Automated Reasoning for Web Page Layout

Formally proving pages accessible:
- Specifying visual properties
- Formalizing browser rendering
- Overcoming solver limitations
- Finding modularity in web pages
2019 Open Enrollment is over. Still need health insurance?

You can enroll in or change plans if you have certain life changes, or qualify for Medicaid or CHIP.

SEE IF I CAN ENROLL  SEE IF I CAN CHANGE

Looking for coverage for a small business? Learn more

NEED TO SUBMIT DOCUMENTS?

SEE HOW
2019 Open Enrollment is over. Still need health insurance?

You can enroll in or change plans if you have certain life changes, or qualify for Medicaid or CHIP.

SEE IF I CAN ENROLL  SEE IF I CAN CHANGE

Looking for coverage for a small business? Learn more

NEED TO SUBMIT DOCUMENTS?  SEE HOW

How We Find Layout Bugs
2019 Open Enrollment is over. Still need health insurance?

You can enroll in or change plans if you have certain life changes, or qualify for Medicaid or CHIP.

SEE IF I CAN ENROLL
SEE IF I CAN CHANGE

Looking for coverage for a small business? Learn more

NEED TO SUBMIT DOCUMENTS?

SEE HOW

How We Find Layout Bugs
Layout Bugs are Endemic

HealthCare.gov
Missing button

Bank of America
Hidden text

FAFSA
Overlapping text

Walgreens
Hidden tabs
Layout Bugs are Endemic

1. HealthCare.gov
   - Missing button

2. Bank of America
   - Hidden text

3. FAFSA
   - Overlapping text

4. Walgreens
   - Hidden tabs
Layout Bugs are Endemic

- HealthCare.gov: Missing button
- Bank of America: Hidden text
- FAFSA: Overlapping text
- Walgreens: Hidden tabs

12% of Americans have disabilities
Layout Bugs are Endemic

HealthCare.gov
Missing button

Bank of America
Hidden text

FAFSA
Overlapping text

Walgreens
Hidden tabs

12% of Americans have disabilities
Accommodation legally required (under ADA)
How We Find Layout Bugs

Industry standard:
Manual Testing

Manual review of renderings
Manual selection of configurations
How We Find Layout Bugs

Industry standard: Manual Testing
State of the Art: Automated Tests

Manual review of renderings
Auto-comparison of renderings
Manual selection of configurations
Random test array of configurations
How We Find Layout Bugs

Industry standard: Manual Testing  
State of the Art: Automated Tests  
My Work: Verification

Manual review of renderings  
Auto-comparison of renderings  
Formal specification for valid renderings

Manual selection of configurations  
Random test array of configurations  
Automated proof for all configurations
How VizAssert Works
How VizAssert Works

Formalize

Accessibility

Equations

Web Page
How VizAssert Works

- Formalize
- Solve

Accessibility ➔ Equations ➔ Web Page

- Green check mark (Equations)
- Red x (Web Page)
How VizAssert Works

Formalize  Solve  Scale Up

Accessibility  Equations  Web Page
How VizAssert Works

Formalize

Solve

Scale Up

Accessibility

Equations

Web Page
Web Pages

HTML + CSS + JS
Web Pages

- Attractive
- Easy to Use
- Mobile Friendly
- Accessible
- Simple

HTML + CSS
Web Pages

- Attractive
- Easy to Use
- Mobile Friendly
- Accessible
- Simple

HTML + CSS
Accessibility Guidelines

Developed by accessibility researchers
Accessibility Guidelines

Developed by accessibility researchers

Size & position guidelines
- Text is at least 14px tall
- Lines are at most 80 chars
- Text doesn’t overlap

Screen-reader assistance
- Screen-reader text is offscreen
- Header hierarchy matches sizes

Functionality guidelines
- Text zoom up to 200%
- Scrolling in only one dimension

