, only ETS is safe: in all cases, all inputs that reveal faults also reveal ETS spectra differences, and no other spectra share this trait.
The imprecision data presented in Figure 3 (right) illustrates that not all inputs that elicit spectra differences also produce failures: all varieties of spectra incur imprecision. Significantly, ETS incurs a much higher median degree of imprecision (95%) than the other varieties of spectra, for which median degree of imprecision ranges from 3% to 17%. Thus, the cost of the safety achieved by ETS, in terms of degree of imprecision, is high. However, imprecision results for all spectra other than ETS display a large interquartile range; thus, degree of imprecision varies a great deal more, overall, than degree of unsafety.

One additional significant result is that in terms of both degrees of imprecision and unsafety the CPS, PCS, and BCS spectra display nearly identical behavior. These results do not support the conjecture that path spectra will be more sensitive indicators of different behavior than branch spectra, where behavior is measured in terms of program failure behavior.

Toward our second objective, Table 4 lists data pertaining to the relationship between various spectra. Figure 4 provides a graphical view of the portion of that data that compares the various spectra with OPS, within the context of the entire universe of inputs as applied to all (program, modified program) pairs (see the caption for further description.) A notion of the relationship among the spectra can be obtained by comparing the sizes of shaded areas in Figure 4. In particular, it is easy to see the relationship between ETS, DCS, DHS, and CPS from this figure.

Figure 5 provides a closer view of the relationship between CPS, PCS, PHS, BCS, and BHS that shows, for each of these spectra types, the number of inputs for which spectra differences occurred. Like the imprecision and unsafety results, this figure illustrates the near equivalence, over our programs, modified versions, and input universes, of the CPS, PCS, and BCS spectra, as well as the relative closeness of the PHS and BHS spectra. Furthermore, where analytical results foretell that neither PHS nor BCS subsumes the other, in practice BCS subsumes PHS: BCS spectra differences occurred in 5836 cases in which PHS spectra differences did not occur, but in all cases in which PHS spectra differences occurred, BCS spectra differences also occurred.

These results do not support the conjecture that path spectra will be more sensitive indicators of different behavior than branch spectra. For the 245,087 inputs, PCS is never more sensitive than BCS, and CPS is more sensitive than PCS on only 7 inputs. Essentially, despite the analytical differences between spectra depicted by our subsumption hierarchy, in the cases we studied, the CPS, PCS and BCS spectra collapse into
one another. The three spectra have nearly equivalent abilities to distinguish differences in program behaviors.

Figure 4: Graphical comparison of OPS with the other spectra. The eight outer squares represent the comparison of the OPS spectra to the other eight spectra, respectively, as labeled. Each such square represents the entire universe of input points over all (program, modified version) pairs. Within the outer squares, the lightly shaded areas indicate the percentages of input points that caused only $XS$-spectra differences ($XS \neq OPS$), the medium shaded areas represent the percentages of input points that caused only OPS-spectra differences, and the darkly shaded areas represent the percentages of input points under both $XS$ and $OPS$.

![Graphical comparison of OPS with other spectra](image)

5 Conclusions

We have described an empirical investigation of the correlation between spectra and program failure behavior, and the relationships between various types of spectra.

We emphasize that our study has considered only one scenario in which the usefulness of program spectra has been postulated: the scenario in which spectra from a program and modified version, run on the same input, are compared. Our results do not provide data applicable to the use of spectra when a single program is run on two slightly different inputs. However, given the degree to which spectra types produced nearly equivalent results for our subjects, further empirical study of the relationship between spectra types in the latter scenario would be appropriate.

We also stress that our study has focused on just one indicator of program behavior – fault-revealingness. This indicator is important, and the fact that spectra do correlate with it – at least in one direction – is significant. Whether spectra will correlate with other measures of behavior, such as measures based on sequences of execution states [10], is a subject for future investigation. Our comparisons of spectra to each other, however, are not restricted to fault-revealing behavior.

Finally, as discussed earlier, there are some threats to validity for this experiment, primarily concerning representativeness of subjects. Additional experimentation with a variety of subjects is necessary to lessen these threats.

Keeping the foregoing in mind, however, our results support several conclusions. First, although the execution trace spectra emerged as the only spectra to necessarily exhibit differences for inputs that exhibit faults, certain other types of spectra differences (i.e., CPS, PCS, and BCS) also correlate with high frequency (at
Table 4: Comparison of spectra summarized over all (program, modified version) pairs, considering each for the entire input universe (245,087 inputs). Column A lists the spectra compared; Column B lists the total number of inputs that cause spectra differences of type S1; Column C lists the total number of inputs that cause spectra differences of type S2; Column D lists the total number of inputs that cause spectra differences of type S1 but not of type S2; Column E lists the total number of inputs that cause spectra differences of type S2 but not of type S1.

Cost against the potential savings, while also considering the criticality of the application. In many situations, it may not be cost-effective to collect the complete traces required for CPS spectra; however, if our results generalize, PCS and BCS spectra possess power almost equivalent to that of CPS, and may be cost-effective alternatives. Furthermore, estimates suggest that profiling necessary to collect BCS spectra incurs a 16% run-time overhead whereas the profiling necessary to collect PCS spectra incurs a 30% run-time overhead [4]. In the absence of a gain in sensitivity, use of the PCS spectra instead of the BCS spectra would not be worth the added overhead required to collect the more sensitive spectra.

References