[simula . research laboratory]

Applications of Machine Learning in Software Testing

Lionel C. Briand

Simula Research Laboratory

and

University of Oslo

Acknowledgments

- Yvan Iabiche
- Xutao Liu
- Zaheer Bawar
- Kambiz Frounchi

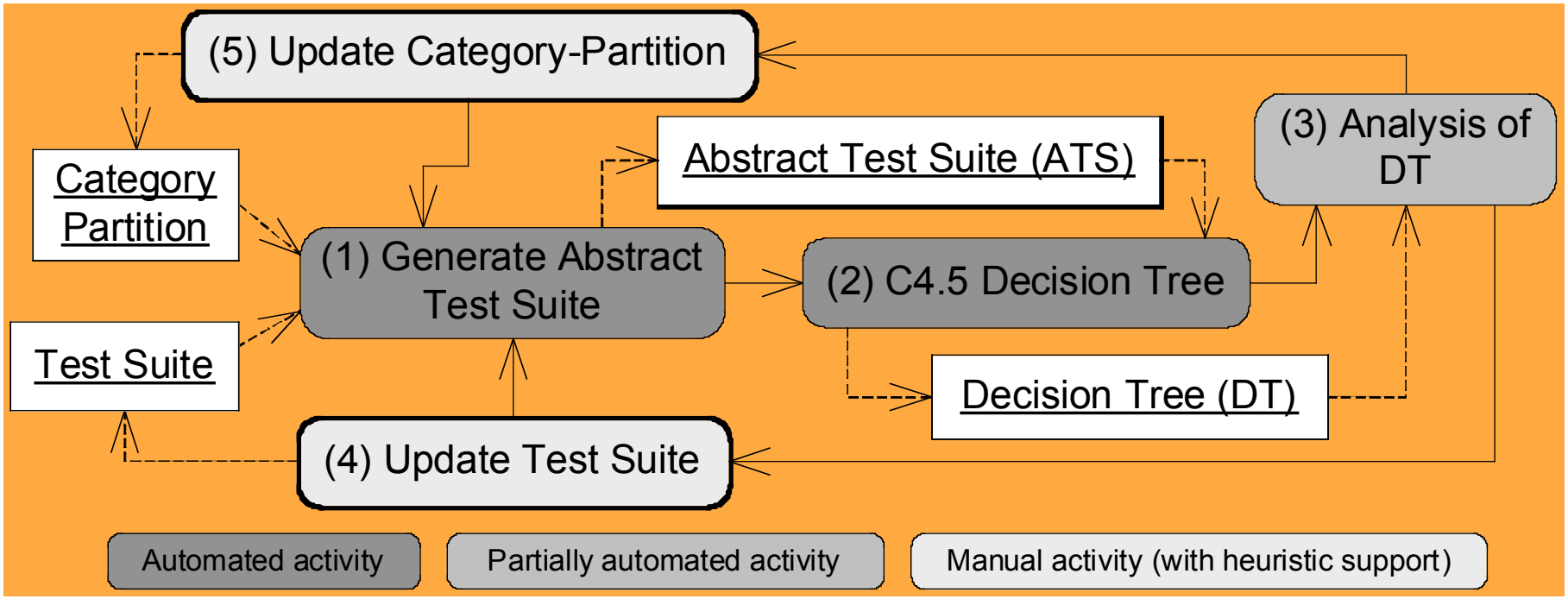
Motivations

- There are many examples of ML applications in the testing literature, but not always where it could be the most useful or practical
- Limited usage of ML in commercial testing tools and practice
- Application of ML in testing has not reached its full potential
- Examples: Applications of machine learning for supporting test specifications, test oracles, and debugging
- General conclusions from these experiences

Black-box Test Specifications

- Context: Black-box, specification testing
- Black-box, specification testing is the most common practice for large components, subsystems, and systems. But it is error-prone.
- Learning objective: relationships between inputs & execution conditions and outputs
- Usage: detect anomalies in black-box test specifications, iterative improvement
- User's role: define/refine categories and choices (Category-partition)
- Just learning from traces is unlikely to be practical in many situations: Exploit test specifications