All handled by VizAssert
Accessibility Guidelines

Developed by accessibility researchers

Select page elements

All handled by VizAssert

Size & position guidelines
- Text is at least 14px tall
- Lines are at most 80 chars
- Text doesn’t overlap

Screen-reader assistance
- Screen-reader text is offscreen
- Header hierarchy matches sizes

Functionality guidelines
- Text zoom up to 200%
- Scrolling in only one dimension
Accessibility Guidelines

Developed by accessibility researchers

Select page elements
Constrain geometry

All handled by VizAssert

Size & position guidelines
- Text is at least 14px tall
- Lines are at most 80 chars
- Text doesn’t overlap

Screen-reader assistance
- Screen-reader text is offscreen
- Header hierarchy matches sizes

Functionality guidelines
- Text zoom up to 200%
- Scrolling in only one dimension
Accessibility Guidelines

Guideline
Text zoom up to 200% with same functionality

HealthCare.gov
Missing button
Accessibility Guidelines

Guideline
Text zoom up to 200% with same functionality

Behavior
The search bar and search button must be inside the toolbar (for text zoom ≤ 2)
Accessibility Guidelines

Guideline
Text zoom up to 200% with same functionality

Behavior
The search bar and search button must be inside the toolbar (for text zoom \( \leq 2 \))

HealthCare.gov
Missing button
Formal Specifications

Behavior

The search bar and search button must be inside the toolbar (for text zoom ≤ 2)
Formal Specifications

Behavior

CSS Selector

div[role=banner] [type]
must be inside
#header + div + div
(for text zoom ≤ 2)

CSS Selector

HealthCare.gov
Missing button
Formal Specifications

Behavior

div[role=banner] [type]
geometry.within
#header + div + div
(for text zoom ≤ 2)
Formal Specifications

**Behavior**

The search bar and search button must be inside the toolbar (for text zoom \(\leq 2\))

**Visual Logic**

Formalize:

```
all elements matching
  div[role=banner] [type]
are geometry.within
the #header + div + div
when text.zoom \(\leq 200\%
```
Formal Specifications

Behavior
The search bar and search button must be inside the toolbar (for text zoom ≤ 2)

Visual Logic
all elements matching div[role=banner] [type] are geometry.within the #header + div + div when text.zoom ≤ 200%, window.width ≥ 800

Formalize

Only on desktop

HealthCare.gov
Missing button
Specification to Equations

Visual Logic

all elements matching
  div[role=banner] [type]
are geometry.within
the #header + div + div
when text.zoom ≤ 200%,
  window.width ≥ 800
Specification to Equations

Visual Logic
all elements matching
  div[role=banner] [type]
are geometry.within
the #header + div + div
when text.zoom ≤ 200%,
  window.width ≥ 800

Equations over Renderings
within(B, A) =
  B.left ≥ A.left &&
  B.top ≥ A.top &&
  B.left + B.width ≤ A.left + A.width &&
  B.top + B.height ≤ A.top + A.height

Sizes & positions of elements
How VizAssert Works

- Formalize
- Solve
- Scale Up

Accessibility

Equations

Web Page
Web Browser

Web Page
HTML + CSS

Configuration
Size: 1920x1280
Text zoom: 1.5x
Web Browser

Web Page
HTML + CSS

Configuration
Size: 1920×1280
Text zoom: 1.5×

Rendering

Browser

Firefox
Chrome
Safari
Edge
Modern Web Browsers:

- Millions of lines of code
- Decades of development
- Dozens of developers
Symbolic Web Browser

Web Page
HTML + CSS

Configuration
Size: $W \times H$
Text zoom: $Z \times$

Symbolic

Rendering

Browser

A
B
C
Symbolic Web Browser

Web Page
- HTML + CSS

Configuration
- Size: $W \times H$
- Text zoom: $Z \times$

Symbolic

Browser

Rendering

| A | B | C |
Symbolic Web Browser

Web Page
HTML + CSS

Configuration
Size: $W \times H$
Text zoom: $Z \times$

Browser

Rendering

Symbolic

Sizes & positions depend on $W, H, Z$
Symbolic Web Browser

Web Page
HTML + CSS

Configuration
Size: $W \times H$
Text zoom: $Z \times$

Rendering

Problem: write a symbolic web browser
Symbolic Web Browser

Web Page
HTML + CSS

Configuration
Size: \( W \times H \)
Text zoom: \( Z \times \)

