Web Pages

Specifications section, **Lecture 10**

Pavel Panchekha

CS 6110, U of Utah

6 February 2020

Quantifier elimination

Equality $x + y = y + x$

Integer $n \mid x$

Arrays $F(i) \rightarrow G(a[i])$

First-order

Unquantified Input

Tseityn

Nelson-Oppen

Γ

Integer Array

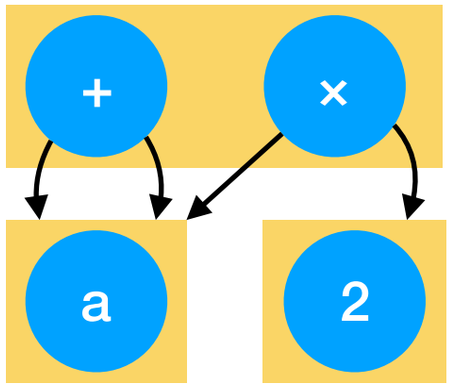
Conjunctive Form

Per-Theory Queries

DPLL(T)

Domain Reasoning

Equality



Model building

Term database

Equivalence classes

Integers

$$x + 2y \leq z$$
$$z \leq 2x - y$$
$$\downarrow$$
$$x + 2y \leq 2x - y$$

Matrix form

Variable elimination

Complexity of integers

Arrays

$$a[k := 2] = b$$
$$\downarrow$$
$$b[k] = 2$$
$$a =_k b$$

Mutation graph

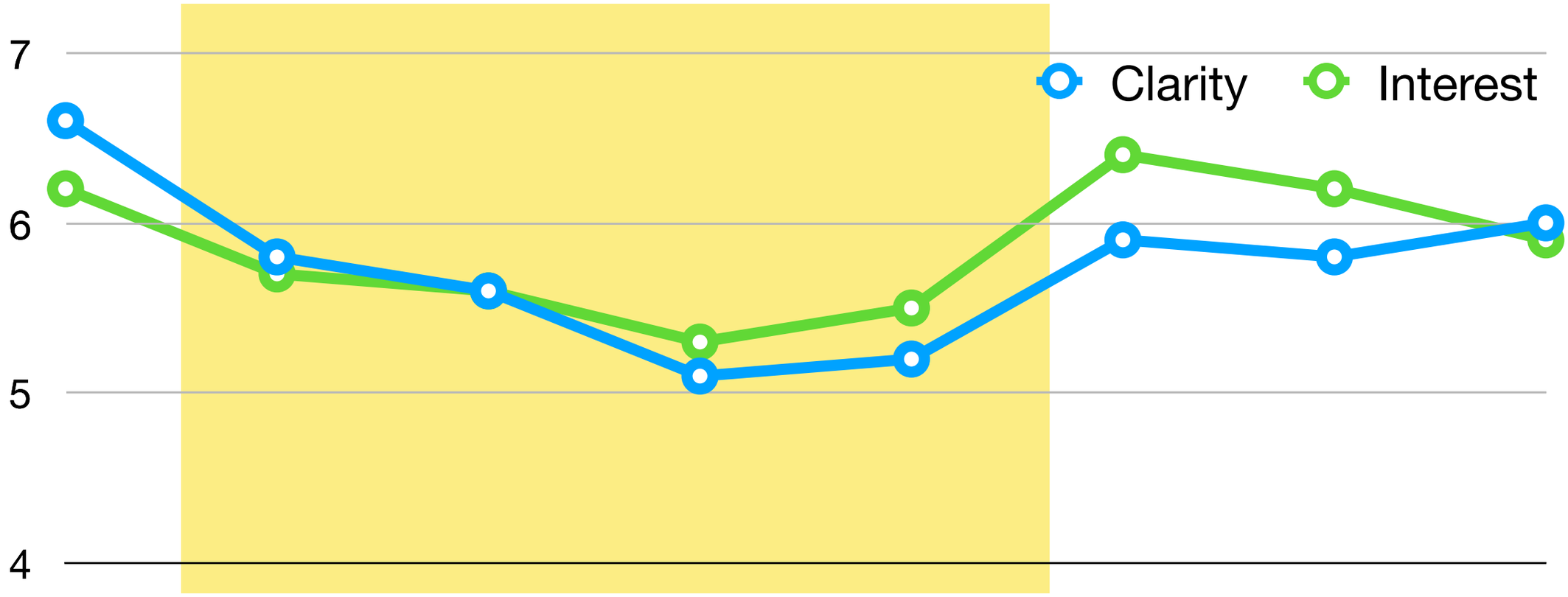
Backward propagation

Translation to theory

Theory or Practice

Stoked to move onto more algorithmic related material.

Don't listen to theory haters. I wanna freebase the theory stuff.



Web Page Reasoning

My Research on Web Page Accessibility

Get 2019 health coverage. Heal

https://www.healthcare.gov

HealthCare.gov

Español

Log in

Get Coverage | Keep or Update Your Plan | See Topics | Get Answers

Search

SEARCH

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

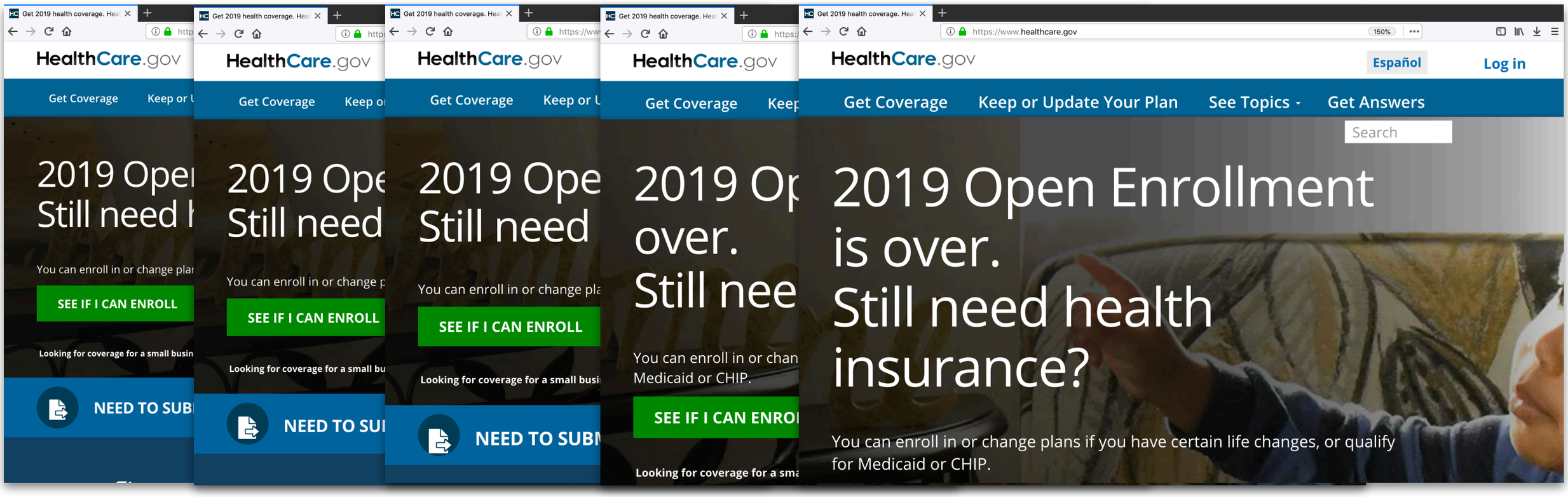
SEE IF I CAN CHANGE

Looking for coverage for a small business? Learn more

NEED TO SUBMIT DOCUMENTS?