Iterative Improvement Process

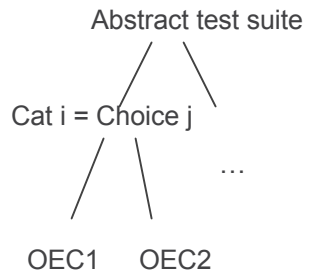


Abstract Test Cases

- Using Category and choices to derive abstract test cases
  - Categories (e.g., triangle side s1 = s2), choices (e.g., true/false)
  - CP definitions must be sufficiently precise
  - (1,2,2) => (s1 <> s2, s2 = s3, s1<>s3)
  - Output equivalence class: Isosceles, etc.
  - Abstract test cases make important properties of test cases explicit
  - Facilitate learning

Examples with Triangle Program

```
1       (a vs. b) = a!=b
2       |   (c vs. a+b) = c<=a+b
3       |   |   (a vs. b+c) = a<=b+c
4       |   |   |   (b vs. a+c) = b<=a+c
5       |   |   |   |   (b vs. c) = b=c
6       |   |   |   |   |   (a) = a>0: Isosceles (22.0)
```



Example 1:

```
(a vs. b) = a=b
|   (b vs. c) = b!=c
|   |   (a vs. b+c) = a<=b+c
|   |   |   (c vs. a+b) = c<=a+b: Isosceles (24.0/2.0)
```

Example 2:

```
(a vs. b) = a=b
|   (c) = c>0
|   |   (b vs. c) = b!=c: Isosceles (24.0/2.0)
```

