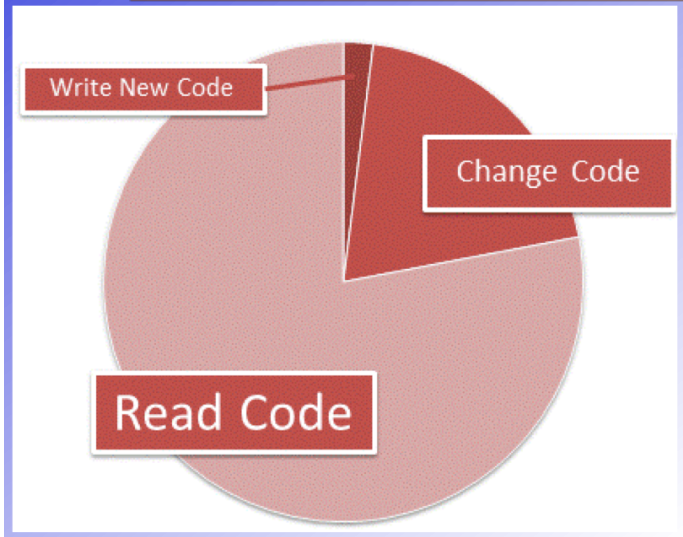
# Modifying Test Suite Composition for Effective Statistical Debugging

MaSTRI

Ross Gore
University of Virginia
rjg7v@virginia.edu

# Motivation

MaSTRI

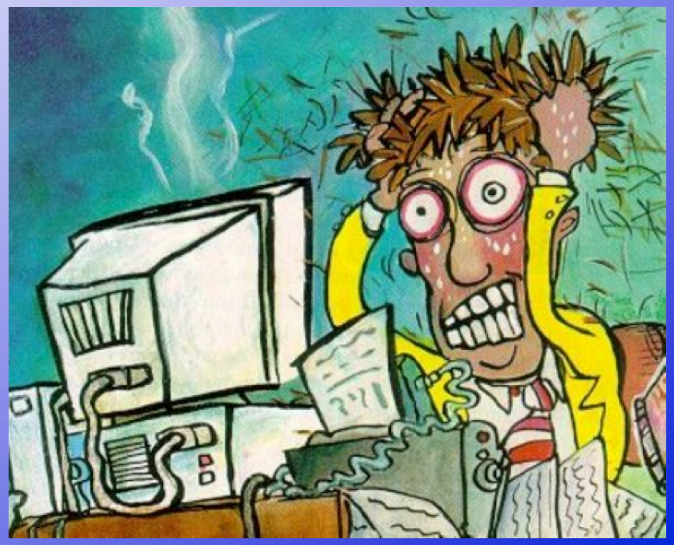



Efficient debugging is needed:

- Lots of software ships with faults
- Programmers read code to fix faults

Harder for safety-critical:

- Floating-point data types and computations
- Numerical Analysis errors
- Affects millions of lives and billions of $$



# Predicate-level Statistical Debugging

MaSTRI

Given a failing subject program, rank the likelihood that each predicate (*p*) reflects the fault.

$$Suspiciouness\left(p\right) = \frac{f_p}{f_p + s_p}$$

*Suspiciousness*(*p*) based on:

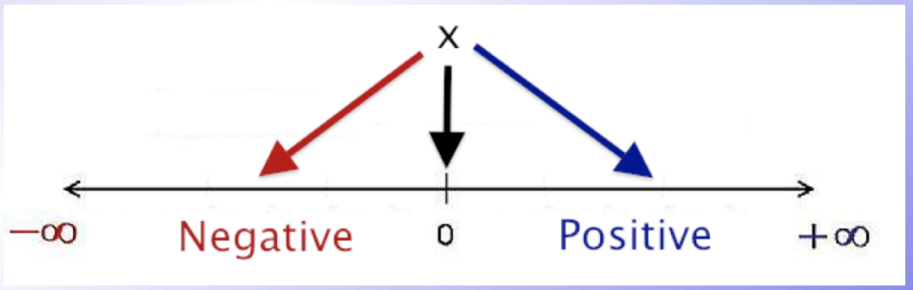
- Test Cases (Program Inputs)
- Execution Profiles
- Status of Test Cases
  - Successful (passing) - *s*
  - Failing - *f*

