Learning Effective Oracle Comparator Combinations for Web Applications

Sara Sprenkle

WASHINGTON AND LEE
UNIVERSITY

Emily Hill

Lori Pollock

Testing Web Applications

- 772 million people online in May 2007*
- Amazon.com had 142K *unique* visitors, on average visit 3.1 times, in May 2007*
  - Earned $10.7 billion in revenue in 2006
- Need reliable web applications
  - Errors are seen by many, quickly
  - Frequent maintenance/update cycles
  - Need fast, effective oracles

* Source: comScore

Why Oracle Effectiveness Matters

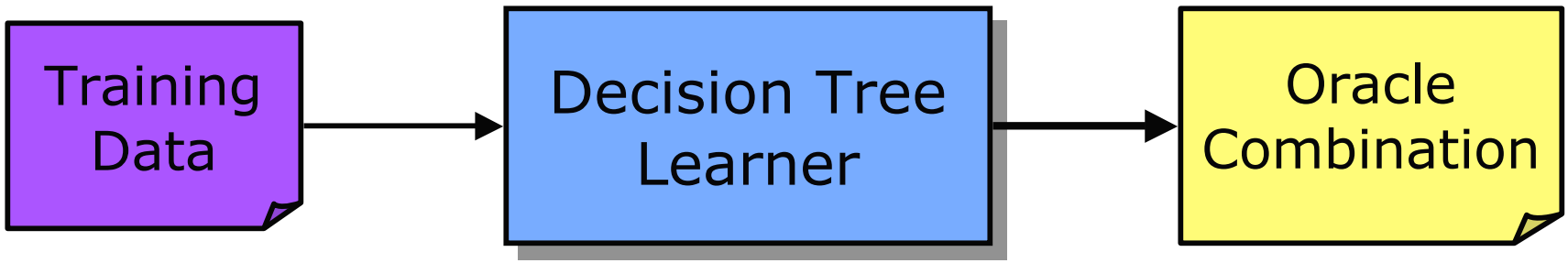
- Need oracle to be accurate
  - Allow only the applications that behave as expected to be deployed
- False negatives (say OK when faulty)
  - Loss of revenues, consumer confidence
- False positives (say faulty when not)
  - Overwhelm testers, waste developers' time
- Testers need to consider tradeoffs between false negatives, false positives
  - May use >1 oracle to improve effectiveness

Oracle Selection Problem

- Given a suite of oracle comparators, which oracle comparator or combination should you select for your web application?
  - Most effective oracle combination
  - Select cheaply, systematically

Overview of Our Approach

- Decision tree learning to determine systematically best oracle comparator
  - Intuitive, easily interpreted model
- Train on oracles' failure detection results for seeded faults, application behavior



Overview of Contributions

- Learned effective oracle combinations for 4 representative web applications using decision tree learning
- Evaluated learned oracles on non-training set data
- Proposed a methodology for selecting an effective oracle comparator combination

Background: Oracle Comparators

- Suite of 22 general, automated HTML oracle comparators [ISSRE07]
  - HTML: common output format
  - What user sees
  - Failures from other output may show up in HTML

