SAPIENZA
UNIVERSITÀ DI ROMA

Combining Model Checking and Spectrum-Based Fault Localization with Multiple Counterexamples

Mohammed Bekkouche[1]
m.bekkouche@esi-sba.dz
Enrico Tronci[2]
tronci@di.uniroma1.it

[1]LabRI-SBA Laboratory, École Supérieure en Informatique, Sidi Bel Abbes, Algeria
[2]Computer Science Department, Sapienza University of Rome, Italy

ISNIB 2025

Jan 29-30th , 2025

Presentation Outline

Introduction

- Software Fault Localization: Identifying faults is costly; traditional methods like spectrum-based and model checking are promising but limited.
- Fault Localization Methods: Spectrum-based evaluates suspicion levels via tests, model checking identifies minimal faulty instructions, and our combined approach ranks errors but was limited to one counterexample.

Proposed Enhancement

Using multiple counterexamples in our combined approach improves fault localization precision by reducing non-faulty coverage and minimizing suspicious instructions.

- Experimental Results: TCAS benchmark tests confirm that using multiple counterexamples enhances fault localization accuracy.

State of the Art

Fault localization techniques can be categorized into slicing, spectrum-based, statistics-based, machine learning-based, data mining-based, information retrieval-based, model checking-based, and emerging methods.

- Spectrum-based fault localization
  SBFL ranks program elements based on suspicion levels using coverage data from test executions, known for its efficiency and simplicity
  Techniques : Tarantula, Ochiai, FLCN, ConsilientSFL, ...
- Fault localization in model checking
  Model checking verifies program adherence to specifications by exploring all possible states, generating counterexample traces for error localization
  Techniques : Explain, BugAssist, SNIPER, LocFaults, ...

Combining Model Checking and Testing for Fault Localization

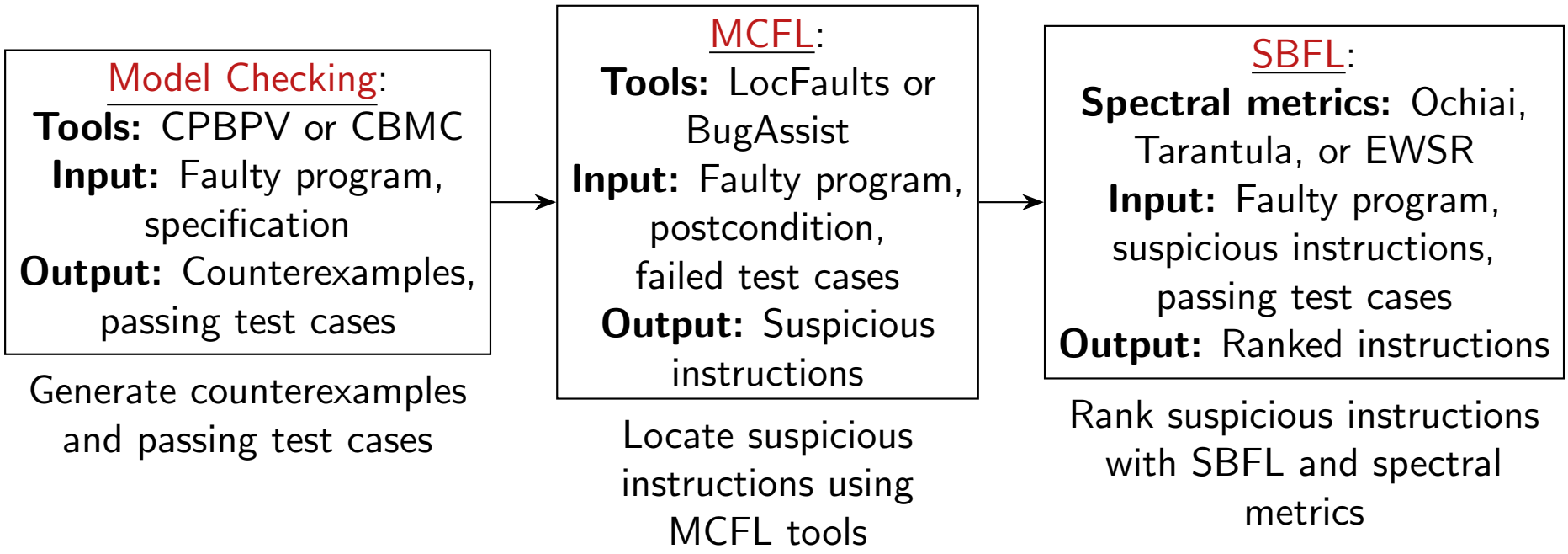

Figure: Integrated Approach: Combining Testing and Model Checking for Fault Localization.