# Elastic Predicates

MaSTRI

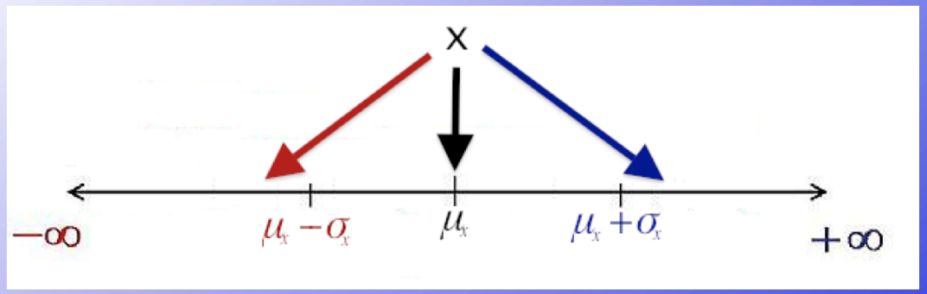
## Static Predicates

- All variables (x) partitioned the same
  - Value is negative, x < 0
  - Value is zero, x = 0
  - Value is positive, x > 0




## Elastic Predicates

- Variables (x) partitioned based on observed values
  - Value is a lot higher than average, $x > \mu_x + \sigma_x$
  - Value matches average, $x = \mu_x$
  - Value is a lot lower than average, $x < \mu_x - \sigma_x$