Example of an HTML Response

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>hiperspace lab</title>
<style>a:hover{color:#952C2C; text-decoration:none}… </style>
<script language="Javascript"> … </script>
</head>
<body>
<table border=0 cellspacing=0 width="580">
<tr> <td rowspace=2>
<img src="books.gif" width=310></td></tr>
…</table>
<!-- Sidebar Links -->
<ul>…
<li><a href="alumni.html">Alumni</a>
</ul>
</body>
</html>
```

Start tag

Content

Style

Layout

Comment

Content

Attribute

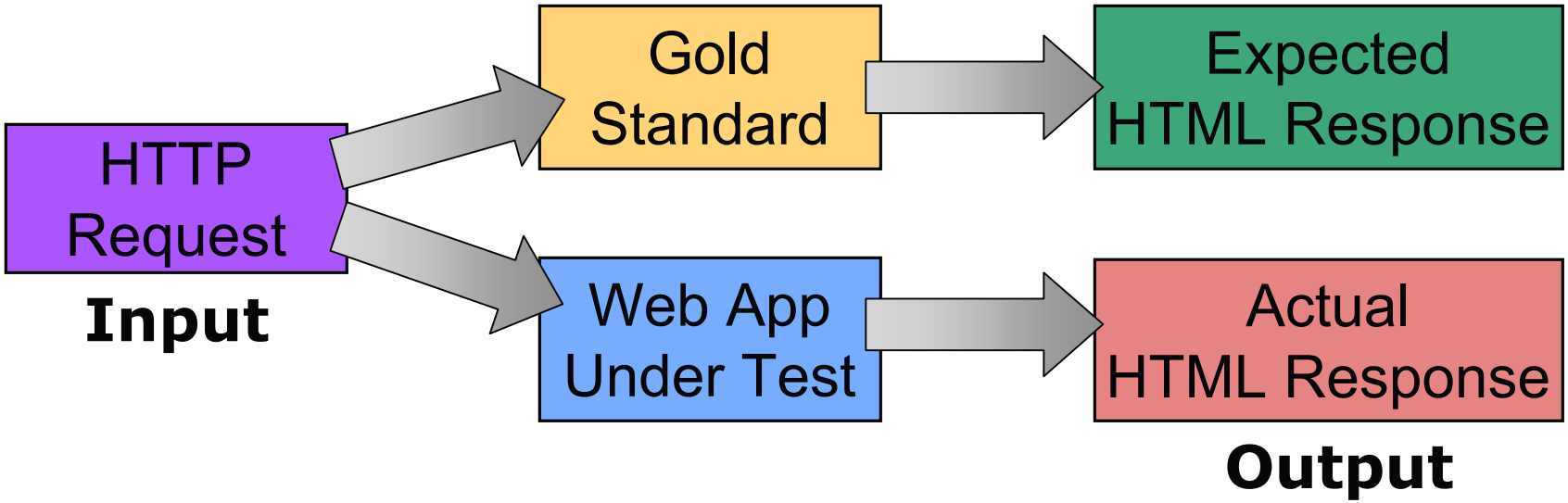
Close tag

- Browser collapses white space

HTML Oracle Challenges

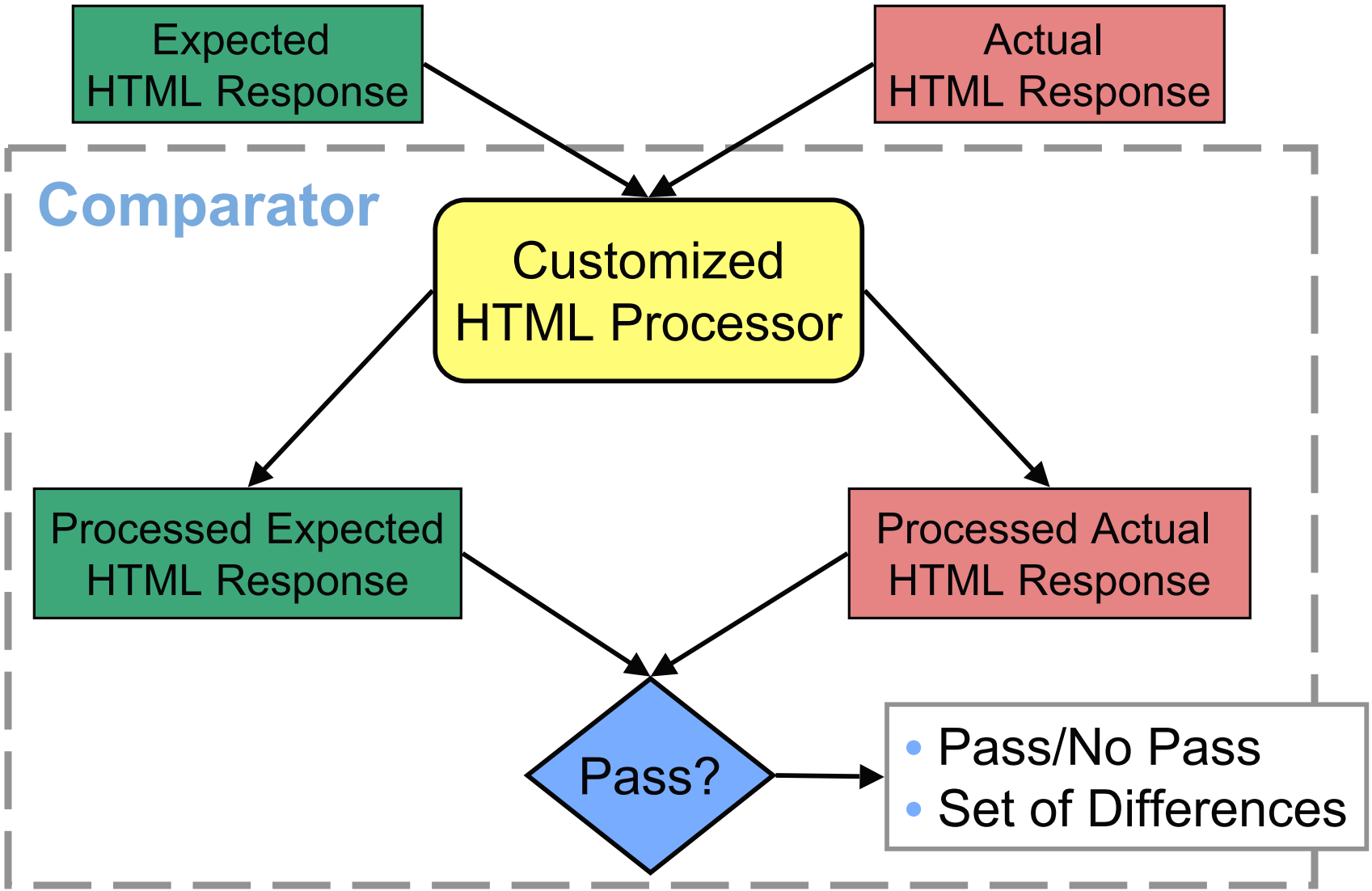
- Many different HTML components
- Differences in various components may or may not matter
  - Which matter depends on the application
- Created 22 customized oracles to address these differences