SEE HOW

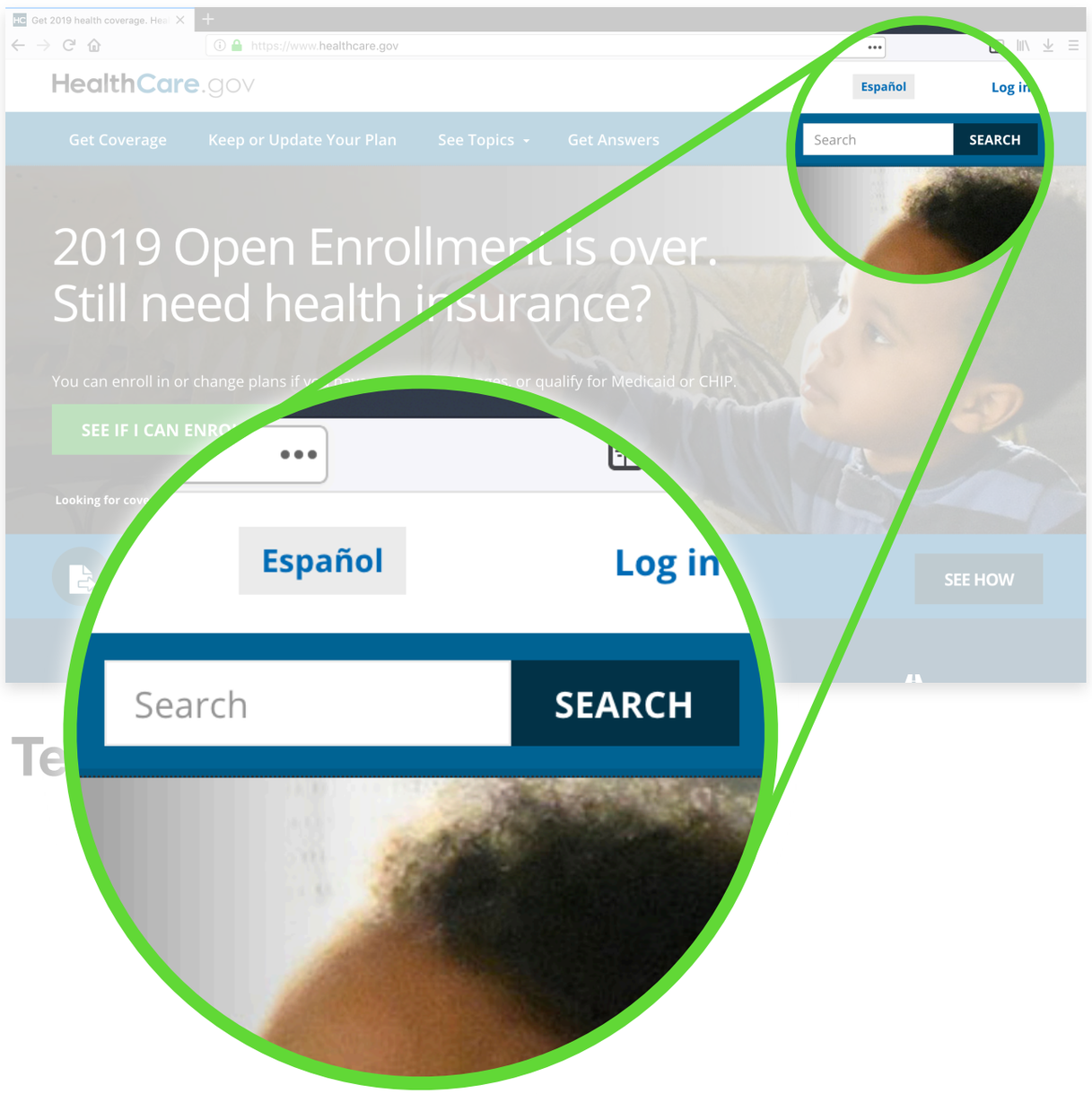
Text Zoom

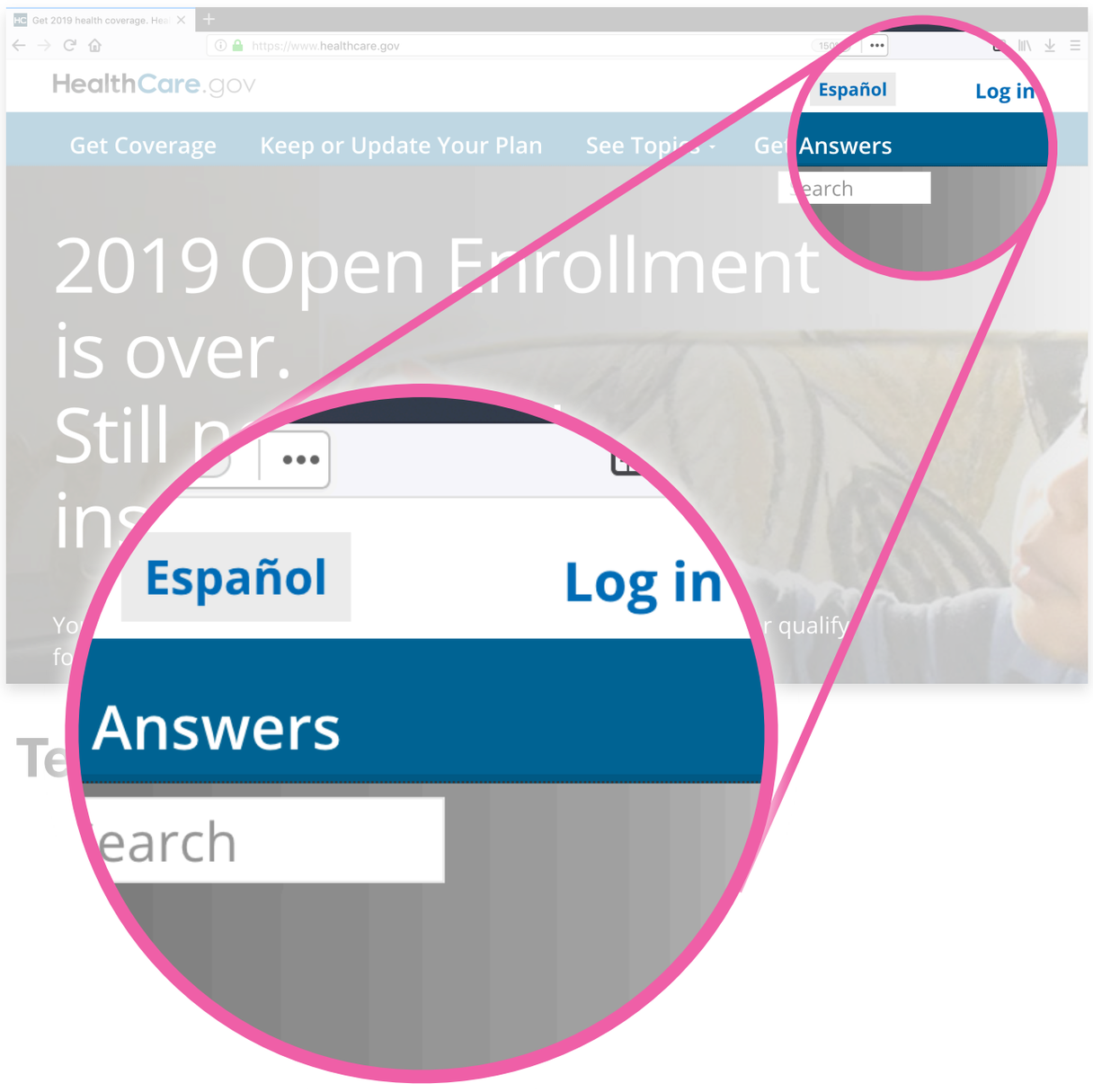


Text zoom: 100%

Text zoom: 150%

Text-zoom Bug

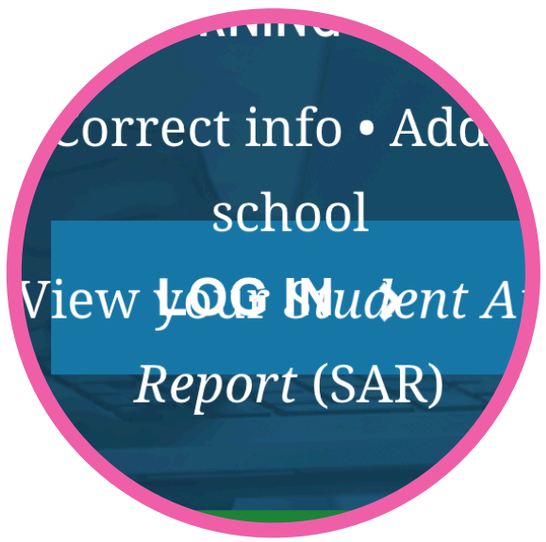




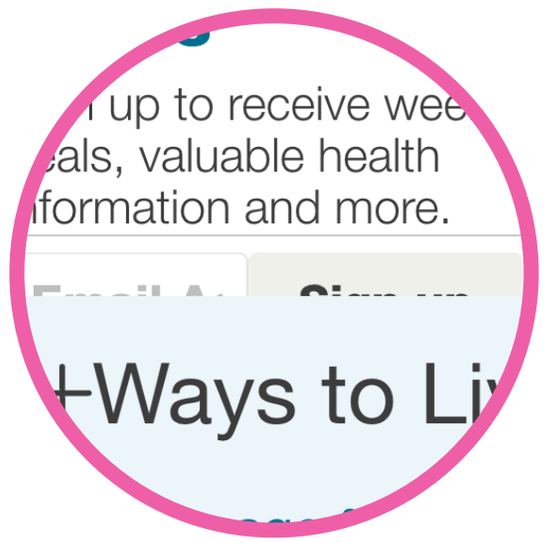
Layout Bugs are Endemic

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:

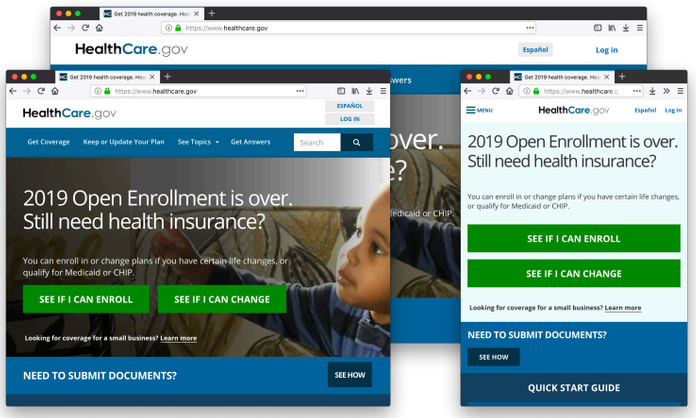
Manual Testing

State of the Art:

Automated Tests

My Work:

Verification



Manual review
of renderings

Manual selection
of configurations

Auto-comparison
of renderings

Random test array
of configurations

Formal specification
for valid renderings

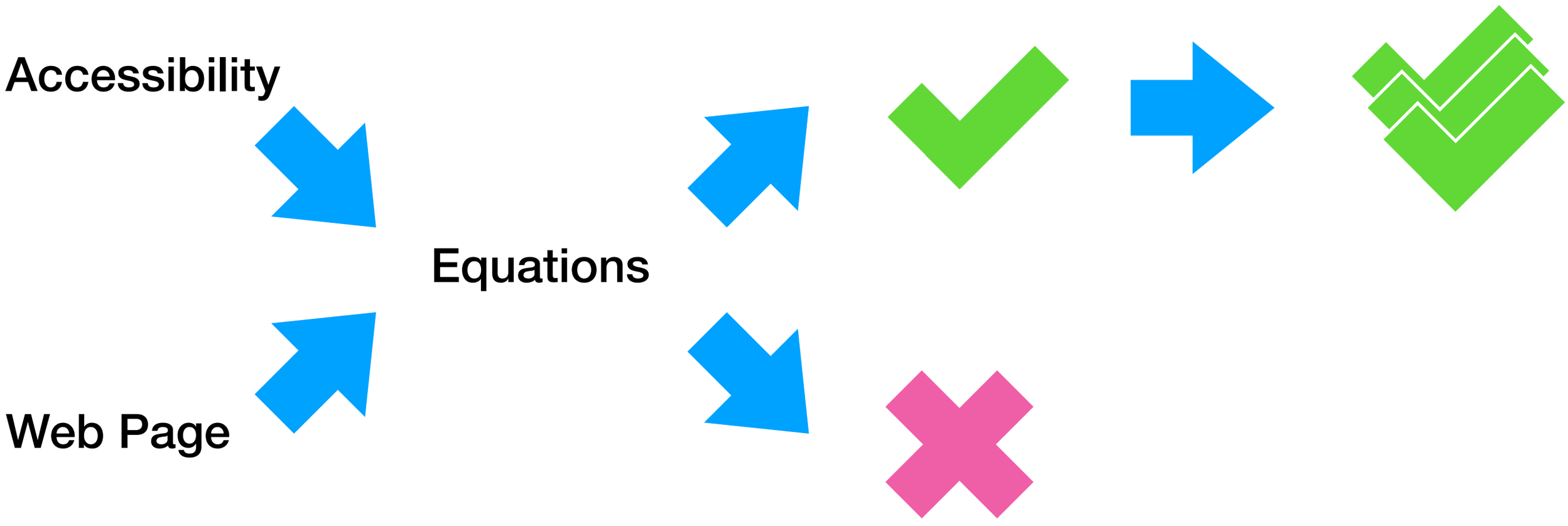
Automated proof
for all configurations

How It Works

Formalize

Solve

Scale Up



How It Works

Formalize

Solve

Scale Up

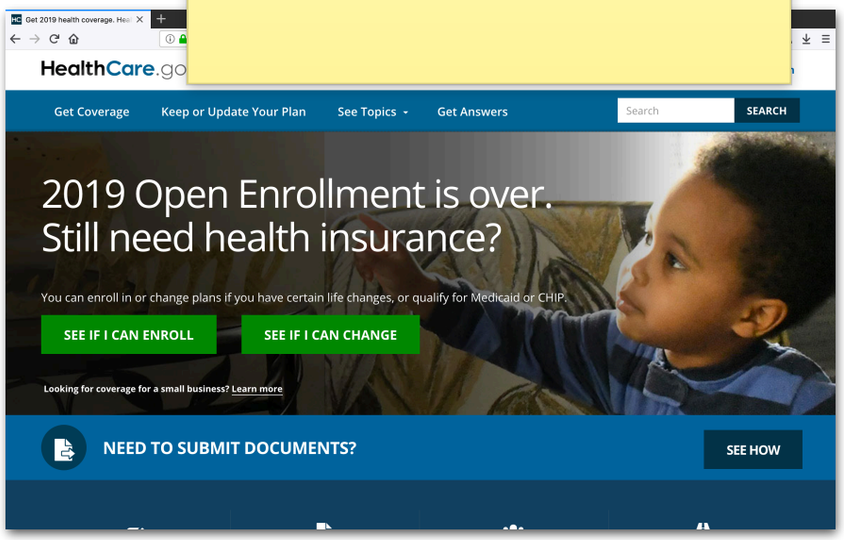
Accessibility

Equations

Web Page

Web Pages

JS is “interaction” not behavior



HTML + CSS + JS

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

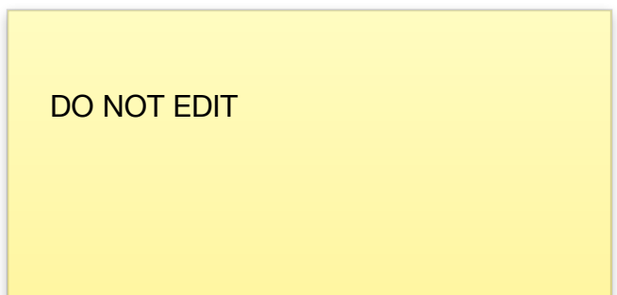
Accessibility Guidelines

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

Behavior

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

HealthCare.gov
Missing button

Formal Specifications

Behavior

CSS Selector

div[role=banner] [type]

geometry.within

Geometric primitive

#header + div + div

CSS Selector

text.zoom ≤ 200%

Inequality

HealthCare.gov
Missing button

Formal Specifications

Behavior

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

HealthCare.gov
Missing button

Formalize

Visual Logic

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

Only on desktop

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

within(B, A) :=

B.**left** $\geq$ A.**left** $\wedge \cdots$

Sizes & positions

Quantifiers

$\langle assertion \rangle ::= \forall b_1, \ldots \in \mathcal{B} : \langle cond \rangle$

$\langle cond \rangle ::= \langle cond \rangle \wedge \langle cond \rangle \mid \neg \langle cond \rangle \mid \langle cond \rangle \vee \langle cond \rangle$
$\mid \langle real \rangle = \langle real \rangle \mid \langle real \rangle < \langle real \rangle \mid \langle real \rangle > \langle real \rangle$
$\mid \langle box \rangle = \langle box \rangle \mid \langle box \rangle.\text{type} = \langle type \rangle \mid \langle box \rangle.\text{whitespace}$

$\langle real \rangle ::= \mathbb{R} \mid \langle real \rangle + \langle real \rangle \mid \langle real \rangle - \langle real \rangle \mid \mathbb{R} \times \langle real \rangle$
$\mid \langle box \rangle.\langle dir \rangle \mid \langle color \rangle.\text{r} \mid \langle color \rangle.\text{g} \mid \langle color \rangle.\text{b}$

$\langle color \rangle ::= \text{transparent} \mid \text{rgb}(\langle real \rangle, \langle real \rangle, \langle real \rangle)$
$\mid \langle box \rangle.\text{fg} \mid \langle box \rangle.\text{bg} \mid \gamma(\langle box \rangle.\text{fg}) \mid \gamma(\langle box \rangle.\text{bg})$

$\langle box \rangle ::= b_i \mid \text{root} \mid \text{null} \mid \langle box \rangle.\text{ancestor}(\langle cond^* \rangle)$
$\mid \langle box \rangle.\text{parent} \mid \langle box \rangle.\text{first-child} \mid \langle box \rangle.\text{last-child}$
$\mid \langle box \rangle.\text{next} \mid \langle box \rangle.\text{prev}$

$\langle type \rangle ::= \text{window} \mid \text{inline} \mid \text{line} \mid \text{text} \mid \text{block}$

$\langle dir \rangle ::= \text{top} \mid \text{right} \mid \text{bottom} \mid \text{left}$