Examples of Detected Problems: Misclassifications

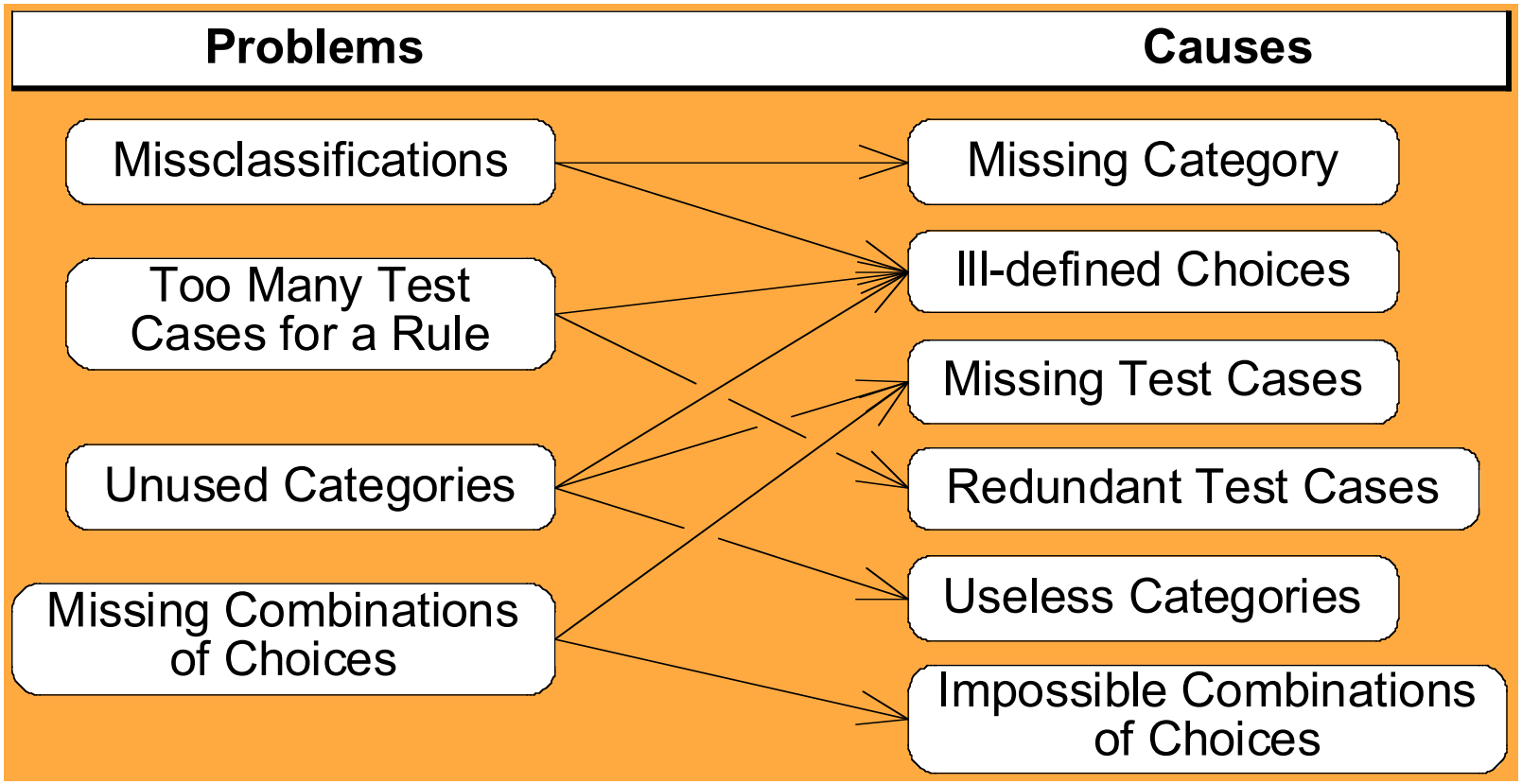
Example: ill-defined Choices

- Ill-defined choices make render a category a poor predictor of output equivalence classes
- Example: Category (c vs. a+b)

```
c < a+ b   (should be <=)
c >= a +b  (should be >)
```