  - Tradeoffs between oracles' false positives and false negatives

Background: Our Oracle Process



- Expected results: use original version of application (assumed to be correct)
  - Gold Standard

Background: Our Oracle Process



Partial Ordering of Implemented Comparison Algorithms

Document-based oracles:

Document (**D**)
DocumentBase (**DB**)
DocumentBase-CollapsedWS (**DB-W**)

Content-based oracles:

Content (**C**)
Content-CollapsedWS (**C-W**)
Content-CollapsedWS-Dates (**C-WD**)

Structure-based classes:

Tags (**T**)
- Forms (**N+F**)
- TagNames+Impt Attrs (**N+I**)
  - TagNames (**N**)
  - UnorderedLinks (**N+U**)

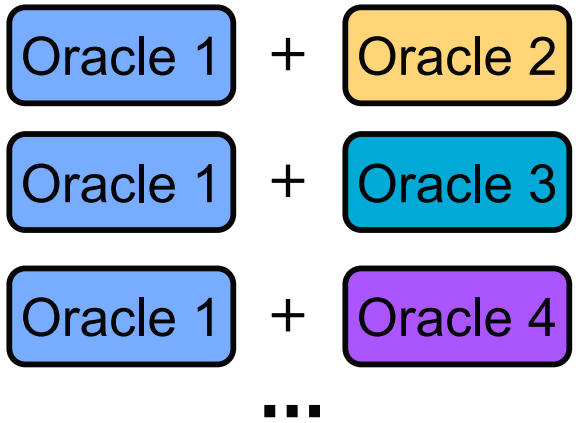
More information

Fewer False Negatives

Fewer False Positives

ISSRE07 Combinations Summary

- Investigated if unioning or intersecting oracle comparators → a more effective oracle