Combining Model Checking and Testing for Fault Localization

Table: Spectral Formulas: Ochiai, Tarantula, WSR, and EWSR, with Their Respective Equations

Ochiai	$\frac{e_f}{\sqrt{(e_f + e_p) \cdot (e_f + n_f)}}$
Tarantula	$\frac{\frac{e_f}{e_f + n_f}}{\frac{e_p}{e_p + n_p} + \frac{e_f}{e_f + n_f}}$
WSR	$WSR = \frac{e_f \times n_p + n_f \times e_p}{e_f \times n_p + n_f \times e_p + e_f \times e_p + n_f \times n_p}$
EWSR	$EWSR = WSR \times \frac{1}{\sqrt{(e_f + e_p) \times (n_f + n_p)}}$

Explanatory Example

Listing: An example of a faulty program.

```
1   int fct2(int i, int j) {
2           int r;
3           if (i<10 && j<10) {
4                   r = i - j; // This instruction should be: r = i + j
5                   if (r >= 0)
6                       return 1;
7                   else
8                       return -1;
9           } else
10              return -2;
11  }
12  void main() {
13          int a, b, c;
14          __CPROVER_assume(a==3 && b==3 && c==5);
15          int res=0;
16          if (b<c){
17                  if (a<b)
18                      res=fct2(a,c);
19                  else if (a<c)
20                      res=fct2(b,c);
21          } else{
22                  if (a>b)
23                      res=fct2(a,c);
24                  else if (a>c)
25                      res=fct2(b,c);
26          }
27          if (res==0)
28              res=fct2(a,b);
29
30          assert(res==1);
31  }
```

Explanatory Example

Model Checking Step

- Generate 8 passing and 6 failing test cases for subsequent stages. Passing test cases produce expected outputs, while failing ones yield unexpected outputs. Details of test case generation are not the focus of present article.

MCFL Step

- Use BugAssist tool to localize faults based on counterexamples from failed test cases.
- BugAssist identifies suspicious instructions explaining non-conformance to the postcondition.
- Tool accepts ANSI-C program input, annotated with postconditions and optional preconditions in ACSL.
- Example program (Listing 1) is ready for BugAssist input, with a precondition (line 14) and postcondition (line 30).
- BugAssist provides a set of suspicious instructions for each counterexample.
- Repeat the process for other counterexamples and associate each with the corresponding suspicious instructions (e.g., $(a = 3, b = 3, c = 5)$ leads to instructions: $\{4, 5, 16, 17, 19, 27\}$).

Explanatory Example

SBFL Step

Table: Ranking of instructions in our example program based on the Model Checking tool BugAssist and the Ochiai spectral metric using multiple counterexamples.

Statement	(5, 3, 3)	(5, 5, 5)	(4, 1, 4)	(4, 5, 3)	(5, 2, 1)	(10,11,13)	(11,10,13)	(22,19,21)	(3, 3, 5)	(4, 7, 5)	(4, 3, 5)	(3, 1,-5)	(3,-5, 2)	(2, 3,-5)	e_f	e_p	n_f	n_p	Ochiai	Rank
3	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	8	5	0	0.14	10
4	1	1	1	1	1	0	0	0	1	1	1	1	1	1	6	5	0	3	0.74	1
5	1	1	1	1	1	0	0	0	1	1	1	1	1	1	6	5	0	3	0.74	1
6	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	4	6	4	0	11
8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	6	7	0	11
10	0	0	0	0	0	1	1	1	0	0	0	1	1	1	3	3	3	5	0.5	5
15	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	8	6	0	0	10
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	6	8	0	0	0.65	3
17	0	0	1	0	0	1	1	1	1	0	1	0	1	0	3	4	3	4	0.46	7
18	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	6	7	0	11
19	0	0	1	0	0	0	1	1	1	0	1	0	1	0	3	3	3	5	0.5	5
20	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	6	7	0	11
22	1	1	0	1	1	0	0	0	0	1	0	1	0	1	3	4	3	4	0.46	7
23	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	6	6	0	11
24	0	1	0	1	0	0	0	0	0	1	0	0	0	1	2	2	4	6	0.41	9
25	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	6	7	0	11
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	6	8	0	0	0.65	3
28	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	3	6	5	0	11
Result	0	0	0	0	0	0	0	0	1	1	1	1	1	1						