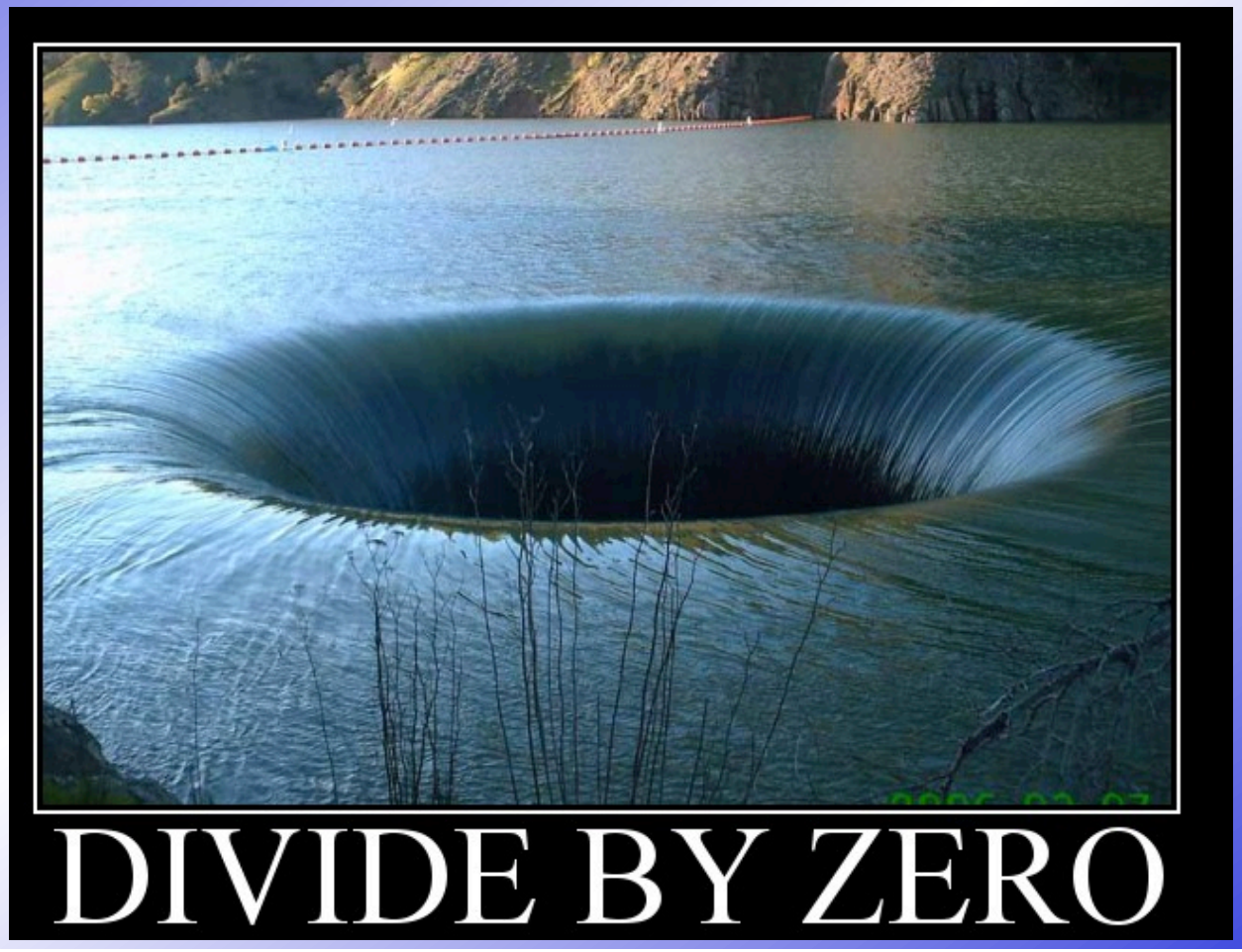
# Predicate-level Statistical Debugging

MaSTRI

```
 1  double
 2  mean(int x [])
 3  {
 4      int i = 0;
 5      double sum = 0;
 6      while (i > x.length){
 7          sum = sum + x[i];
 8          i = i + 1;
 9      }
10      if (x.length >= 0){ /** off by one **/
11          return (sum / x.length);
12      }
13      else {
14          return (0);
15      }
16  }
```

# Divide by Zero

MaSTRI



# Predicate-level Statistical Debugging

MaSTRI

$(x.length = 0)_6$

$(x.length = 0)_{10}$

$(x.length = 0)_{11}$

```
1  double
2  mean(int x [])
3  {
4      int i = 0;
5      double sum = 0;
6      while (i > x.length){
7          sum = sum + x[i];
8          i = i + 1;
9      }
10     if (x.length >= 0){ /** off by one **/
11         return (sum / x.length);
12     }
13     else {
14         return (0);
15     }
16 }
```

# Elastic Predicate Summary

MaSTRI

- Elastic predicates enable improved effectiveness
  - Floating point computations
  - Numerical analysis errors
  - Incurs additional space and time
- Improvements hold in the face of:
  - Sparse Sampling
  - Incomplete Test Suites

# Motivating Example

MaSTRI

```
 1  int
 2  distance(int x, int y)
 3  {
 4       int diff = x - y;
 5       if (!( diff > 1)){ /* off by one */
 6            int dist = 0;
 7            dist = y - x;
 8            print(dist);
 9       }
10       int dist = 0;
11       dist = x - y;
12       return dist;
13  }
```

# Motivating Example

MaSTRI

| Statement | Predicate | TC 1: {2,2} | TC 2: {5,4} | TC 3: {5,1} | TC 4: {-4,-2} | TC 5: {1,0} | Ranks |
|---|---|---|---|---|---|---|---|
| 5 | diff = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 5 | diff > 0 | 0 | 1 | 0 | 0 | 1 | ***3*** |
| 5 | diff < 0 | 0 | 0 | 1 | 1 | 0 | 12 |
| 6 | dist = 0 | 1 | 1 | 1 | 1 | 1 | 4 |
| 6 | dist > 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 6 | dist < 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist > 0 | 0 | 0 | 1 | 1 | 0 | 12 |
| 7 | dist < 0 | 0 | 1 | 0 | 0 | 1 | ***3*** |
| 8 | dist = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 8 | dist > 0 | 0 | 0 | 1 | 1 | 0 | 12 |
| 8 | dist < 0 | 0 | 1 | 0 | 0 | 1 | ***3*** |
| OUTCOME | ——— | PASS | FAIL | PASS | PASS | FAIL | ——— |

# Motivating Example

MaSTRI

| Statement | Predicate | TC 1: {2,2} | TC 2: {5,4} | TC 3: {5,1} | TC 4: {-4,-2} | TC 5: {1,0} | Ranks |
|---|---|---|---|---|---|---|---|
| 5 | diff = 0 | 1 | 0 | 0 | 0 | 0 | |
| 5 | diff > 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| 5 | diff < 0 | 0 | 0 | 1 | 1 | 0 | |
| 6 | dist = 0 | 1 | 1 | 1 | 1 | 1 | 4 |
| 6 | dist > 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 6 | dist < 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist > 0 | 0 | 0 | 1 | 1 | 0 | |
| 7 | dist < 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| 8 | dist = 0 | 1 | 0 | 0 | 0 | 0 | |
| 8 | dist > 0 | 0 | 0 | 1 | 1 | 0 | |
| 8 | dist < 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| OUTCOME | ——— | PASS | FAIL | PASS | PASS | FAIL | |