  - *Unioning* carefully selected comparators: better effectiveness than individual comparators
- Across all apps, N+I ∪ C-WD was best
- ➔ Did not exhaustively combine oracles or try different operations to combine

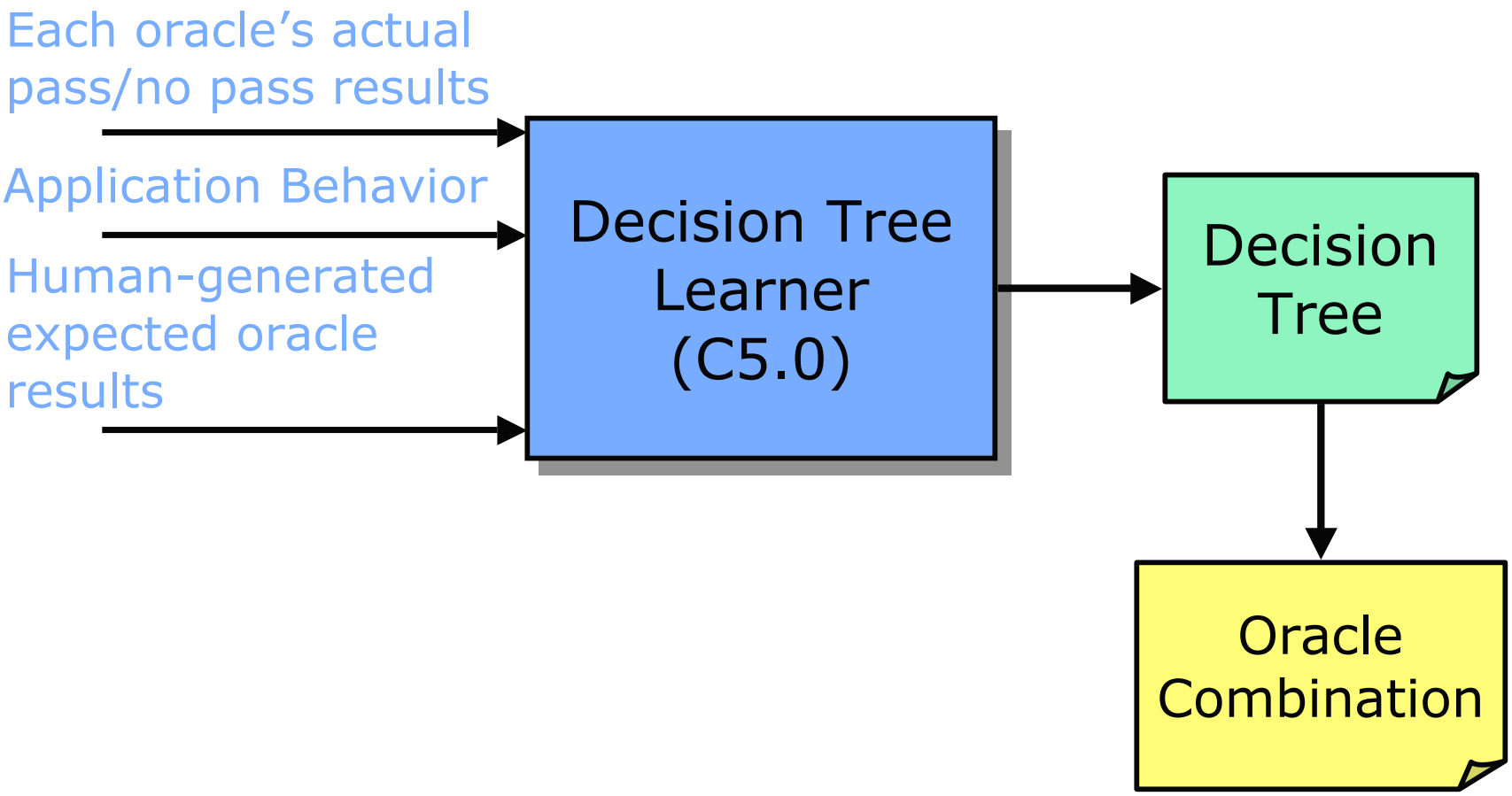


Selecting an Effective Oracle

- Given the tradeoffs between all the classes and combinations, need systematic technique to select effective oracle
- Decision Tree Learning
  - Inductive learning method
  - Constructs a classifier/decision tree for a training data set
- Our classification problem:
  - Given: a feature set containing oracle results, application behavior
  - Classify an HTTP request/HTML response as pass/no pass

Decision Tree Learning

Training Data



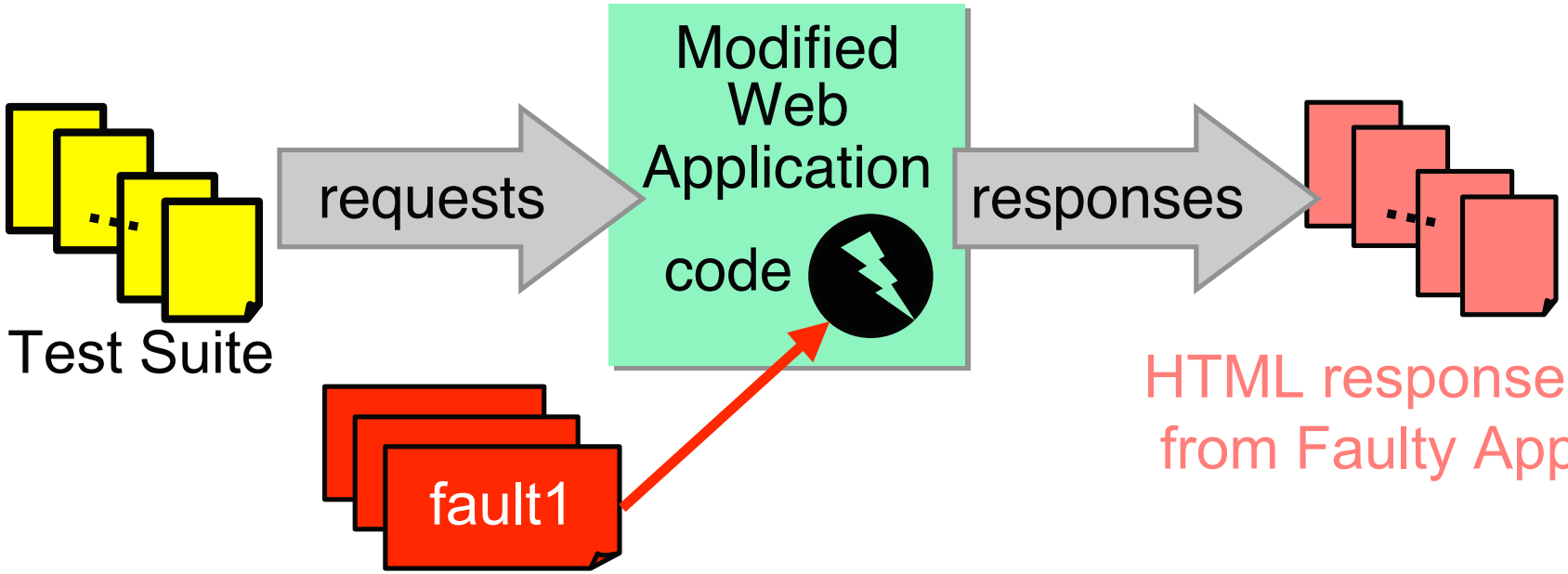
Example of Training Data

Deterministic/ Nondeterministic

Pass/No Pass

Request/ Response	App. Behavior	Response Behavior	Oracle 1	…	Oracle n	Expected Oracle