- Misclassifications where c = a+b

Linking Problems to Potential Causes



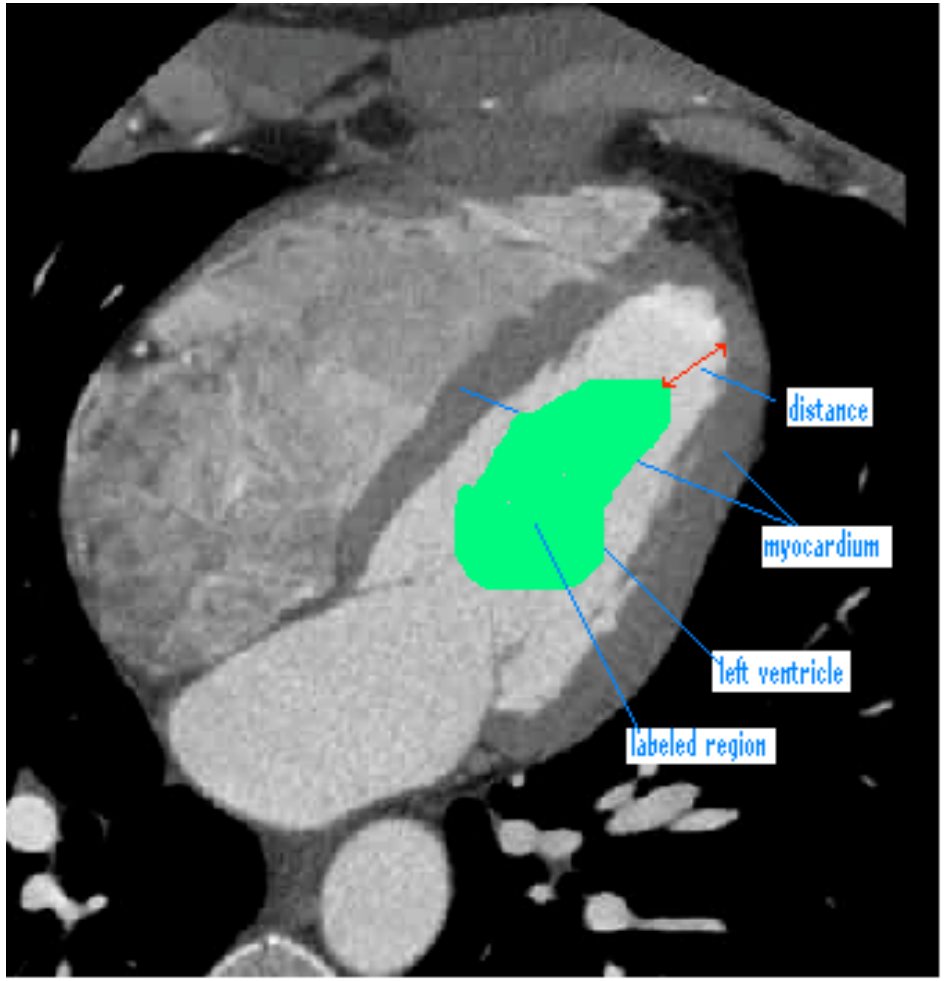
Case Study: Summary of Results

- Experiments with students defining and refining test case specifications using category partition
- Taxonomies of decision tree problems and causes complete
- Student achieved a good CP specification in two or three iterations
- Reasonable increase in test cases led to a significant number of additional faults.
- Our heuristic to remove redundant test cases leads to significant reduction in test suite size (~50%), but a small reduction in the number of faults detected may also be observed.

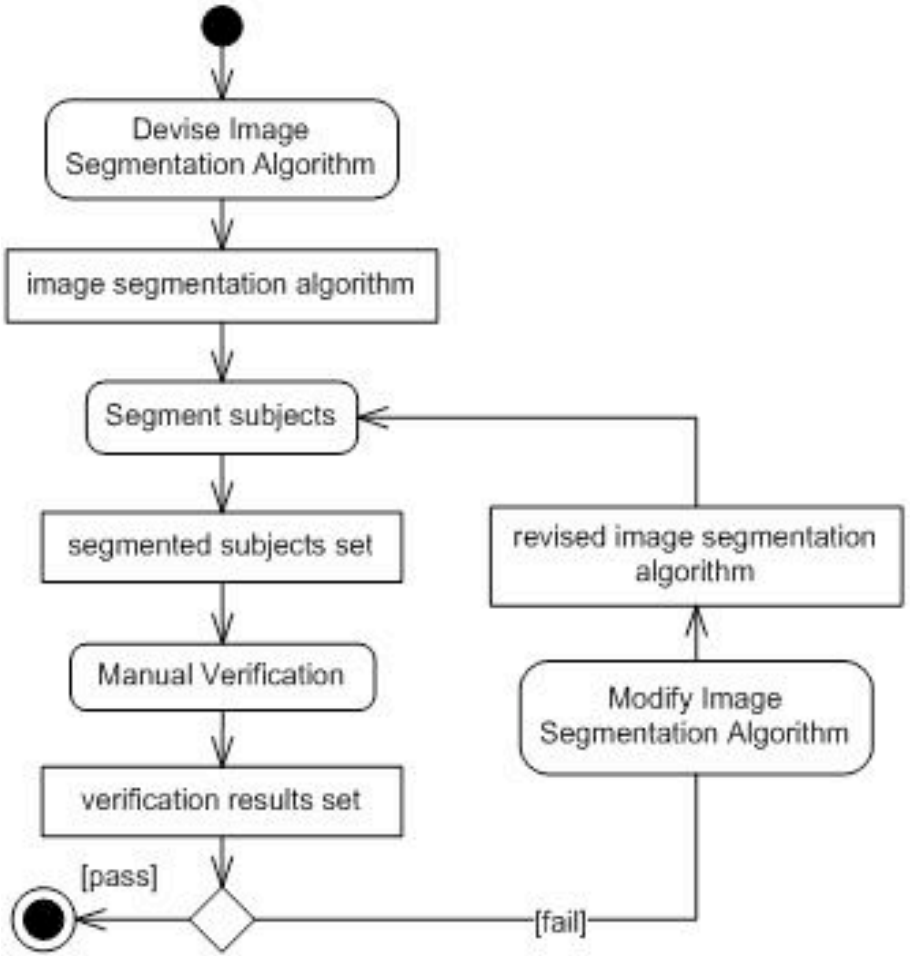
Test Oracles

- Context: Iterative development and testing, no precise test oracles
- Learning objectives: Model expert knowledge in terms of output correctness and similarity
- Usage: avoid expensive (automate) re-testing of previously successful test cases (segmentations)
- User's role: Expert must help devise a training set to feed the ML algorithm.
- Example is image segmentation algorithms for heart ventricles