# Motivating Example

MaSTRI

| Statement | Predicate | TC 1: {2,2} | TC 2: {5,4} | TC 3: {5,1} | TC 4: {-4,-2} | TC 5: {1,0} | Ranks |
|---|---|---|---|---|---|---|---|
| 5 | diff = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 5 | diff > 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| 5 | diff < 0 | 0 | 0 | 1 | 1 | 0 | |
| 6 | dist = 0 | 1 | 1 | 1 | 1 | 1 | 4 |
| 6 | dist > 0 | 0 | 0 | 0 | 0 | 0 | |
| 6 | dist < 0 | 0 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 7 | dist > 0 | 0 | 0 | 1 | 1 | 0 | 12 |
| 7 | dist < 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| 8 | dist = 0 | 1 | 0 | 0 | 0 | 0 | 12 |
| 8 | dist > 0 | 0 | 0 | 1 | 1 | 0 | 12 |
| 8 | dist < 0 | 0 | 1 | 0 | 0 | 1 | *3* |
| OUTCOME | ——— | PASS | FAIL | PASS | PASS | FAIL | ——— |

# Confounding Bias

MaSTRI

| Statement | Predicate | Abbreviated Name |
|---|---|---|
| 5 | `diff > 0` | $(\texttt{diff} > 0)_5$ |
| 7 | `dist < 0` | $(\texttt{dist} < 0)_7$ |
| 8 | `dist < 0` | $(\texttt{dist} < 0)_8$ |

$(diff > 0)_5$ causes $(dist < 0)_7$ and $(dist < 0)_8$ to be true in every failing test case.

This creates bias in the suspiciousness estimate for $(dist < 0)_7$ and $(dist < 0)_8$.

# Confounding Bias

```
1  int
2  distance(int x, int y)
3  {
4       int diff = x - y;
5       if (!( diff > 1)){ /* off by one */
6            int dist = 0;
7            dist = y - x;
8            print(dist);
9       }
10      int dist = 0;
11      dist = x - y;
12      return dist;
13 }
```

MaSTRI

| 6 | dist = 0 | 1 | 1 | 1 | 1 | 1 | 4 |
|---|---|---|---|---|---|---|---|

There is no difference between $(\text{dist} = 0)_6$ being true vs. Statement 6 being executed.

This creates bias in the suspiciousness estimate for $(\text{dist} = 0)_6$.

# Reducing Confounding Bias

MaSTRI

- Confounding Biases
  - ***Control for*** the most immediate cause of a statement being executed
  - ***Control for*** the immediate cause of a predicate being evaluated

# Observational Studies

MaSTRI

Create a regression model for each predicate:

- Treatment Variable (***T***)
  - ***T = 1*** if predicate is true in test case (Treatment)
  - ***T = 0*** if predicate is not true in test case (Control)
- Outcome Variable (***Y***)
  - ***Y = 1*** if test case fails
  - ***Y = 0*** if test case is successful (passes)

# Controlling for Failure Flow Bias

MaSTRI

$$Y = \alpha_p^{c,f} + \tau_p^{c,f} T_p + \beta_p^{c,f} C_p + \omega_p^{c,f} F_p + \varepsilon_p^{c,f}$$

$\tau_{ls,p}^{c,f}$ is the least squares suspiciousness estimate

# Matching Test Cases

MaSTRI

- Same pattern of covariates should exist in Treatment and Control Group
  - control flow predecessor true when predicate is not true?
  - predicate evaluated when predicate is not true?
- Test suite is given
  - Likely to be unbalanced

# Estimating Suspiciousness with Matching

MaSTRI

$$Y = \alpha_p + \tau_p T_p + \varepsilon_p$$

$\tau_{ls,p}$ is the least-squares suspiciousness estimate

# Causal Imputation

MaSTRI

**Algorithm 1** *Suspiciousness imputation* for predicates with a *complete lack of overlap* problem.

```
ImputeSusp(p)
1  matchedTestCases ← GetMatchedTestCases(p)
2  if (matchedTestCases = ∅)
3      then
4           ancestor ← GetCFP(p)
5           suspiciousness ← ImputeSusp(ancestor)
6
7  suspiciousness ← τ_{ls,p}
8  return suspiciousness
```