0	D	D	P		P	P
1	D	D	P		N	N
2	N	N	P		P	P
3	N	D	P		P	N
…						
x	N	D	N		P	P

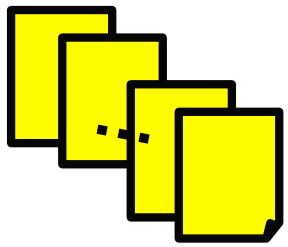
Methodology: Generating Training Data



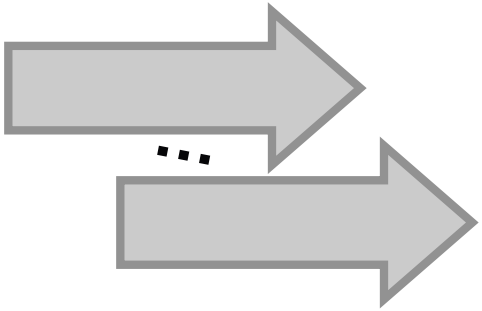
- Faults seeded manually into application code
- Various *categories* of seeded faults
  - data, logic, form, appearance, link

Methodology: Generating Training Data

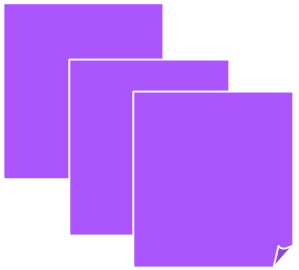
For the test suite of each subject application:



Test Suite



Replay suite on clean and fault-seeded version(s) of code



Generate failure detection reports using each **oracle**

- Compare the HTML document from faulty version with HTML document from clean version
- Manually determine if document exposes fault

Subject Applications

- Four deployed subject applications

Application	Test Cases	Requests	NCLOC*	Faults Exposed