Booleans + Linear Real Arithmetic

+ Equality + Domain concepts

How It Works

Formalize

Solve

Scale Up

Accessibility

Equations

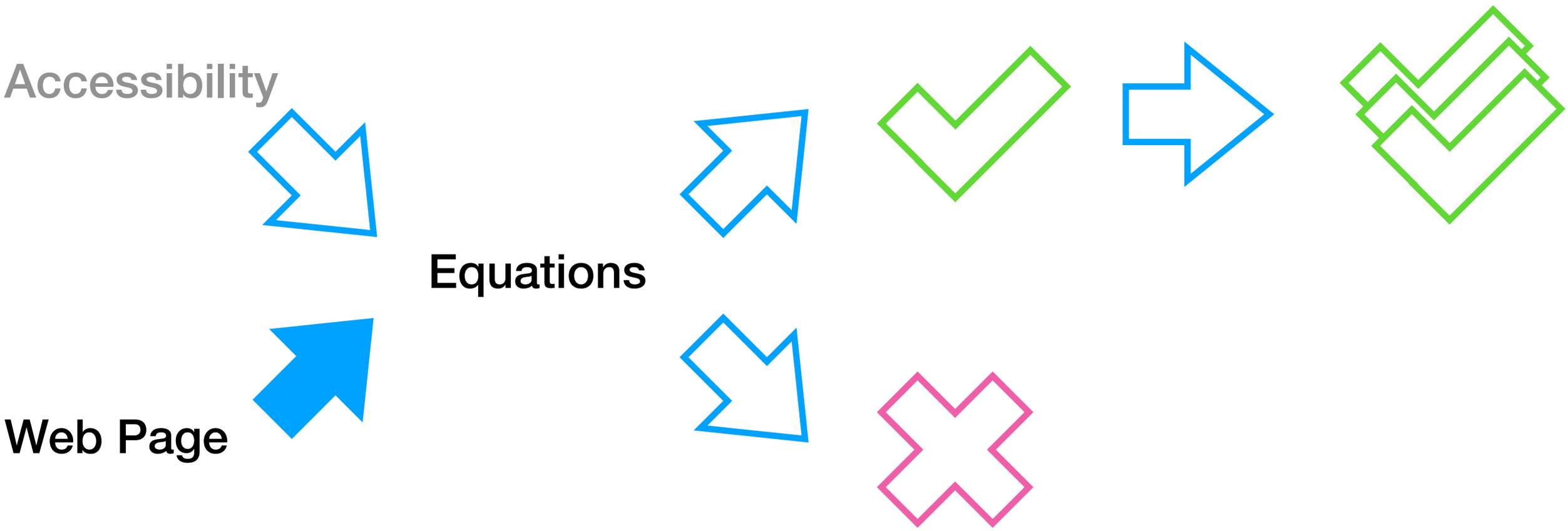
Web Page

How It Works

Formalize

Solve

Scale Up



Web Browser

Web Page

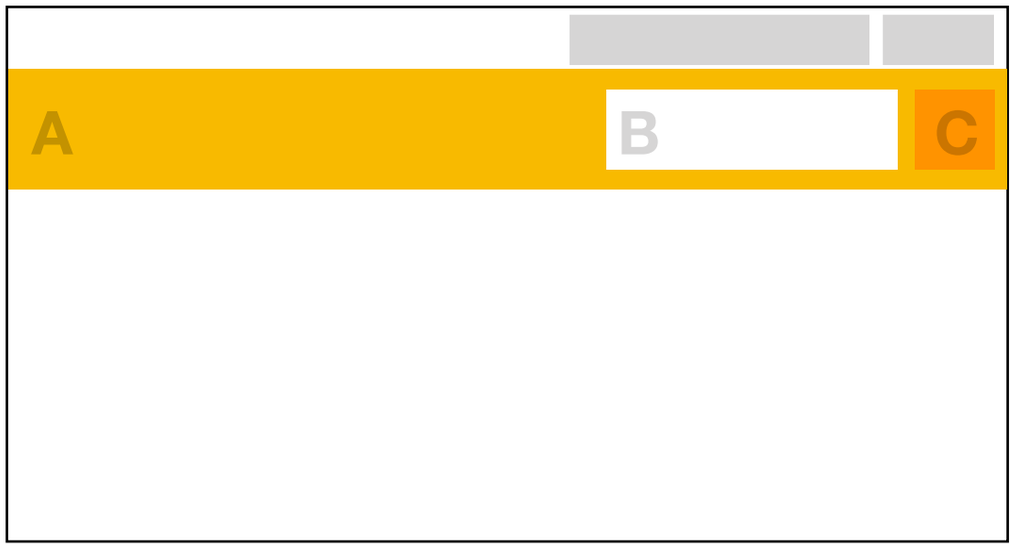
HTML + CSS

Configuration

Size: 1920×1280
Text zoom: 1.5×

Browser

Rendering



Web Browser

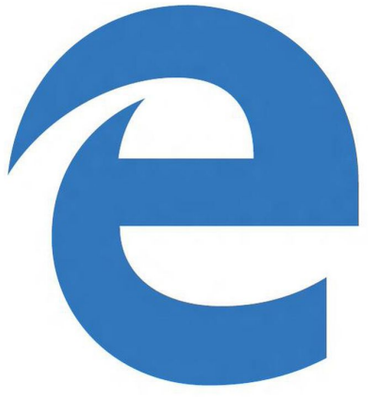
Modern Web Browsers:

- Millions of lines of code
- Decades of development
- Dozens of developers

}

Huge



Web Browser

Web Page

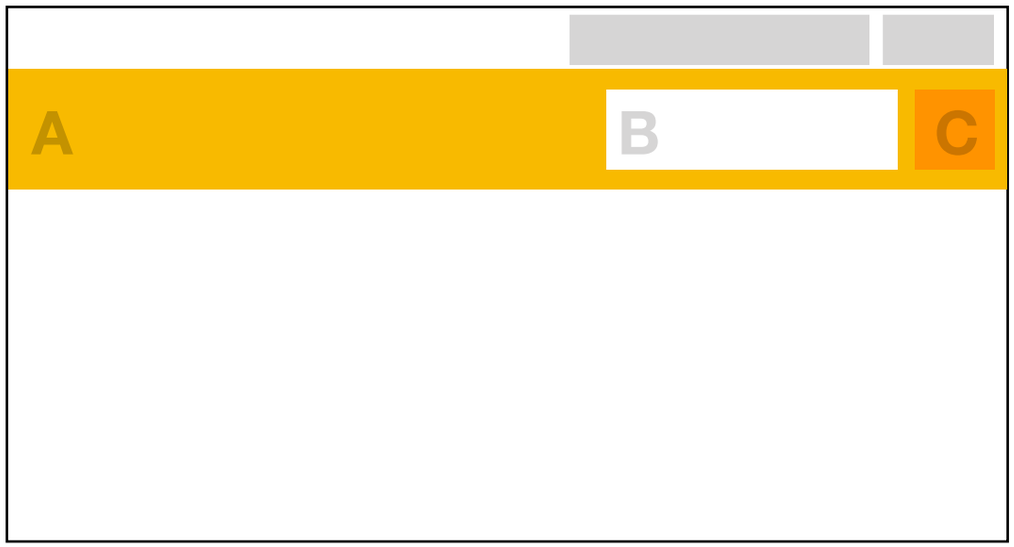
HTML + CSS

Configuration

Size: 1920×1280
Text zoom: 1.5×

Browser

Rendering



Web Browser

Web Page

HTML + CSS

Configuration

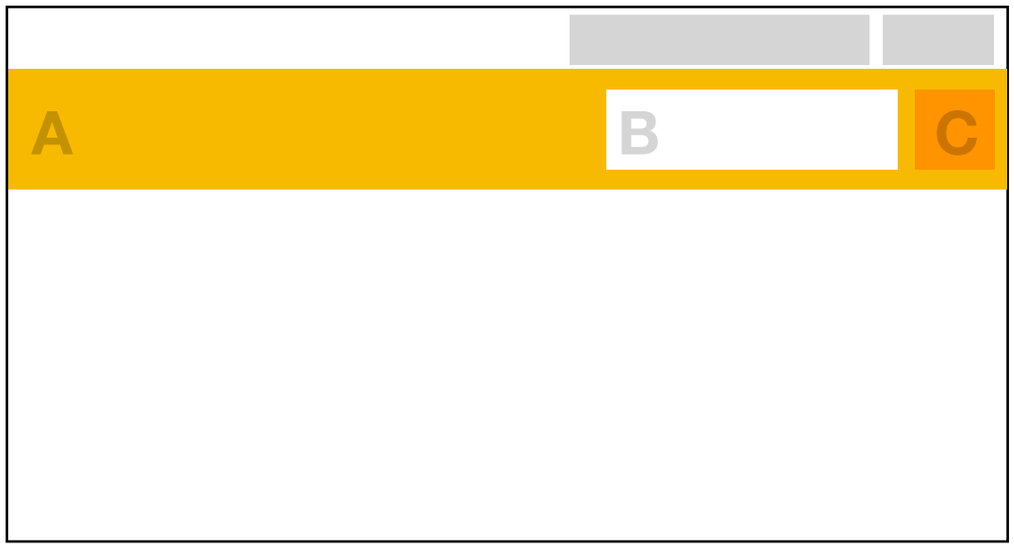
Size: $W \times H$

Text zoom: $Z\times$

Symbolic

Browser

Rendering



Symbolic Web Browser

Web Page

HTML + CSS

Configuration

Size: $W \times H$

Text zoom: $Z\times$

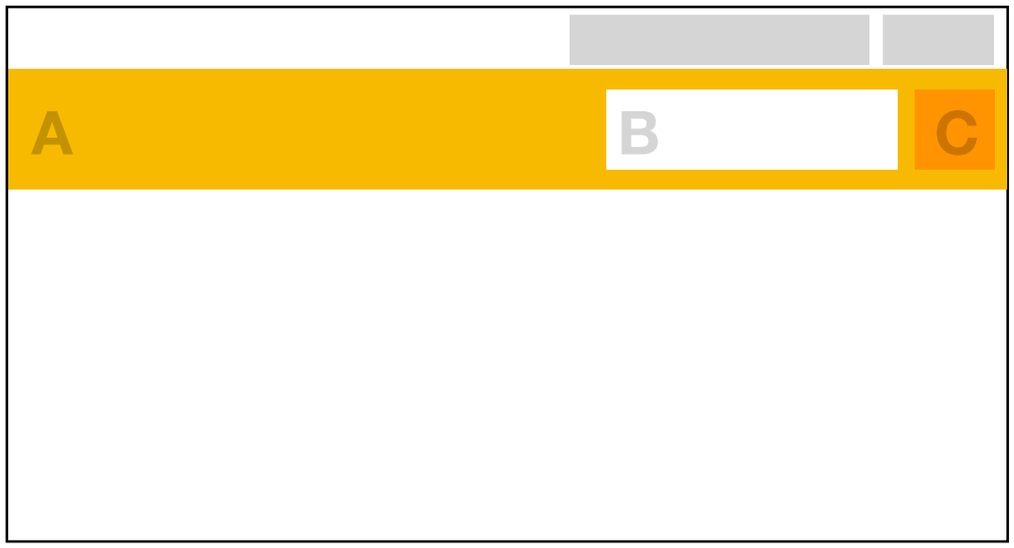