# MD Matching

MaSTRI

$$d_M(a,b) = \sqrt{(a - b)^T S^{-1} (a - b)}$$

# MD Matching

MaSTRI

1. For each test case $T_i$ in the treatment group

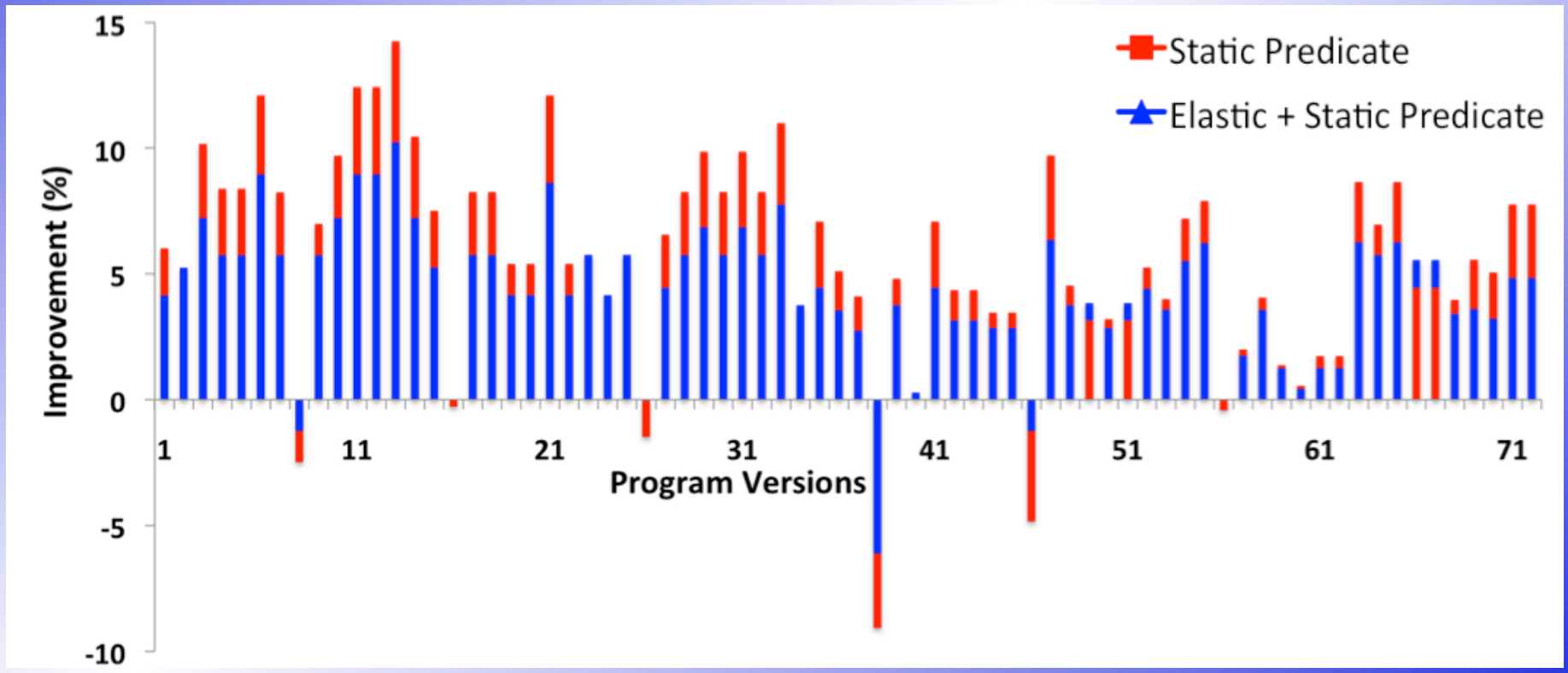
   a. find the test case $C_{min}$ in the control group that minimizes $d_M\left(T_i, C_j\right)$

2. Move $T_i$ and $C_{min}$ to the set of test cases to fit:

$$Y = \alpha_p + \tau_p T_p + \varepsilon_p$$

# Results

MaSTRI



# Conclusion

MaSTRI

- Balanced covariate values in test cases improve effectiveness
  - Especially for elastic predicates
- Large cost in terms of efficiency

# Questions / Comments

MaSTRI