Masplas	169	1103	999	22
E-Commerce Bookstore (Book)	125	3564	7791	36
Course Project Manager (CPM)	890	12352	9300	96
DSpace	75	3183 (3023 HTML pages)	49513	20

Deterministic behavior

Nondeterministic, real-time behavior

*Non-Commented Lines of Code

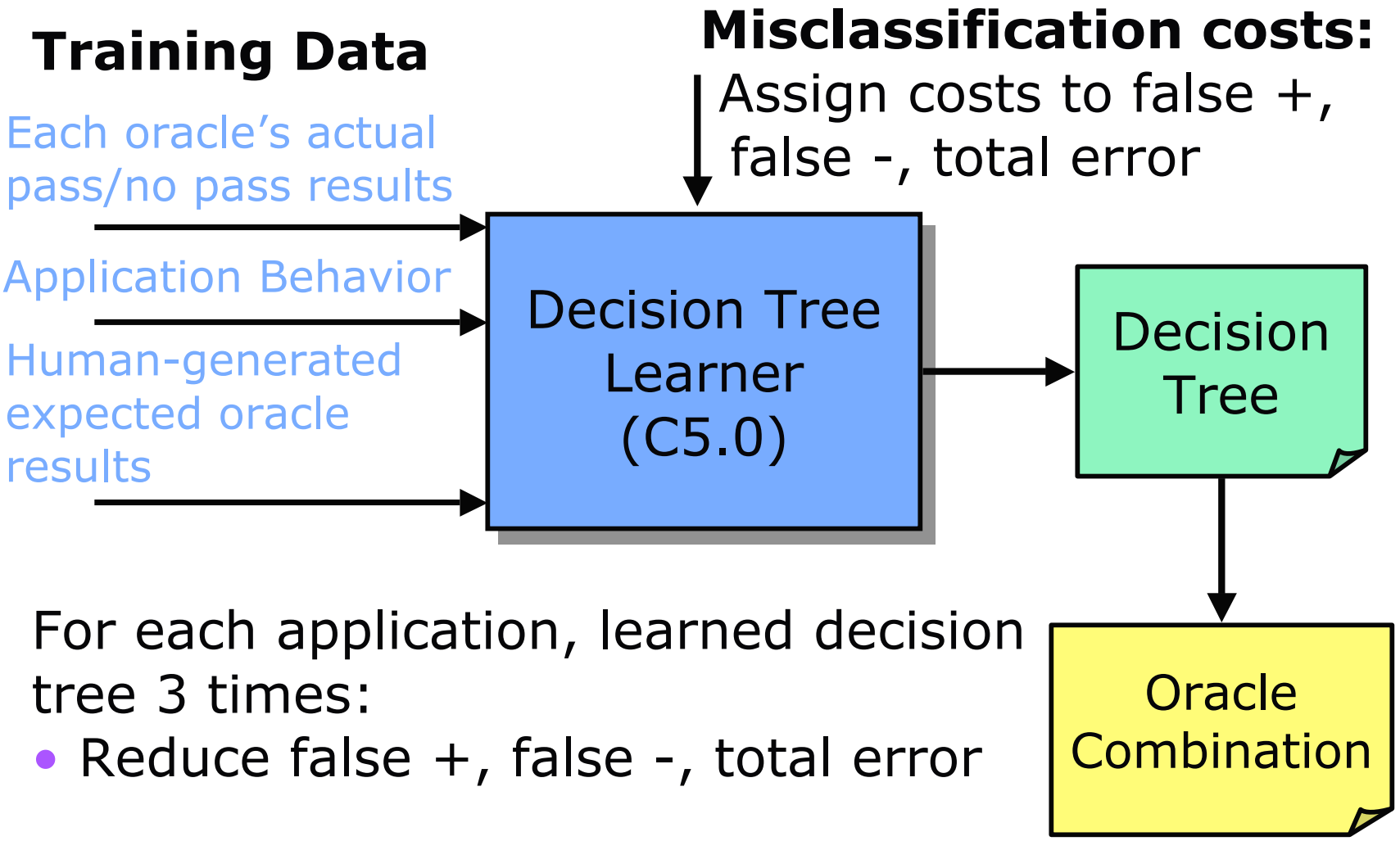
Total: 174 Faults

Expected Oracle Results

Humans determined if an HTML response exposed a failure

App	Pass		No Pass	Total
	Trivial	Non-Trivial		
Masplas	22924	1208	134	24266
Book	107990	18764	1550	128304
CPM	1164661	19449	1682	1185792
DSpace	53356	5651	4653	63660
Total	1348931	45072	8012	1402022

Decision Tree Learning



For each application, learned decision tree 3 times:

- Reduce false +, false -, total error

Example Decision Tree Output

DSpace - Reduce False Negatives

Decision tree:

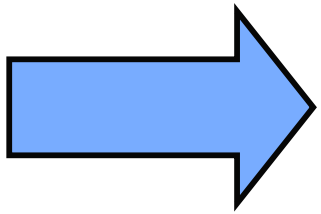
Says "no pass"

right/wrong

