Modular Verification of Web Page Layout

Pavel Panchekha,
Michael D. Ernst, Zachary Tatlock, Shoaib Kamil
Web page layouts are programs.
Layout Bugs

Legal liability under the ADA

Image occludes button
Verification tools

Heuristic
ReDeCheck [ASE’15]
rRLG [ISSTA’17]
IFIX [ICST’18]

Specifications
Cornipickle [ICST’15]
VizAssert [PLDI’18]

Page
Monolithic
Independent
Verification tools

Heuristic
ReDeCheck [ASE’15]
rRLG [ISSTA’17]
IFIX [ICST’18]

Specifications
Cornipickle [ICST’15]
VizAssert [PLDI’18]
Verification tools

VizAssert:
- Monolithic
- Independent

Troika:
- Modular
- Resuable
Troika

Proof Assistant for Web Page Layout
A Troika Proof

Goal:

Every link is scrollable

∀b, b ∈ $(a) \implies b . \text{top} \geq 0 \land b . \text{left} \geq 0
A Troika Proof

\[ \forall b, b \in $(a) \implies b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0 \]

1. Decompose web page
A Troika Proof

∀b, b ∈ $(a) \implies b\.top \geq 0 \land b\.left \geq 0$

1. **Decompose** web page

2. **Modularize** property

∀b ∈ H, b ∈ $(a) \implies b\.top \geq 0 \land b\.left \geq 0$

3. **Verify** components
A Troika Proof

∀b, b ∈ $(a) \implies b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0$

1. **Decompose** web page

2. **Modularize** property

∀b ∈ H, b ∈ $(a) \implies H \cdot \text{top} \geq 0 \land H \cdot \text{left} \geq 0 \implies b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0

Rely/Guarantee preconditions
Troika

Decomposing web pages

Verifying components

Composing properties
Troika

Decomposing web pages

Verifying components

Composing properties
Components

Within Page
Tabs, comments, items

Across Pages
Headers, menus, footers

Across Sites
Widgets, themes

In Troika, identified by CSS selectors
Components

Template

Components

Components + template form partition of page
Troika

Decomposing web pages  Verifying components  Composing properties
Troika

Decomposing web pages  Verifying components  Composing properties
Interdependence

Template's depends on component's

Component's layout depends on template's
Isolating Components

Component's layout depends on template's
Isolating Components

Component's layout depends on template's

Arbitrary template
Isolating Components

Precondition
- Width available
- Current font size
- Floating boxes

Component's layout depends on template's

Component
How Verification Works

val page = "http://healthcare.gov/"

abstract params : BrowserParams

abstract layout : Layout

require layout = render (page, params)
require ¬ theorem(layout)

Valid counterexample to web page theorem
How Verification Works

```scala
val template[component] = "http://healthcare.gov"

abstract params : BrowserParams

abstract layoutt[layoutc] : Layout

require precondition(layoutt)

require layoutc = render (component, params)

require ¬ c.property(layoutc)
```
Troika

Decomposing web pages

Verifying components

Composing properties
Troika

Decomposing web pages

Verifying components

Composing properties
Composing proofs

Template

Precondition

Component

Precondition $\Rightarrow$ Property
Composing proofs

Checking implications is whole-page reasoning!

Very slow on large pages
Whole-page Logic

Why so slow? Complex browser behavior

Reasoning about rendering: complex formulas

Many real & boolean variables

Idea: check preconditions without layout model

Component properties abstract over layout
Whole-page Logic

val page = "http://healthcare.gov/"

abstract params : BrowserParams

abstract layout : Layout

require layout = render (page, params)
require ¬ theorem(layout)
val page = "http://healthcare.gov/"

abstract params : BrowserParams

abstract layout : Layout

for c ∈ C, require c.property(layout_c)

require layout = render (page, params)

require ¬ theorem(layout)
Troika

Decomposing web pages

Verifying components

Composing properties
Troika

Decomposing web pages

Verifying components

Composing properties
Benefits

Problem Size + Parallelism + Caching

Modularity enables scale
Large web page
Multiple similar pages
Multiple similar sites
Modular proofs

3 pages verified against 2 theorems
All links within scrollable area
All lines less than 80 characters long

30, 39, and 49 components
Sidebar, heading, each paragraph of post

30 and 23 lines of proof
Including boilerplate, components, and properties
Faster Verification

3-4 minutes to verify each page

Monolithic verification up to 19 hours

Even faster for incremental proof development
8 Parallel Threads

Parallelizes well at 8 threads

- Easy to parallelize across computers
- Finer-grained components → more parallelism
Caching

Significant reuse of components

Reuse between different blog posts on same blog

20% of components reused, 12% faster

No caching of article text
Proof reuse

Proofs reusable across sites

- Different WordPress blog with same theme
- Same proof applied without change
- Shared tweaks gave 6× speed-up
Benefits

Problem Size + Parallelism + Caching

Modularity enables scale
Decomposing web pages

Verifying components

Composing proofs
## All Results

<table>
<thead>
<tr>
<th>Property</th>
<th>Components</th>
<th>Time (s)</th>
<th>Time (s, 8x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joel Post 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>39</td>
<td>184 (3.2x)</td>
<td>30 (16x)</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>159 (130x)</td>
<td>27 (780x)</td>
</tr>
<tr>
<td>Joel Post 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>49</td>
<td>194 (3.0x)</td>
<td>44 (11x)</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>209 (340x)</td>
<td>48 (1500x)</td>
</tr>
<tr>
<td>Other Blog</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>30</td>
<td>214 (9.0x)</td>
<td>66 (35x)</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>179 (10x)</td>
<td>52 (33x)</td>
</tr>
</tbody>
</table>

19 hours without modularity!