Heart Ventricle Segmentation



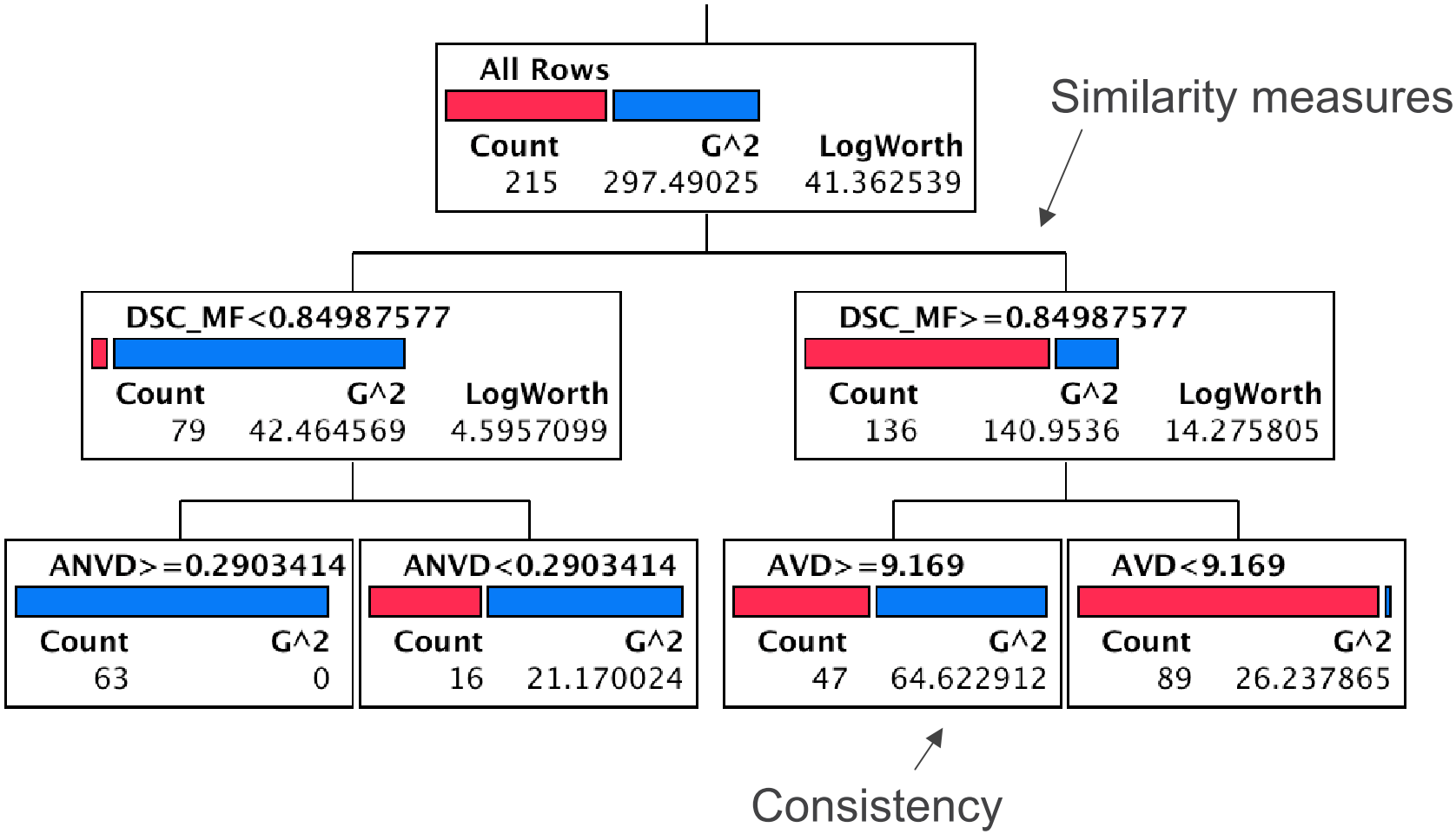
Iterative Development of Segmentation Algorithms