Symbolic

Symbolic

Browser

Rendering

Symbolic



Sizes & positions depend on W, H, Z

Symbolic Web Browser

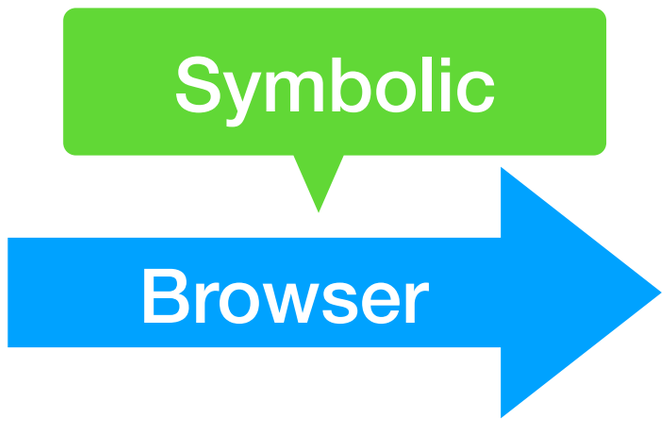
Web Page

HTML + CSS

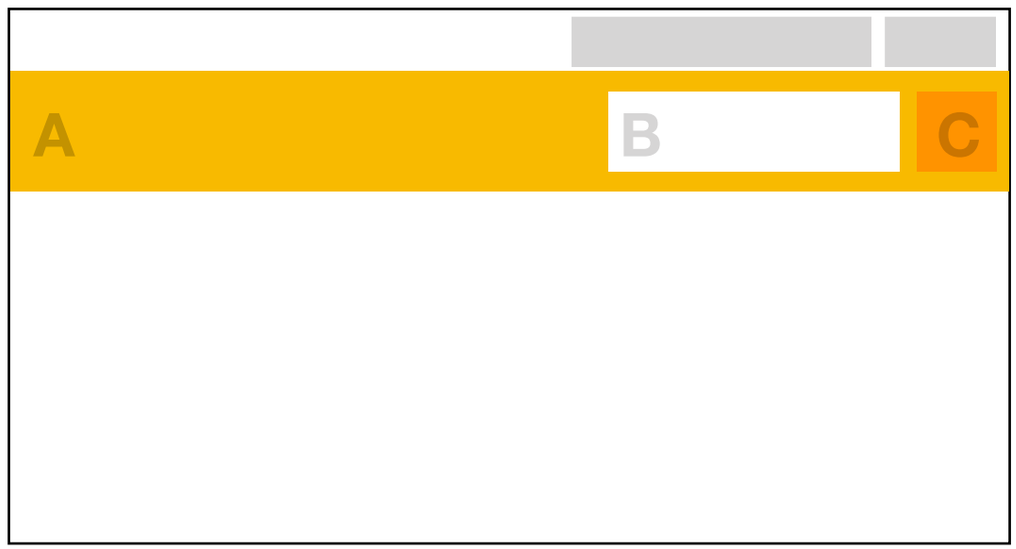
Configuration

Size: $W \times H$

Text zoom: $Z\times$



Rendering



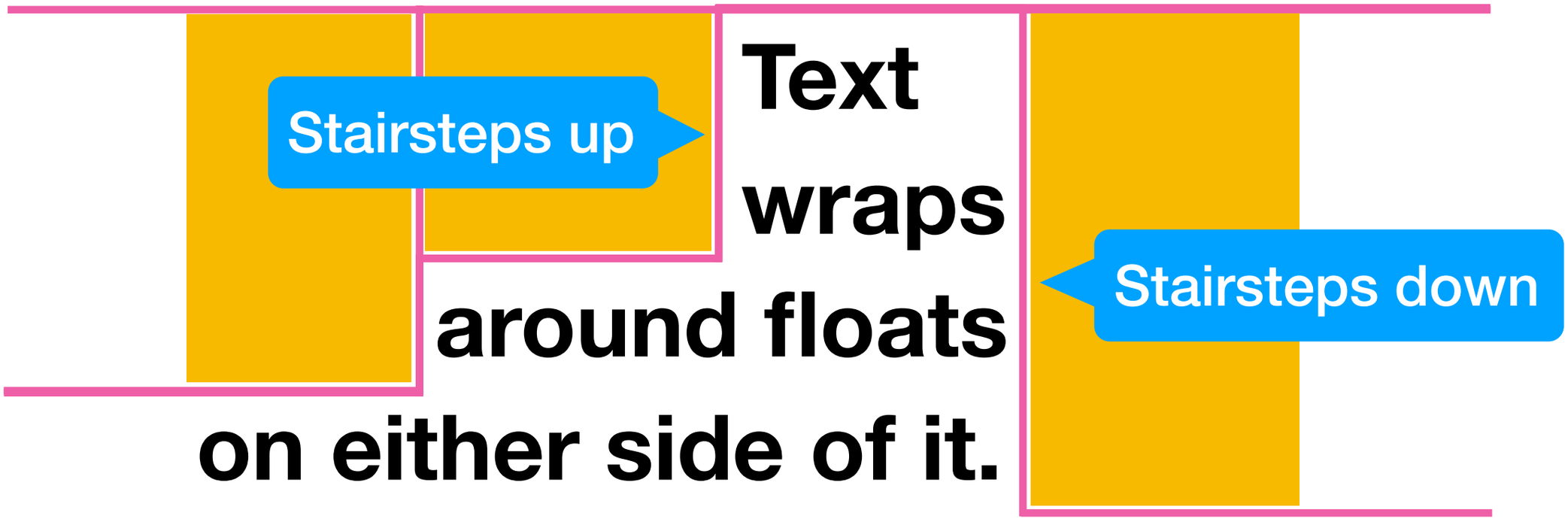
Problem: write a symbolic web browser

Encodings for visual properties

Scaling symbolic reasoning

Encoding Visual Props

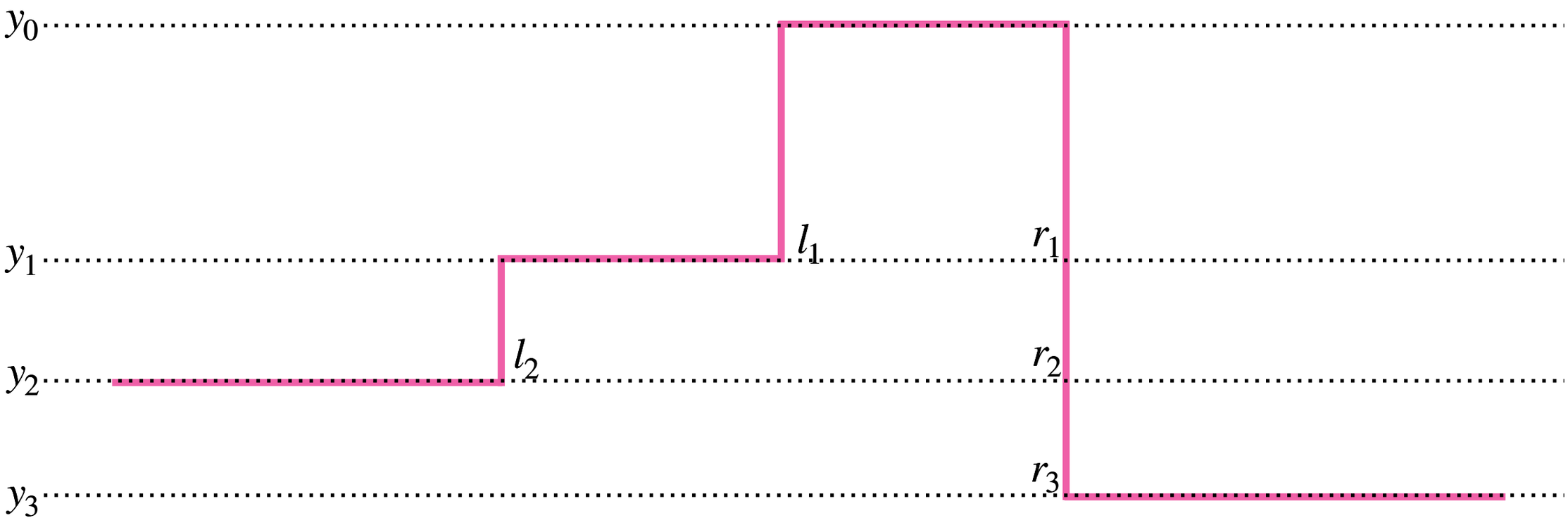
Encoding Visual Props



ezone.**add**(size, position)