Explanatory Example

Original Contribution

- This study demonstrates the superiority of using multiple counterexamples over a single one in fault localization approach.
- The following table reveals that relying on a single counterexample results in inconsistent rankings of the faulty instruction (line 4)
  which appears 3rd in four cases and 4th in two cases, emphasizing the need for a comprehensive multi-counterexample approach

Table: Ranking of instructions in our example program based on the Model Checking tool BugAssist and the Ochiai spectral metric using a unique counterexample.

Statement / Counterexample	3	4	5	6	8	10	15	16	17	18	19	20	22	23	24	25	27	28
(3, 3, 5)	7	3	3	7	7	7	7	5	2	7	1	7	7	7	7	7	5	7
(4, 7, 5)	7	3	3	7	7	7	7	5	7	7	7	7	2	7	1	7	5	7
(4, 3, 5)	7	3	3	7	7	7	7	5	2	7	1	7	7	7	7	7	5	7
(3, 1,-5)	7	3	3	7	7	1	7	5	7	7	7	7	2	7	7	7	5	7
(3,-5, 2)	6	4	4	9	9	1	9	6	3	9	1	9	9	9	9	9	6	9
(2, 3,-5)	8	4	4	8	8	2	8	6	8	8	8	8	3	8	1	8	6	8

Results

Benchmark used:

- The approach was tested on Siemens' TCAS benchmark suite, simulating an aircraft collision avoidance system with 173 lines of code across 41 versions (excluding v33 and v38 due to out-of-scope errors) and 1068 test cases.

Implementation of MCFL step:

- BugAssist is used to identify suspicious instructions in faulty TCAS program versions, leveraging multiple counterexamples to enhance fault localization.
  $\longrightarrow$ This improves our previous work that utilized only a single counterexample per version.
- Counterexamples are determined by comparing outputs of the erroneous and correct program versions across 1068 test cases.

Results

Implementation of SBFL step:

- Instructions for each erroneous version are classified using suspicious instructions from counterexamples, passing test cases, and the Ochiai spectral metric.
- Suspicion degrees are assigned based on coverage.
- Faulty instructions are ranked by suspicion degree, with tie elements resolved using the MID formula:

$$\text{MID} = S + \left(\frac{E - 1}{2} \right) \tag{1}$$

where S is the starting position, and E is the number of tied instructions.

  - The program spectrum is calculated using both failed and passing test cases.
  - For failed cases, covered code lines belong to the set of suspicious instructions.
  - Ochiai spectral metric prioritize faulty code lines.

Comparison:

- Results are compared with testing-only approaches using Ochiai, Tarantula, and EWSR spectral formulas.