Symbolic
Browser

Rendering

Problem: write a symbolic web browser

- Encodings for visual properties
- Scaling symbolic reasoning
Encoding Visual Props

Text wraps around floats on either side of it.
Encoding Visual Props

Text wraps around floats on either side of it.

“Exclusion zone” data structure
Encoding Visual Props
Encoding Visual Props

ezone.add(size, position)
Encoding Visual Props

Text wraps around floats on either side of it.

ezone.add(size, position)  ezone.place(size) \rightarrow position
Encoding Visual Props

```
ezone.add(size, position)    ezone.place(size) → position
```

Stairsteps up

Stairsteps down
Encoding Visual Props

ezone.add(size, position)  ezone.place(size) → position
Encoding Visual Props

ezone = \( y_0 \) \( y_1 \) \( y_2 \) \( y_3 \)

\[ ezone.add(size, position) \quad ezone.place(size) \rightarrow position \]
Encoding Visual Props

ezone = y_0 (y_1, l_1, r_1) (y_2, l_2, r_2) (y_3, _l_, r_3)
ezone.add(size, position)  ezone.place(size) → position

Compute place calls in one pass
Encoding Visual Props

ezone = \{(y_0, l_1, r_1), (y_2, l_2, r_2), (y_3, \perp, r_3)\}

ezone.add(size, position) ezone.place(size) \rightarrow position

O(n) equations
Validating the Formalization
Validating the Formalization

Standard Tests

W3C®
Validating the Formalization

Non-automated

Standard Tests

W3C®
Validating the Formalization

Non-automated

Standard Tests

Browsers

W3C®
Validating the Formalization

Non-automated

Standard Tests

W3C®

Browsers

Evaluate: Differential testing vs browsers

Pass thousands of conformance tests

Found bugs in existing browsers

[OOPSLA'16]
How VizAssert Works

Formalize

Solve

Scale Up

Accessibility

Equations

Web Page

Formalizing visual properties
How VizAssert Works

Formalize

Solve

Scale Up

Accessibility

Equations

Web Page

Formalizing visual properties
Solving the Equations

Equations → Solver

Proof of layout

Counter example
Solving the Equations

Equations → QFLRA → Proof of layout

Counter example
Solving the Equations

Equations → QFLRA → Proof of layout → Counter example

Quantifier Free
Linear
Real Arithmetic
Solving the Equations

Equations → QFLRA

- Proof of layout
- Counter example

Quantifier Free
Linear
Real Arithmetic → No rounding error
Solving the Equations

Equations → QFLRA

- Quantifier Free
- Linear → No multiplication
- Real Arithmetic → No rounding error

Proof of layout
Counter example
Solving the Equations

Equations → QFLRA

Proof of layout

Counter example

Quantifier Free → No nested loops
Linear → No multiplication
Real Arithmetic → No rounding error
Solving the Equations

Equations → QFLRA

- Proof of layout
- Counter example

Simulated Rounding Error:
\[ a = b \rightarrow -\varepsilon < a - b < \varepsilon \]

Real Arithmetic → No nested loops
No multiplication
No rounding error
Solving the Equations

Equations → QFLRA

Quantifier Free → No nested loops
Linear → No multiplication
Real Arithmetic → No rounding error

Proof of layout
Counter example
Solving the Equations

Equations → QFLRA → Simplifying Multiplication

- Proof of layout
- Counter example
- No nested loops
- No multiplication
- No rounding error
Solving the Equations