```
cwd = 1: 1 (6164/522)
cwd = 0:
:...f = 0: 0 (57487)
    f = 1: 1 (9)
```

Says "pass"

Says "no pass"



C-WD ∪ N+F

Measuring Effectiveness

- Do not include trivial passes
  - Better distinguish between comparators

- False positive error rate

$$\frac{\#\text{ reported failures}}{\#\text{ non-trivial pass} + \#\text{ no pass}}$$

- False negative error rate

$$\frac{\#\text{ failures missed}}{\#\text{ non-trivial pass} + \#\text{ no pass}}$$

Learned Oracle Results

App	Oracle	False -	False +	Total
Masplas	DB-W	0	0	0
Book	T ∪ C	4.25	0	4.25
	D	0	7.63	7.63
CPM	dD ∪ N+F-Sel	0.43	0	0.43
	D	0	7.96	7.96
DSpace	N+U	.45	0.04	0.49
	N-S+U	0.82	0	0.82
	C-WD ∪ N+F	0	5.07	5.07

resp behavior

- Generally, low error rates
- Tradeoffs between false positives/false negatives

Learned Oracle Results

App	Oracle	False -	False +	Total
Masplas	DB-W	0	0	0
Book	T ∪ C	4.25	0	4.25
	D	0	7.63	7.63
CPM	dD ∪ N+F-Sel	0.43	0	0.43
	D	0	7.96	7.96
DSpace	N+U	.45	0.04	0.49
	N-S+U	0.82	0	0.82
	C-WD ∪ N+F	0	5.07	5.07

- Cross-validation: applying learned oracle on *other* three applications did not yield an effective oracle

Evaluation of Learned Oracles

- Evaluate on non-training set data
- Experiment 1: Effect of nondeterministic, real-time behavior
  - Focus on false positives
  - Executed CPM and DSpace test suite 9 times over several months
- Experiment 2: Failure detection
  - Seeded faults in all four applications
  - Two sets of data
    - ➔ Failure detection results
    - Normalized failure detection results to reduce bias towards CPM

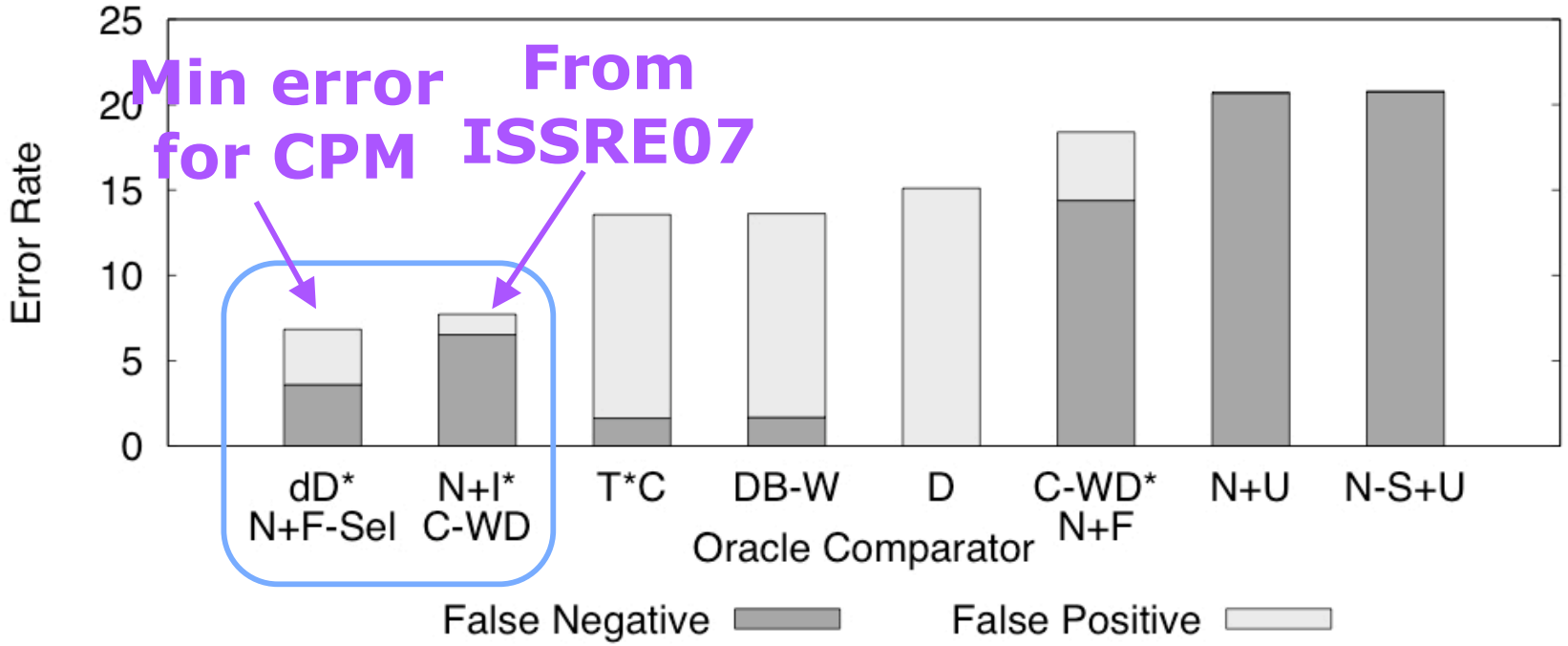
Experiment 1 Results