Study

- Many (imperfect) similarity measures between segmentations in the literature
- Oracle: Are two segmentations of the same image similar enough to be confidently considered equivalent or consistent?
  - Vi Correct & Vi+1 consistent => Vi+1 correct
  - Vi Correct & Vi+1 inconsistent => Vi+1 incorrect
  - Vi Incorrect & Vi+1 consistent => Vi+1 incorrect
- Machine learning uses training set of instances where that question was answered by experts + similarity measures

Classification Tree Predicting Consistency of Segmentations



Results

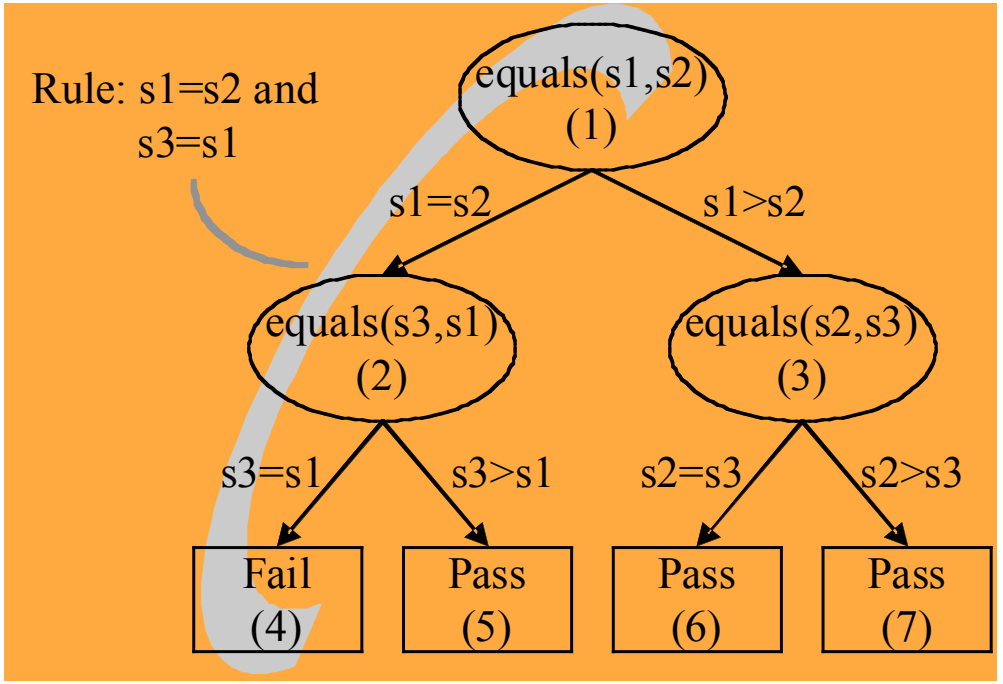
- Three similarity measures selected
- Cross-validation ROC area: 94%
- For roughly 75% of comparisons, the decision tree can be trusted with a high level of confidence
- For 25% of comparisons, the expert will probably have to perform manual checks
- More similarity measures to consider
- Similar results with other rule generation algorithms (PART, Ripper)

Fault Localization (Debugging)

- Context: Black-box, specification testing
- Learning objective: relationships between inputs & execution conditions and failure occurrences
- Usage: Learn about failure conditions, refine statement ranking techniques in the presence of multiple faults
- User's role: define categories and choices (Category-partition)
- Techniques ranking statements are unlikely to be of sufficient help for debugging
- Still need to address the case of multiple faults (failures caused by different faults)
- Failure conditions must be characterized in an easily understood form