ezone.**place**(size) → position

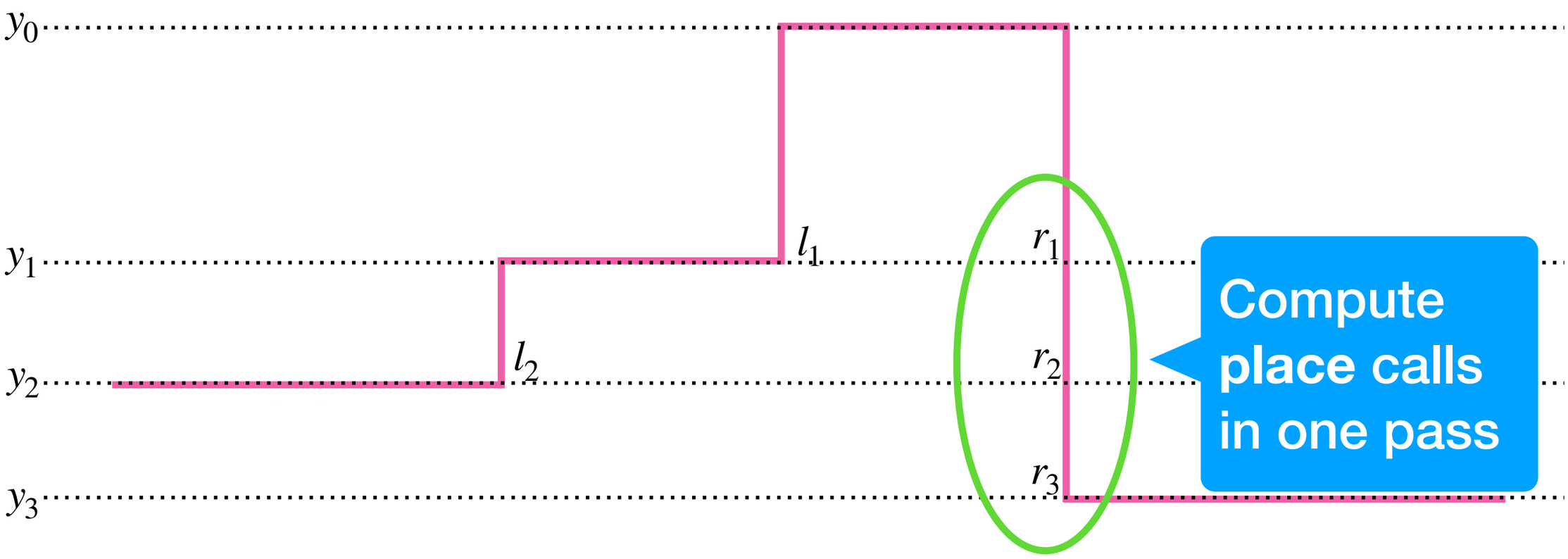
Encoding Visual Props



ezone.**add**(size, position)

ezone.**place**(size) → position

Encoding Visual Props



ezone = y_0 (y_1, l_1, r_1) (y_2, l_2, r_2) $(y_3, \perp, r_3)$

ezone.**add**(size, position)

O(n) equations

ezone.**place**(size) → position

O(n) equations

Validating the Formalization

Non-automated

Standard Tests

Browsers

Evaluate: Differential testing vs browsers

Pass thousands of conformance tests

Found bugs in existing browsers

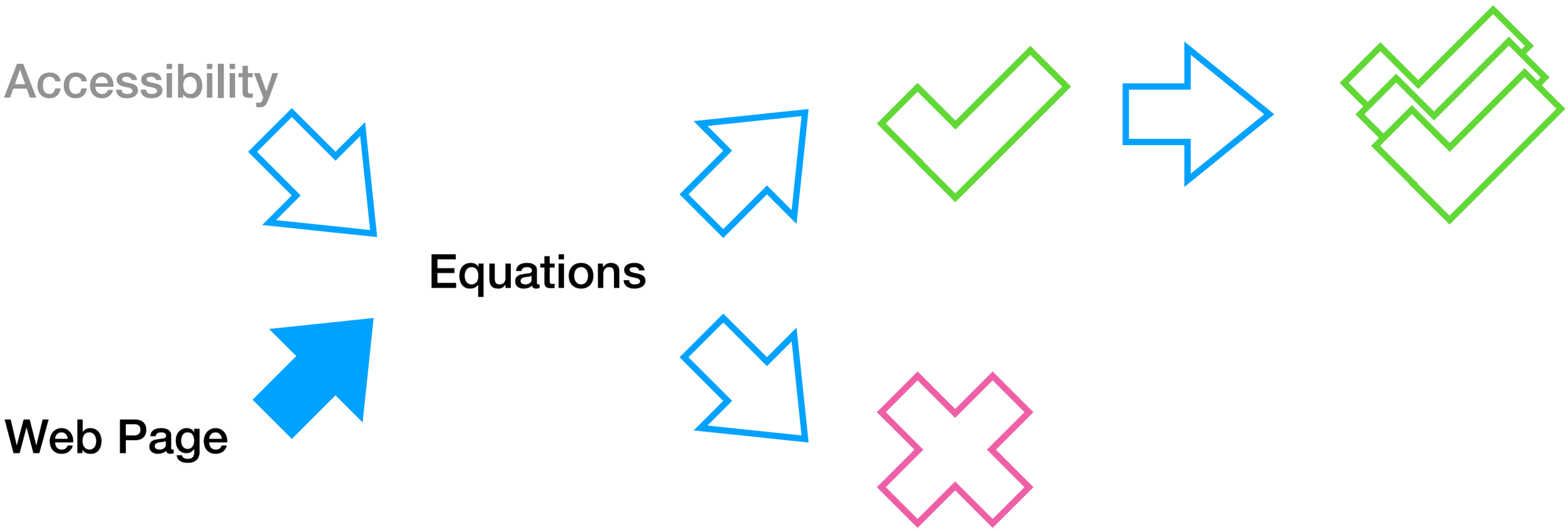
[OOPSLA'16]