Equations → QFLRA

- **Quantifier Free** → No nested loops
- **Linear** → No multiplication
- **Real Arithmetic** → No rounding error

Proof of layout
Counter example
Solving the Equations

Equations → QFLRA

- Bounded work per element
  - Only one loop: render each element
- Quantifier Free → No nested loops
- Linear → No multiplication
- Real Arithmetic → No rounding error

Proof of layout

Counter example
Solving the Equations

Equations → QFLRA → Proof of layout

Problem: bounded work per element

- Incrementalization, fusion, and unrolling
- Inspired by compiler optimizations
Computing Line Height

“The line box height is the distance between the uppermost box top and the lowermost box bottom”
Computing Line Height

The line box height is the distance between the uppermost box top and the lowermost box bottom.
Computing Line Height

Ascent  Box  Baseline

“The line box height is the distance between the uppermost box top and the lowermost box bottom”

Incrementalization: update running maximum
Computing Margins
Computing Margins

“Adjoining vertical margins collapse; two margins are adjoining if and only if …”
Computing Margins

"Adjoining vertical margins collapse; two margins are adjoining if and only if ..."
Computing Margins

“Adjoining vertical margins collapse; two margins are adjoining if and only if…”

Recurse to find zero-height elts

Fusion: interleave with outer render loop
Computing Float Layout

Text wraps around floats on either side of it.
Computing Float Layout

Text wraps around floats on either side of it.

Arbitrary-size data structure
Computing Float Layout

Text wraps around floats on either side of it.

Arbitrary-size data structure

Unroll up to k corners
Solving the Equations

Equations → QFLRA → ? Insufficient k → Proof of layout

Counter example
Solving the Equations

Equations → QFLRA

Increment $k$ → Insufficient $k$

✅ Proof of layout
❌ Counter example
Verifying Real Pages

Websites from design forum FreeWebsiteTemplates.com
Verifying Real Pages

Only use formalized subset of CSS

62 web pages
Verifying Real Pages

Only use formalized subset of CSS

62 web pages
14 properties

From Apple, Google, DOJ accessibility guides
Verifying Real Pages

Only use formalized subset of CSS

62 web pages
14 properties

From Apple, Google, DOJ accessibility guides
Verifying Real Pages

- Only use formalized subset of CSS
- 62 web pages
- 14 properties
- 476 sensible combinations

From Apple, Google, DOJ accessibility guides
Verifying Real Pages

- Verified: 81%
- Reproduced: 14%
- Unreproduced: 3%
- Timed out (30min): 2%

Evaluate: verified majority of real-world inputs

- Found many real accessibility bugs
- Few false positives and few timeouts

[PLDI'18]
How VizAssert Works

1. Formalize
2. Solve
3. Scale Up

Accessibility → Equations → Web Page

Formalizing visual properties → Overcoming solver limitations
How VizAssert Works

Formalize
Solve
Scale Up

Accessibility
Equations
Web Page

Formalizing visual properties
Overcoming solver limitations
Scaling Verification

- 100 elts
- 200 elts
- 300 elts
- 400 elts

- Formalization
- Solving
Scaling Verification

Rendering is the slow part!
Scaling Verification

Problem: reason about large pages quickly
Scaling Verification

Problem: reason about large pages quickly
Scaling Verification

Problem: reason about large pages quickly

- Divide web page into small components
- Combine components with rely/guarantee logic
Isolating Components
Isolating Components