Results

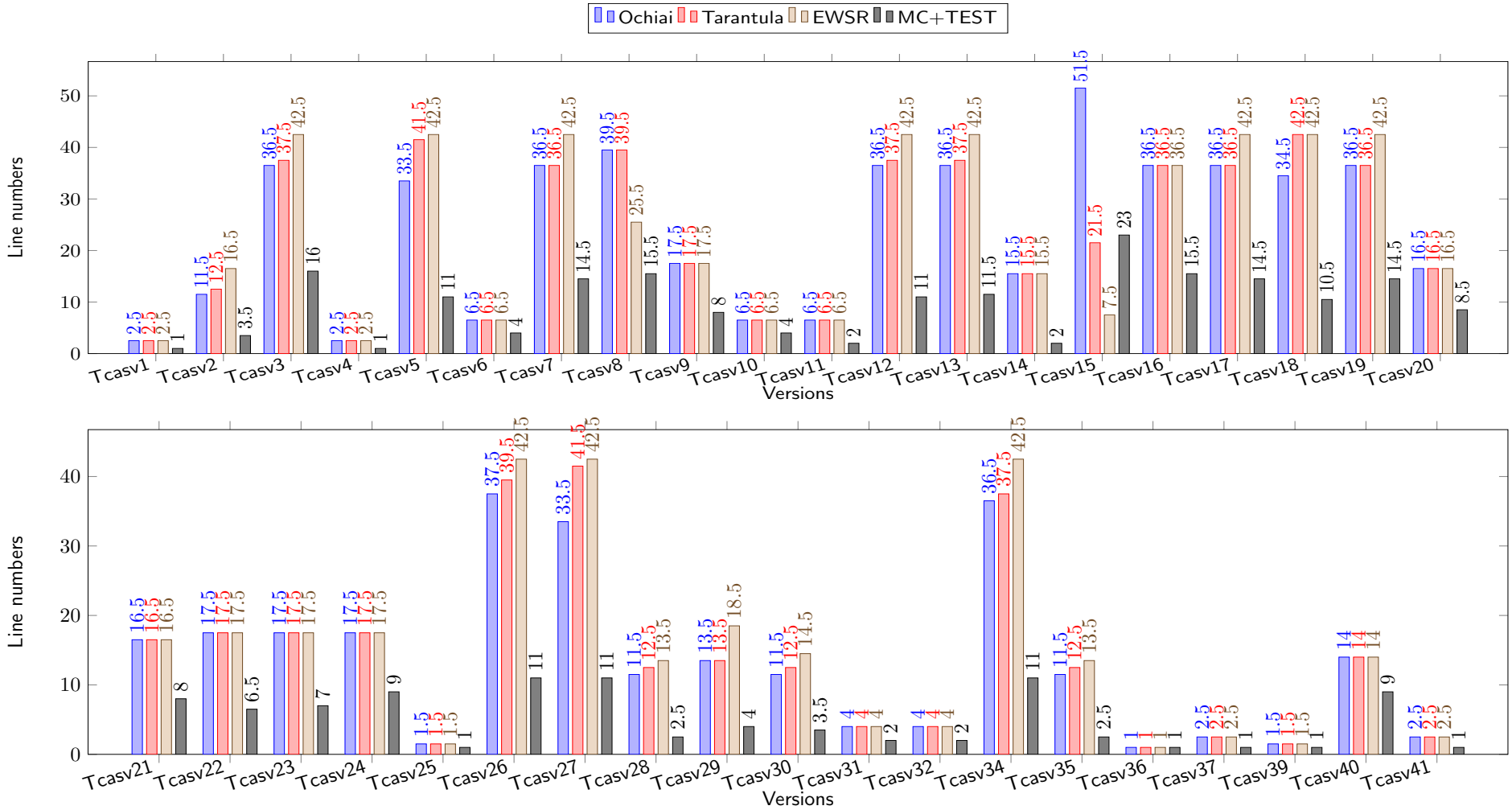

Figure: The Average Position of the Faulty Instruction in Each Program Version of the TCAS Suite from Siemens using Multiple Counterexamples

Results

Visualization of Results:

- This previous figure presents the TCAS benchmark results using multiple counterexamples.
- Each cluster of bars corresponds to a version:
  - Bars 1-3: Average faulty instruction positions for Ochiai, Tarantula, and EWSR spectral formulas.
  - Bar 4: Results from the combined approach with the Ochiai formula.

Insights from the Data:

- Ochiai consistently outperforms Tarantula and EWSR, delivering higher accuracy in fault localization across most versions.
- The integrated approach, combining model checking and testing, consistently outperforms pure spectrum-based methods, especially in versions with high average positions.

Results

- The previous figure presents the position of the faulty instruction for each erroneous version in TCAS in the ranking returned by our integrated approach using multiple counterexamples.
- To assess the significance of this improvement compared to what was presented in the CSA 2024 paper (we used only one counterexample for each version) :
  - We calculated the position of the erroneous instruction in the ranking obtained by our approach but using only one counterexample at a time
  - Then, we calculated the average positions obtained for all counterexamples of each TCAS version
    $\rightarrow$ see bar 2 in each bar cluster in Figure of slide 16 and the blue points in Figure of slide 17
  - We compared this result with our approach considering all counterexamples for each erroneous TCAS version at once
    $\rightarrow$ see bar 1 in each bar cluster in Figure of slide 16 and the red points in Figure of slide 17
  - Figure of slide 17 presents the standard deviation for each erroneous version, measuring the dispersion of faulty instruction positions around the average in our approach using one counterexample at a time.