Learned from CPM and DSpace

Oracle Comparator	False Positive Error	
	CPM	DSpace
N+U	0	20.44
N-S+U	0	9.36
dD ∪ N+F-Sel	38.7	10.9
C-WD ∪ N+F	98.8	37.6
N+I ∪ C-WD	100	51.8
T ∪ C	100	100
DB-W	100	100
D	100	100

- Higher error because compared test suite executions separated by several months

Experiment 2 Results



- Error rates < 1.5% if include trivial pass
- Fewest false negatives: document-based
- Need to consider both structure, content

Methodology for Selecting an Oracle Comparator Combination

1. Create oracle training data
   - Use faults from bug reports
   - Execute oracles on test-suite responses
   - Manually determine pass/no pass results
   - Identify responses with nondeterministic behavior
2. Use decision trees to learn the best oracle combination
   - Tailor to goals: fewest false positives, false negatives, total error
3. Evolve combination as application evolves
   - Application's behavior and what constitutes a failure may change
   - Repeat steps 1 and 2

Related Work

- HTML-based oracle comparators
  - Elbaum05, DiLucca 02 -- few details of implementation, no evaluation of false positives, negatives with nondeterminism
- Machine learning to classify program executions as pass/no pass
  - Bowring04, Haran07, Podgurski03
  - Use execution profile for features

Conclusions & Future Work

- Proposed methodology for selecting effective oracle comparator combinations
  - Decision tree learning on representative faults from bug reports
- Evaluated learned oracles on non-training set data
- Future Work
  - Evaluate on additional faults
  - Evaluate on additional applications