Statements versus predicates in spectral bug localization

Lee Naish
Hua Jie (Jason) Lee
Kotagiri Ramamohanarao

Computer Science and Software Engineering
University of Melbourne

Outline

Statement spectra

Predicate spectra

Raw data versus aggregated data

Predicate spectra from statement spectra

Theoretical performance

Practical performance

Related work

Conclusion

Statement spectra for bug localization

Basic idea:

Execute the program multiple times using a test suite where we can tell if each result is correct or not, gathering data about each execution

For each statement/..., estimate how likely it is to be buggy based on the data gathered

Rank the statements/... accordingly, then check the code manually, starting with the highest ranked statement until the bug is found (or we give up)

Statement spectra

Collect data on which statements are executed for each test

We count

- The total number of failed tests, F,
- The total number of passed tests, P,

and for each statement S_i, the number of

- failed tests in which it was executed, f_i, and
- passed tests in which it was executed, p_i.

(the number of failed/passed tests *not* executing S_i is implicit in our presentation)

Statement spectra example

The raw data is a binary matrix and a binary vector

We compute F, P and the f_i and p_i for each statement

	T_1	T_2	T_3	T_4	T_5	f	p
Statement_1	1	0	1	1	1	1	3
Statement_2	1	1	0	1	0	2	1
Statement_3	0	1	0	1	1	1	2

$\vdots$

Test Result	1	1	0	0	0

$F = 2,\ P = 3$

Measure "similarity" of matrix rows and result vector

Statement metrics used for ranking

Name	Formula	Name	Formula
Jaccard	$\frac{f}{F+p}$	Tarantula	$\frac{\frac{f}{F}}{\frac{f}{F}+\frac{p}{P}}$
Russell	$\frac{f}{F+P}$	O^p	$f - \frac{p}{P+1}$
Ample	$\left\lvert \frac{f}{F} - \frac{p}{P} \right\rvert$	Ochiai	$\frac{f}{\sqrt{F*(f+p)}}$
Zoltar	$\frac{f}{F+p+\frac{10000F-f*p}{f}}$		
Wong3	$f - h,$ where $h = \begin{cases} p & \text{if } p \leq 2 \\ 2 + 0.1(p-2) & \text{if } 2 < p \leq 10 \\ 2.8 + 0.001(p-10) & \text{if } p > 10 \end{cases}$		

O^p has been proven optimal for a class of single-bug programs

Predicate spectra

Instrument predicates, eg conditions of if-then-else

Also can compare values returned from functions with zero, etc

Count the number of failed/passed tests where the predicate is true (f and p)

Also count the number of failed/passed tests where the predicate is reached/observed (f' and p')

This gives four integers, just like for statement spectra

Predicate metrics

Name	Formula	Name	Formula
Failure	$\frac{f}{f+p}$	Context	$\frac{f'}{f'+p'}$
Increase	$Failure - Context$	FP	$\frac{f}{F}$
Log	$\frac{2}{\frac{1}{Increase}+\frac{\log F}{\log f}}$	Sqrt	$\frac{2}{\frac{1}{Increase}+\frac{\sqrt{F}}{\sqrt{f}}}$
FPC	$Failure + Context$	OFPC	$3f + FPC$
O8FPC	$10f + 8Failure + Context$		

Increase, Log and Sqrt have been used in the CBI system

FPC, OFPC and O8FPC are new; the latter two are based on O^p

FP is equivalent to Russell for ranking

Failure is equivalent to Tarantula (as is Increase if Context is constant)

Raw data and aggregated data

There is no essential difference in the raw data (the binary matrix) collected from predicates and statements

```
1:    if (C)
2:            T;
3:    else    E;
```

Instrumenting the three statements gathers identical information to instrumenting predicate C and ¬C

C and ¬C are observed if and only if line 1 is executed

C is true if and only if line 2 is executed

¬C is true if and only if line 3 is executed

Any instrumented predicate can be turned into an if-then-else with no-ops for T and E

Raw data and aggregated data

Statement metrics have just two degrees of freedom:

$$\sum_{p=0}^{P}\sum_{f=0}^{F} 1 = (P+1)(F+1)$$

Predicate metrics use counts from two different statements:

$$\sum_{p'=0}^{P}\sum_{f'=0}^{F}\sum_{p=0}^{p'}\sum_{f=0}^{f'} 1$$

Thus predicate spectra retain more information

In our unified framework we have F and P ("global" variables), and f, p, f' and p' (for each statement/predicate)