Results

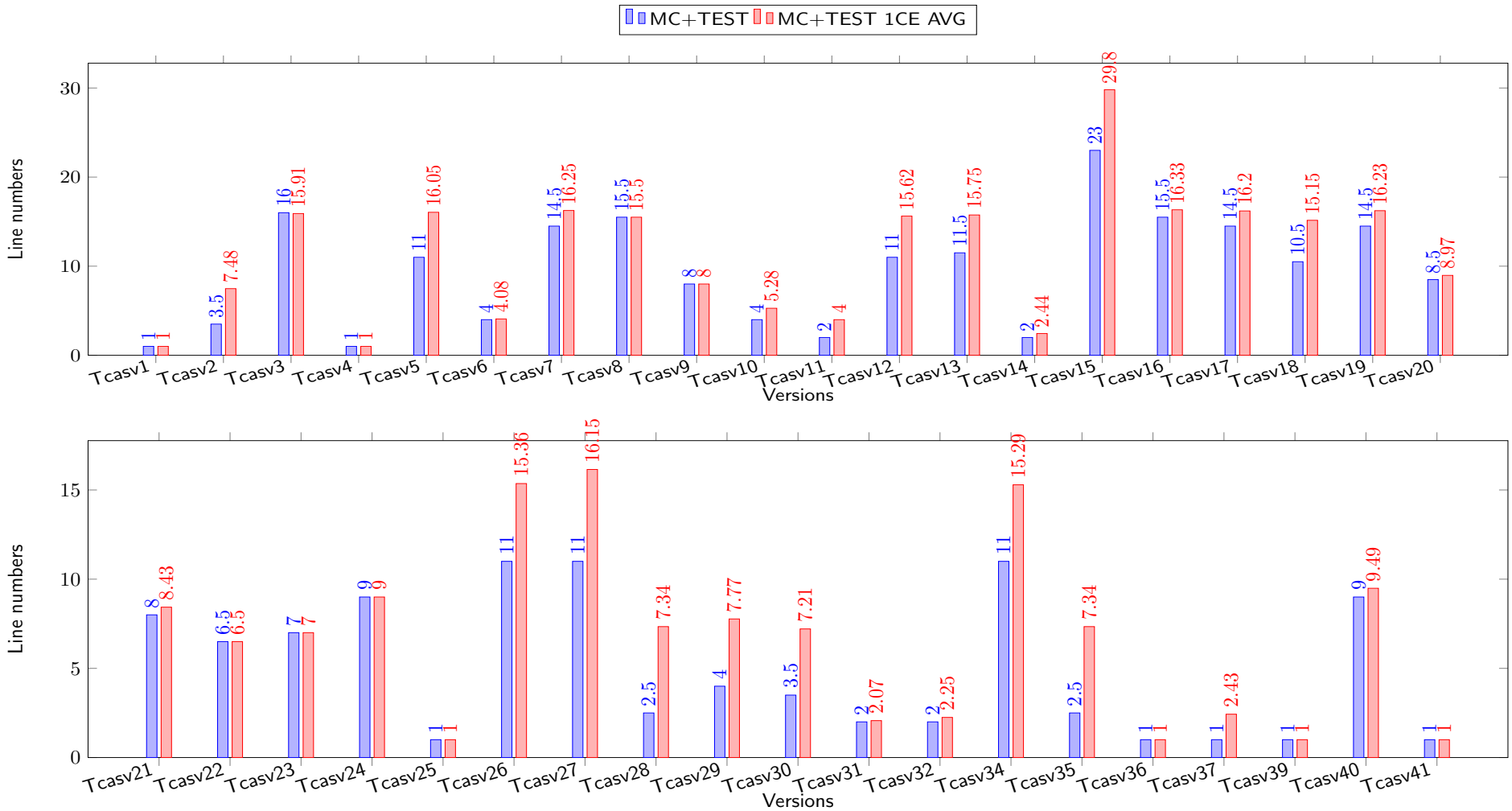

Figure: Average Position of Faulty Instructions in Each TCAS Suite Program Version by Siemens: Utilizing Multiple Counterexamples and Calculating the Average Position Across Each Counterexample

