Automated Repair of HTML Generation Errors in PHP Applications using String Constraint Solving

Hesam Samimi*, Max Schäfer†, Shay Artzi†, Todd Millstein*, Frank Tip†, and Laurie Hendren‡

*University of California, Los Angeles † IBM Research ‡McGill University, Montreal, Canada

WHAT

- Automatic fixing of PHP program HTML generation bugs
- Dynamic approach based on test suite input/output specifications
- By string constraint solving

PROBLEM

- Web browsers tolerate HTML errors, silently correcting them
- Encourages programmers to release buggy applications
- Different browsers correct HTML errors inconsistently
- Buggy programs may be displayed differently depending on browser
- Malformed HTML may even stall browser or expose security vulnerabilities

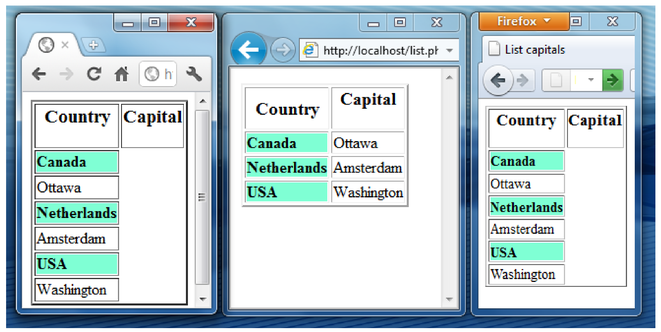

Figure: Different rendering of invalid HTML page in Google Chrome 13.0 (left), Internet Explorer 9.0 (middle) and Firefox 6.0 (right)

OVERVIEW

- PHP scripts generate HTML content based on user inputs, SQL databases
- `print` statements visited during the execution of the script make up the final HTML page
- Automated tool *HTMLTidy* or browser itself can be used to correct bad HTML
- Common HTML generation errors are simple
  - Missing/misplaced/extra HTML tags
  - Wrong tag attributes

MOTIVATION

- Fixes to these bugs often simply involves changes to string literal `print` statements
- Usually easy to correct bad HTML (automatic or manual)
- Not trivial to correct the PHP program that generated it

IDEA

- Instrument program to log `print` statements visited during execution of each test
- Each string literal `print` statement represented by a variable
- Repair constraints for each test:
  - Concatenation of visited variables = expected output
- Solve string constraint problem → correct values for each `print` statement
- Minimal Cost solution → least amount of change to original program

EXAMPLE

Program

```
1 print "<html>"^c1;
2 if (isset($_REQUEST["table"])) {
3   print "<table>"^c2;
4   $header = array("Country", "Capital");
5   print "<tr>"^c3;
6   foreach ( $header as $hvalue ) {
7     print "<th>"^c4 . $hvalue . "</tr>"^c5;
8   }
9   print "</th></table>"^c6;
10 }
11 print "</html>"^c7;
```

Tests

ID	Input	Expected Output
t_1	$\emptyset$	`<html></html>`
t_2	table = "1"	`<html><table><tr><th>Country</th><th>Capital</th></tr></table></html>`

1. Test Runs

ID	Input	Actual Output	Pass?
t_1	$\emptyset$	`<html></html>`	yes
t_2	table = "1"	`<html><table><tr><th>Country</tr><th>Capital</tr></th></table></html>`	no

2. String Constraints

ID	Instrumented Output	= Expected Output
t_1	$c_1.c_7$	= `<html></html>`
t_2	$c_1.c_2.c_3.c_4$.Country.$c_5.c_4$.Capital.$c_5.c_6.c_7$	= `<html><table><tr><th>Country</th><th>Capital</th></tr></table></html>`

3. Cost Optimizing Solution

Print Var	Value	Print Var	Value	Print Var	Value
c_5	`</th>`	c_6	`</tr></table>`	$c_i (i \notin \{5,6\})$	unchanged

Repaired Program

```
1 print "<html>"^c1;
2 if (isset($_REQUEST["table"])) {
3   print "<table>"^c2;
4   $header = array("Country", "Capital");
5   print "<tr>"^c3;
6   foreach ( $header as $hvalue ) {
7     print "<th>"^c4 . $hvalue . "</th>"^c5;
8   }
9   print "</tr></table>"^c6;
10 }
11 print "</html>"^c7;
```

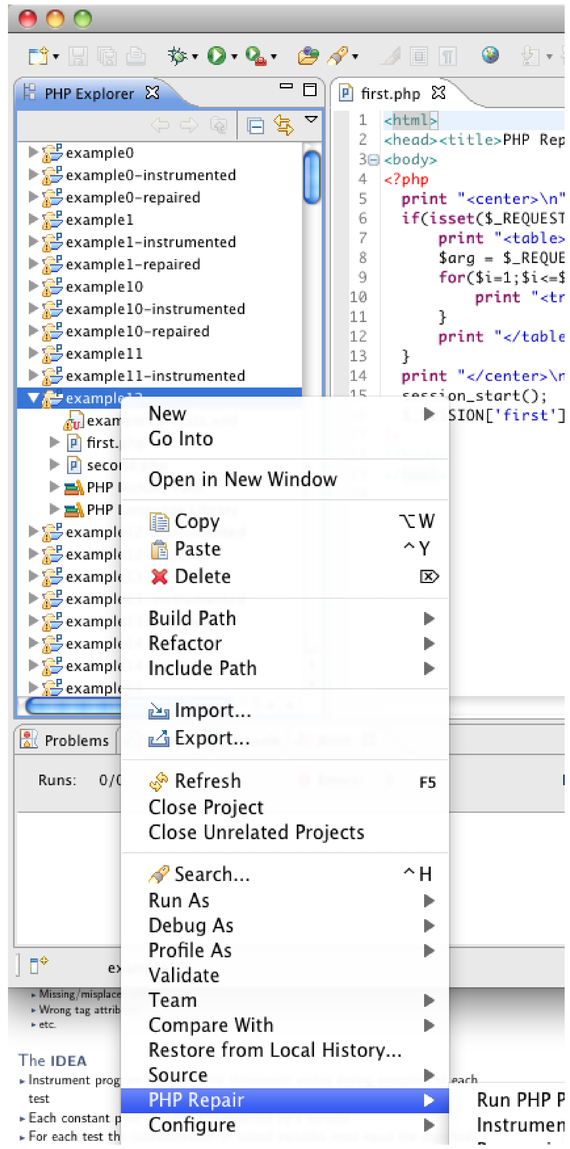

Figure: PHPRepair in PHP Development Toolkit Eclipse plugin

IMPLEMENTATION

- PHP Development Toolkit Eclipse plugin + Automated Repair feature:
  - *PHPQuickFix*: static repairs for simple cases (e.g. `<a><b></a>`)
  - *PHPRepair*: dynamic (test suite) approach above for other cases
- Localization heuristic based on *diff* between actual and expected outputs
  - Makes approach scalable to real apps (reduced search space)
  - Gradual heuristic backoff to maintain *completeness*
- String solving using Kodkod relational SAT-based solver
  - Cost optimizing SAT solver for minimal repairs

EVALUATION and RESULTS

- Approach is **sound**, **complete**, and **minimal** (on literal string fixes)
- 6 real applications from Sourceforge
- Many simple errors fixed using PHPQuickFix
- **86%** all remaining patches automated by PHPRepair
- remaining manual patches out of scope:
  - control flow changes, `print` statements with variables
- **7 sec.** avg time to repair a failing test