Statement metrics just don't use f' or p'

Predicate spectra from statement spectra

We use spectra from *consecutive statements* to guess the program structure and extract predicate/branch spectra (f' and p')

```
1: if (C1) {
2:           S2;
3:           S3;
4: } else { S4;
5:           S5; }
6: S6;
```

If S2 is executed less than S1 we guess its the start of a "then" ($f'_2 = f_1$ and $p'_2 = p_1$)

If S5 is executed less than S6 we guess its the end of an "else"

We scan forwards/backwards to identify other statements with identical spectra to find the rest of the then/else

Model buggy program ITE2

```
if (C1) S1; else S2;
if (C2) S3; else S4 /* BUG */;
```

Executing S4 leads to failure 1/2 the time on average

Given number of tests N, generate multisets of N execution paths

For each multiset, compute spectra, ranking and performance

Compute *overall* performance of each metric

O^p is optimal

Too simple for evaluating predicate metrics — Context (f' and p') the same for S1–S4

Model buggy program

```
if (C1) {
    if (C2) S3; else S4;
    S5;
} else
    S6;
if (C7) {
    if (C8) S9; else S10;
    S11;
} else
    S12;
S13;
```

S3 is buggy; We rank all 13 statements

Context varies between statements

Ideal performance (rank %) with model

Number of tests	5	20	50	200
Statement metrics				
O^p	12.71	8.18	7.75	7.69
Zoltar	12.74	8.20	7.75	7.69
Ochiai	12.74	8.23	7.76	7.69
Jaccard	12.74	8.27	7.78	7.70
SLog	12.78	8.31	7.83	7.72
Ample	16.18	8.33	7.75	7.69
Wong3	15.31	8.44	7.75	7.69
SSqrt	13.19	8.47	7.85	7.76
Tarantula	13.19	8.57	7.88	7.71
Russell	33.52	30.51	28.06	26.96

Ideal performance with model

Number of tests	5	20	50	200
Predicate metrics				
O8FPC	11.21	8.14	7.74	7.69
OFPC	11.21	8.18	7.75	7.69
FPC	11.96	8.54	7.88	7.71
Log	21.64	10.78	9.04	8.22
Increase	22.16	11.29	9.16	8.22
Sqrt	29.95	11.34	9.10	8.24
Context	21.82	18.03	16.51	15.64

Performance with model using heuristics

Num. tests	5		20		50	
Heuristic	Then	Scan	Then	Scan	Then	Scan
O8FPC	11.21	12.21	8.14	8.14	7.74	7.74
OFPC	11.21	12.21	8.18	8.11	7.75	7.74
FPC	11.84	13.00	8.53	8.45	7.87	7.86
Log	21.38	13.33	10.82	12.43	9.06	10.03
Increase	22.07	14.23	11.38	13.00	9.18	10.16
Sqrt	30.49	14.03	11.42	12.86	9.13	10.10
Context	25.38	33.10	20.76	16.26	18.33	14.09

Practical performance

Metric	STS+Unix	Concordance
Statement metrics		
O^p	17.86	10.11
Wong3	18.19	10.15
Zoltar	18.23	10.11
Ochiai	21.63	11.19
Jaccard	23.64	17.68
SLog	26.23	19.59
SSqrt	26.77	20.42
Tarantula	27.09	20.03
Ample	30.16	27.53
Russell	30.02	21.03

Practical performance

Metric	STS+Unix	Concordance
Predicate metrics		
OFPC	17.94	9.95
O8FPC	17.94	10.08
FPC	28.24	19.90
Context	30.18	20.00
Log	44.93	49.05
Increase	45.41	50.04
Sqrt	47.01	51.18

Holmes — path spectra

The Holmes system uses path spectra (sequences of statements)

p and f computed for each path, also p' and f' ("observed" means the first statement in the path is executed)

Log is used to rank paths; not all paths are returned

Comparative performance (???)

What proportion of bugs are found by examining 50% of the code?

Method	% bugs found
Holmes — paths	78
Holmes — predicates	51
Holmes — branch only	9
CBI (predicates)	75
Statements — O^p/OFPC	90
Statements — SLog	83
Statements — Log	54
Random	50

Conclusions

Predicate and statement spectra are based on the same raw data but the method of aggregation used in predicate spectra retains more information

The additional information can be reconstructed from statement spectra with reasonable precision using heuristics

The additional information can improve performance, at least in theory

Metrics used previously for ranking predicates can be significantly improved