Generating Rules - Test case classification

- Using C4.5 to analyze abstract test cases
  - A failing rule generated by the C4.5 models a possible condition of failure
  - Failing test cases associated with a same C4.5 rule (similar conditions) are likely to fail due to the same faults



Accuracy of Fail Rules (Space)

		Predicted Fail	Predicted Pass
Actual	Fail	6045	335
	Pass	550	6655

- Fail test cases:

92% precision, 95% recall

- Similar for Pass test cases

Rules	length	TC	Fail Probability
F109	1	30	100.00%
F025	5	667	99.55%
F104	2	240	98.75%
F106	1	304	98.36%
F105	2	158	97.47%
F107	1	114	97.37%
F009	12	147	97.28%
F010	12	132	96.97%
F003	12	37	94.59%
F004	11	200	94.50%
F012	11	97	93.81%
F005	11	548	93.80%
F070	10	271	93.73%
F069	9	116	92.24%
F087	9	199	91.96%
F016	7	12	91.67%
F086	8	700	91.57%

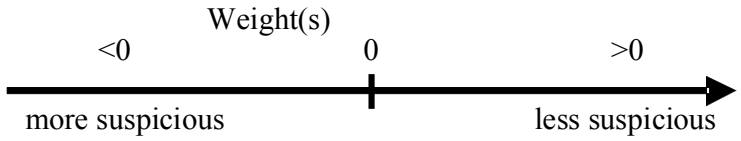
Table 1 Example Fail rules with Fail probabilities and test case coverage

1. defines a triangular grid of antennas (condition 1),
2. defines a uniform amplitude and phase of the antennas (conditions 2 and 3),
3. defines the triangular grid with angle coordinates or Cartesian coordinates, and a value is missing when providing the coordinates (conditions 4 and 5);

Statement ranking strategy

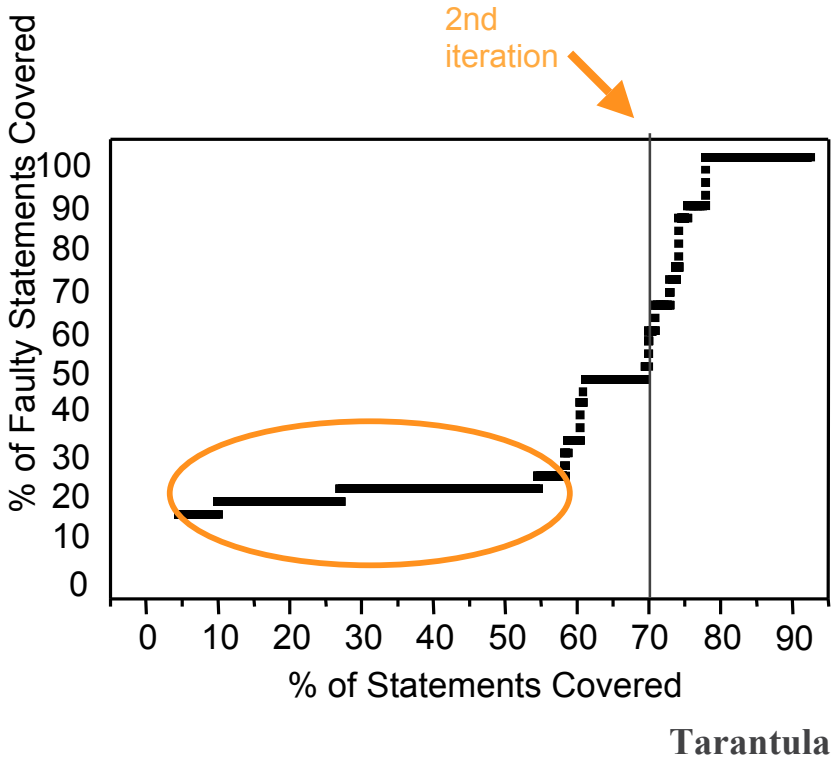
- Select high accuracy rules based on a sufficiently large number of (abstract) test cases
- Consider test cases in each rule separately
- In each test case set matching a failing rule, the more test cases executing a statement, the more suspicious it is, and the smaller its weight: $Weight(R_i,s) \in [-1\ 0]$
- For passing rules, the more test cases executing a statement, the safer it is: $Weight(R_i,s) \in [0\ 1]$