Component’s rendering depends on template’s
Isolating Components

Component’s rendering depends on template’s

Template’s depends on component’s
Isolating Components

Template

Component’s rendering depends on template’s

Component

Template’s depends on component’s

No module or function boundaries!
Isolating Components

Component’s rendering depends on template’s

Component

A B C

No module or function boundaries!
Isolating Components

Component's rendering depends on template's

Arbitrary template

Template layout: part of component configuration
Isolating Components

Precondition:
- Width available
- Current font size
- Floating elements

Component's rendering depends on template's

Component

Template layout: part of component configuration
Rely / Guarantee

Component

Precondition $\Rightarrow$ Specification
Rely / Guarantee

Precondition

Precondition $\Rightarrow$ Specification
Rely / Guarantee

Precondition

Precondition $\Rightarrow$ Specification
Rely / Guarantee

**Precondition**

&&

**Precondition \Rightarrow Specification**
Rely / Guarantee

Specification

Preconditions checked purely logically

No rendering ⇒ fast
Improving Scale

Problem Size
Improving Scale

Problem Size + Parallelism
Improving Scale

Problem Size + Parallelism + Caching
Improving Scale

Evaluate: broader class of verified pages

Scaled to 11x larger pages, to 1400x faster
Scaled to multiple pages on one site
How VizAssert Works

- Formalize
- Solve
- Scale Up

Accessibility → Equations → Web Page

- Formalizing visual properties
- Overcoming solver limitations
- Finding modularity for rely/guarantee
Future Work

Frontend code (JavaScript)

Backend code (Python, PHP, …)

Other platforms (Android, iOS…)

User studies with designers
Future Work

Frontend code (JavaScript)

Backend code (Python, PHP, ...)

Other platforms (Android, iOS, ...)

User studies with designers
Dynamic Web Pages

How can we verify pages if we don’t know their contents?
Dynamic Web Pages

P(posts)

Post N

P(posts')

Induction over structure of the page
Dynamic Web Pages

Induction over structure of the page
Future Work

- Frontend code (JavaScript)
- Backend code (Python, PHP, ...)
- Other platforms (Android, iOS, ...)
- User studies with designers
Dynamic Web Pages

\[ P(\text{posts}) \quad \rightarrow \quad P(\text{posts}') \]

Induction over structure of the page
Future Work

Frontend code (JavaScript)

**Backend code (Python, PHP, ...)**

Other platforms (Android, iOS, ...)

User studies with designers
Web Page Backends
Web Page Backends
Web Page Backends

<div id='header'><h1>A</h1><form>...</form></div>

<li class='search-result'>
<h2>
     <a href="{obj.url}">{obj.name}</a>
</h2>
<p>{obj.description}</p>
</li>
Web Page Backends

```python
write("<div id='header'><h1>A</h1><form>…")

for obj in results:
    write(" <li class='search-result'>
                <h2>
                    <a href="{obj.url}">{obj.name}</a>
                </h2>
                <p>{obj.description}</p>
            </li>")
```
Web Page Backends

write($query)

for obj in results:
    write($obj.name
    $obj.description
    )
Web Page Backends

```python
write( toolbar.height > 10 )

for obj in results:
    write( $obj.name )
    $obj.description && result.height > 0
```
Web Page Backends

```plaintext
write(
    toolbar.height > 10
)

for obj in results:
    write(
        result.y > 10 ⇒
        Q(result) && result.height > 0
    )
```
Web Page Backends

```python
for obj in results:
    write(
        toolbar.height > 10
    )

    write(
        result.y > 10 ⇒
        Q(result) && result.height > 0
    )
```
Future Work

Frontend code (JavaScript)

Backend code (Python, PHP, ...)

Other platforms (Android, iOS...)

User studies with designers
Future Work

Frontend code (JavaScript)

Backend code (Python, PHP, ...)

Other platforms (Android, iOS, ...)

User studies with designers
Future Work

- Frontend code (JavaScript)
- Backend code (Python, PHP, ...)
- Other platforms (Android, iOS, ...)
- User studies with designers
University of Utah Computer Science
University of Utah Computer Science

- Exciting city in the middle of beautiful nature
- Large & growing PL, compilers group
University of Utah Computer Science

- Exciting city in the middle of beautiful nature
- Large & growing PL, compilers group
Thank You!

Formalize
Accessibility
Formalizing visual properties

Solve
Equations
Overcoming solver limitations

Scale Up
Web Page
Finding modularity for rely/guarantee