Results

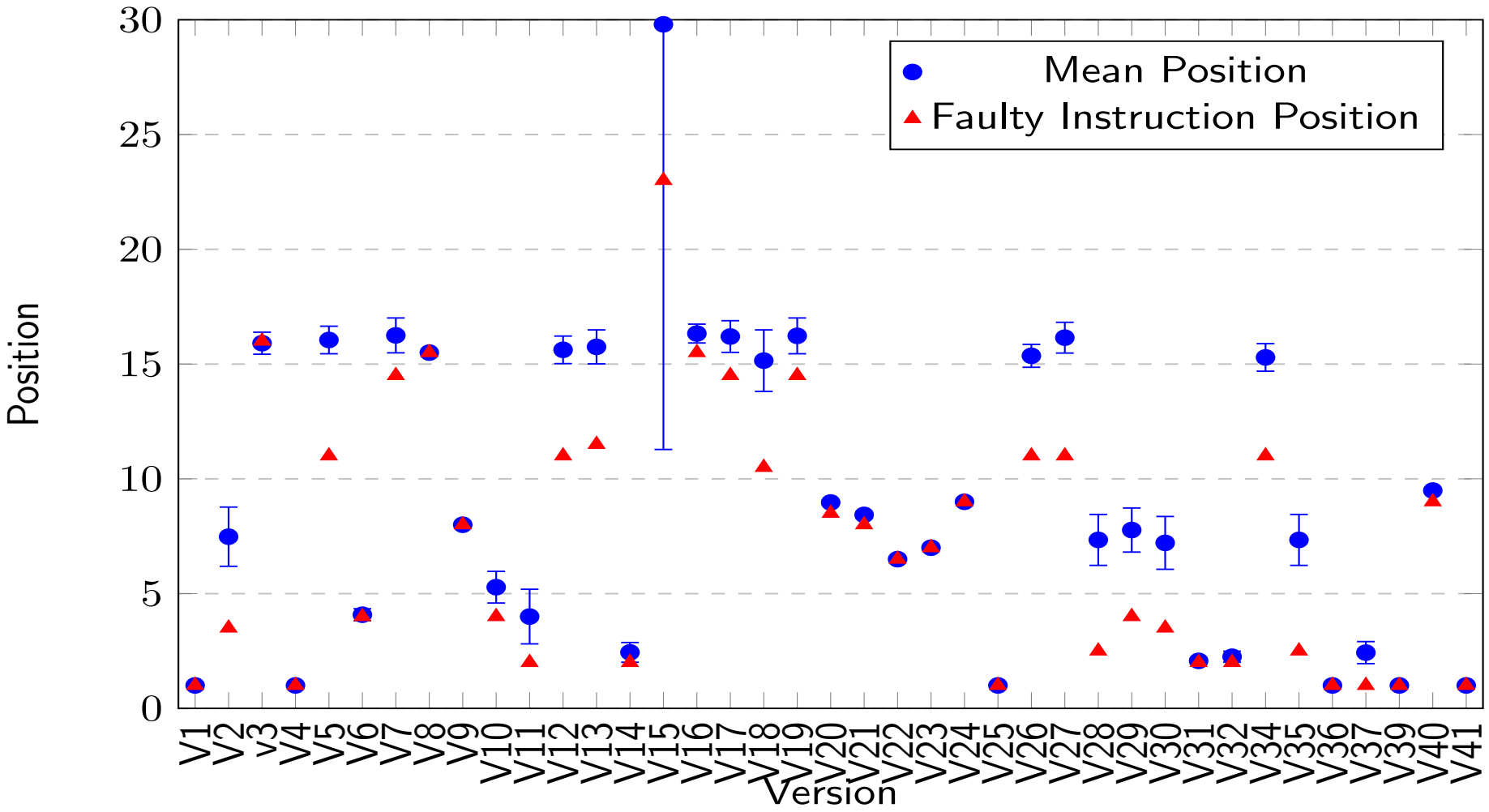

Figure: Plot of Positions with Error Bars (Standard Deviation)

Conclusion

- Study Focus: Evaluation of our integrated fault localization approach combining spectrum-based techniques with model checking.
- Main Results:
  $\longrightarrow$ Significant improvement in fault localization accuracy with multiple counterexamples compared to traditional methods.
  $\longrightarrow$ $\underline{62.80\%}$ average improvement in accuracy across TCAS versions.
  $\longrightarrow$ $\underline{20.39\%}$ increase in accuracy using multiple counterexamples over a single one.

- Conclusion:
  Multiple counterexamples enhance fault localization in our approach, reducing the number of instructions to analyze and improving precision.
- Future Directions:
  $\longrightarrow$ Further refinement of the approach.
  $\longrightarrow$ Expansion to various software systems.
  $\longrightarrow$ Integration into automated fault localization tools for improved software development.

Thank you for your attention.

Questions ?