$$Weight(s) = \sum_{R_i \in R} Weight(R_i, s)$$

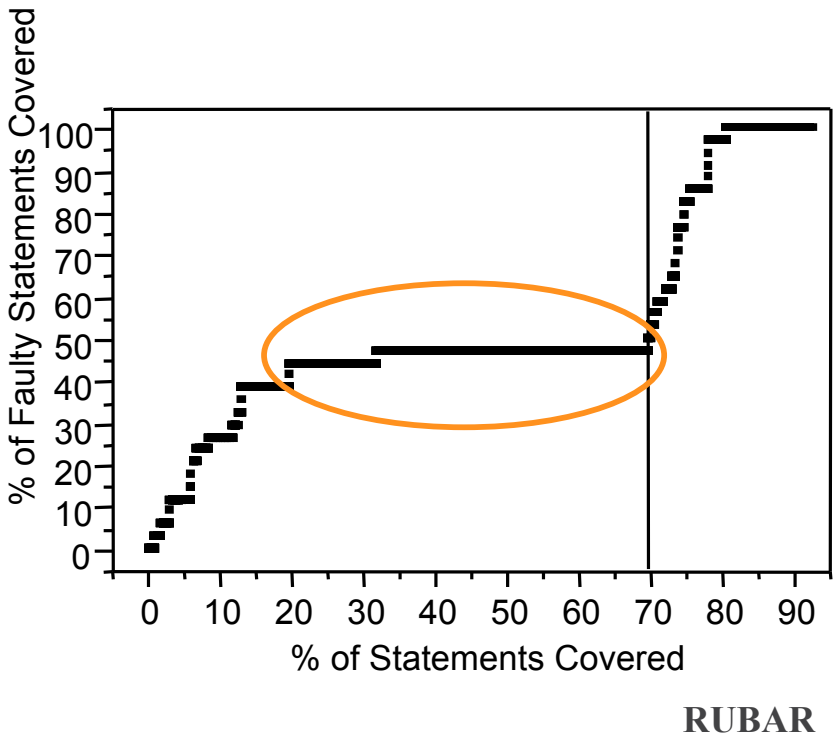


Statement Ranking: Space

- Scenario: for each iteration, fix all the faults in reachable statements



Tarantula



RUBAR

Case studies: summary

- RUBAR more effective than Tarantula at ranking faulty statements thanks to the C4.5 classification rules
- The generated C4.5 classification rules based on CP choices characterizing failure conditions accurately predict failures
- Experiments with human debuggers are needed to assess the cost-effectiveness of the approach

Lessons Learned

- In all considered applications, it is difficult to imagine how the problem could have been solved without human input, e.g., categories and choices
- Machine learning has shown to help decision making -- but it does not help fully automate solutions to the test specification, oracle, and fault localization problems.
- Search for full automation is often counter-productive: It leads to impractical solutions.
- Important question: What is best handled/decided by the expert and what is best automated (through ML algorithms)
- Solutions that best combine human expertise and automated support

References

- L.C. Briand, Y. Labiche, X. Liu, "Using Machine Learning to Support Debugging with Tarantula", IEEE International Symposium on Software Reliability Engineering (ISSRE 2007), Sweden
- L.C. Briand, Y. Labiche, Z. Bawar, "Using Machine Learning to Refine Black-box Test Specifications and Test Suites", Technical Report SCE-07-05, Carleton University, May 2007
- K. Frounchi, L. Briand, Y. Labiche, “Learning a Test Oracle Towards Automating Image Segmentation Evaluation”, Technical Report SCE-08-02, Carleton University, March 2008

? Questions ?

RUBAR iterative debugging process