Modular Verification of Web Page Layout

Pavel Panchekha, Michael D. Ernst, Zachary Tatlock, Shoaib Kamil
Web Page Layout

Web page layouts are programs

HTML
CSS
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

\[ \forall b, b \in \mathbb{R}(a) \implies b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0 \]
A Troika Proof

∀b, b ∈ $(a) \implies \begin{align*} b \cdot \text{top} & \geq 0 \\ b \cdot \text{left} & \geq 0 \end{align*}

1 Decompose web page
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 ∈ $(a) \implies b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0$

3. **Verify** components
A Troika Proof

∀b, b ∈ $\mathcal{A}$  \implies  
\[ b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0 \]

1. Decompose web page

2. Modularize property

∀b ∈ H, b ∈ $\mathcal{A}$  \implies  
\[ H \cdot \text{top} \geq 0 \land H \cdot \text{left} \geq 0  \implies  
\[ b \cdot \text{top} \geq 0 \land b \cdot \text{left} \geq 0 \]
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 layout depends on component's

Component's layout depends on template's

Template's depends on component's
Isolating Components

Component's layout depends on template's
Isolating Components

Component's layout depends on template's
Isolating Components

Precondition: Width available, Current font size, Floating boxes

Component's layout depends on template's

Component

A  B  C
How Verification Works

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

abstract params : BrowserParams

abstract layout : Layout

require layout = render (page, params)
require \( \neg \) theorem(layout)

Valid counterexample to web page theorem
How Verification Works

val template[component] = “http://healthcare.gov”

abstract params : BrowserParams

abstract layout[to][layout] : Layout

require precondition(layout)

require layout = render (component, params)

require ¬ c.property(layout)

Solve for template layout

Only render component
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

| Page | Component | & & |
|------|-----------|-----|
| Precondition | & & | Precondition ⇒ Property |

Checking implications is **whole-page reasoning**!
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)
Whole-page Logic

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
The firm that eventually invested in us, Union Square Ventures, told us that they were so excited by the power of product - that they were only investing in companies that incorporated some kind of game play.

For example, Firepower. Remember Firepower? It was all about making your normal web MIS software go to some cool place where you could click on a game that excitedly generated lots of data for marketers. Or Oranges, which is a game with Flash cards that teaches you foreign languages. These were other VCs investments from that time period.

In the book, I described this as a moment of realization that Stack Overflow has "garnished" a lot of attention. I really don't think there's much of a sexual reputation.

Stack Overflow reputation started as a very simple game. There's a game to play, and you get in trouble if you make a mistake. You get the words right to Attribute the developers who were known for this answer, but it was a bit of a shock. Also, it's not like the person who wrote the answer was odious: that's an effect of the goal.

You would think if your answers were deleted, but you are actually only lose 3 points. We didn't want to punish you as much as we wanted to show where people that your answer was wrong. And we'd actually make you pay the reputation point to disassociate yourself, so you're still rewarded.

That was pretty much the whole system.

Now, this wasn't an original idea. It was originally inspired by kameo, which started as an integer that appeared in parentheses after your name. If you minimized everything you'd got scored, your karma went up as a "reward". That was it. Karma didn't tie a single thing, but it still served as a system for reward and punishment.

What important and karma do is send a message that this is a community with norms. It's not just a place to type words onto the internet. I think that's why they don't want the audience of letting you exercise your freedom of speech. You can get your freedom of speech somewhere else. Our goal is to get the best answers to questions. All the voting makes it clear that are some standards, that some marks are better than others. And that the community itself has a

---

Size & Parallelism

Caching

Proof reuse

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

Bar chart showing performance comparison with a Monolithic baseline.
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 6x speed-up
Benefits

- Problem Size
- Parallelism
- Caching

Modularity enables scale
Decomposing web pages
Verifying components
Composing proofs
## All Results

| Property   | Components | Time (s)     | Time (s, 8x) |
|------------|------------|--------------|--------------|
| **Joel Post 1** |            |              |              |
| A          | 39         | 184 (3.2x)   | 30 (16x)     |
| B          |            | 159 (130x)   | 27 (780x)    |
| **Joel Post 2** |            |              |              |
| A          | 49         | 194 (3.0x)   | 44 (11x)     |
| B          |            | 209 (340x)   | 48 (1500x)   |
| **Other Blog** |            |              |              |
| A          | 30         | 214 (9.0x)   | 66 (35x)     |
| B          |            | 179 (10x)    | 52 (33x)     |

19 hours without modularity!