How It Works

Formalize

Solve

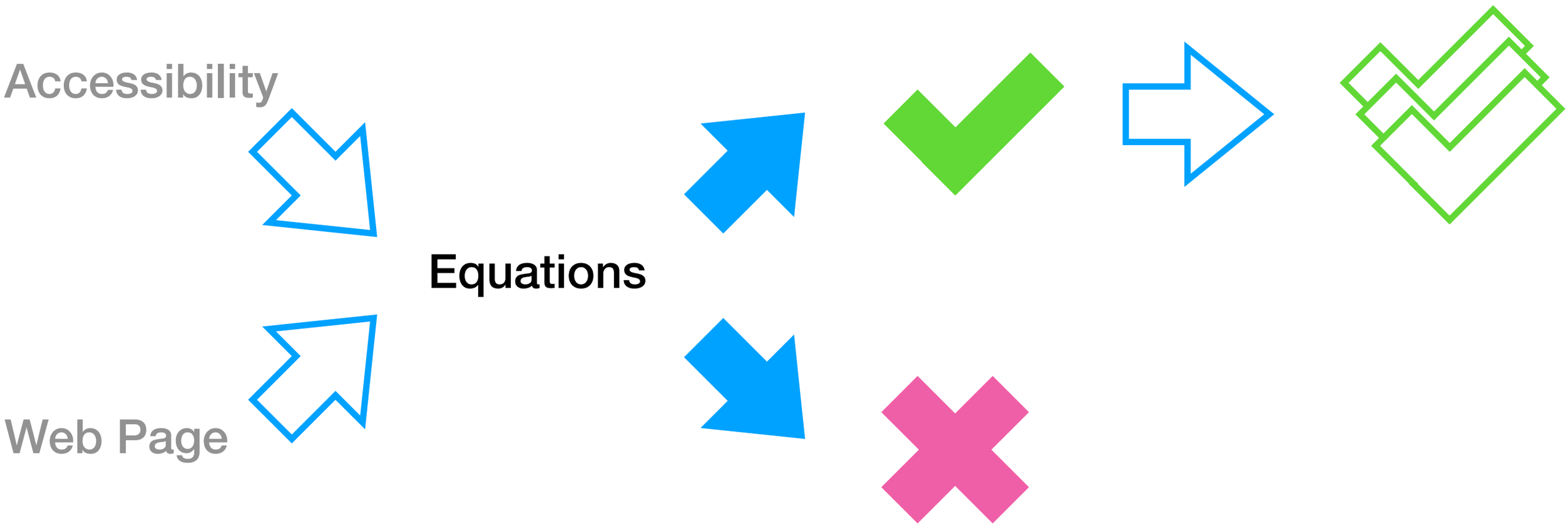
Scale Up



Formalizing visual properties

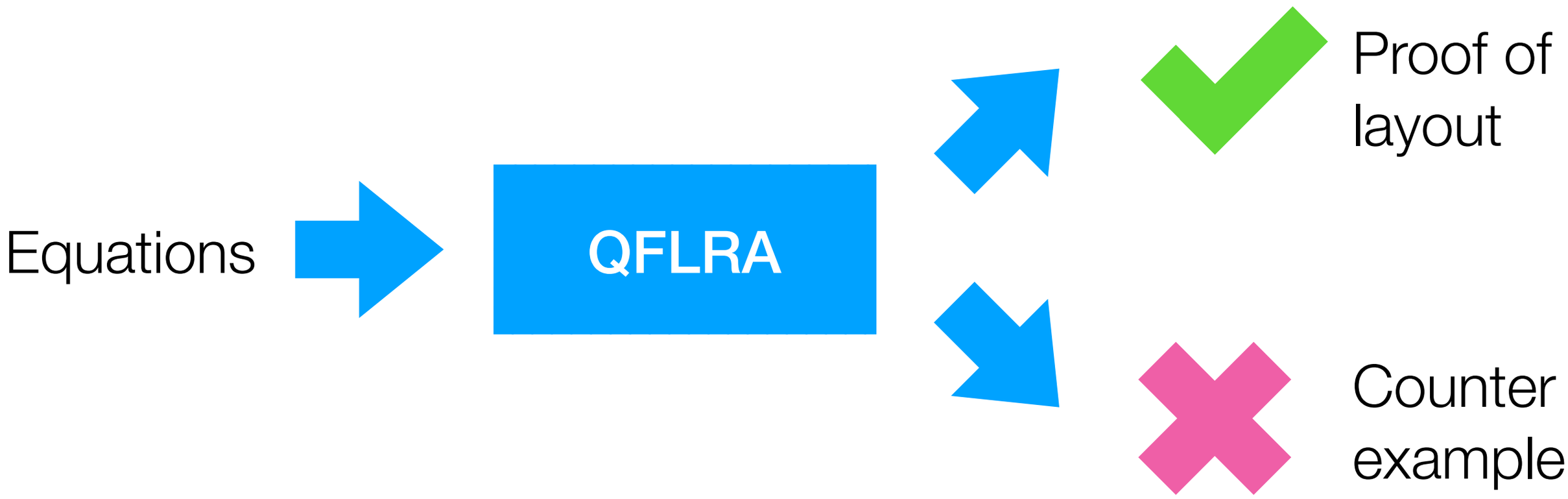
How It Works

Formalize | Solve | Scale Up



Formalizing visual properties

Solving the Equations

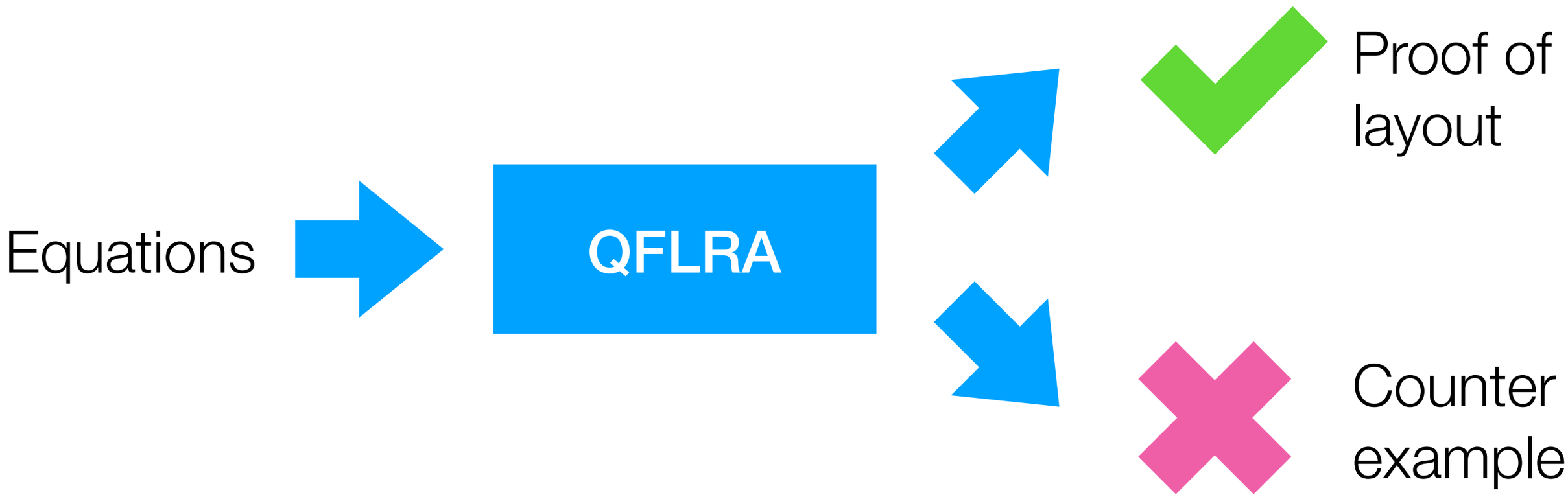


Quantifier **F**ree ⟶ No nested loops

Linear ⟶ No multiplication

Real **A**rithmetic ⟶ No rounding error

Solving the Equations



Simulated Rounding Error

$a = b \longrightarrow -\varepsilon < a - b < \varepsilon$

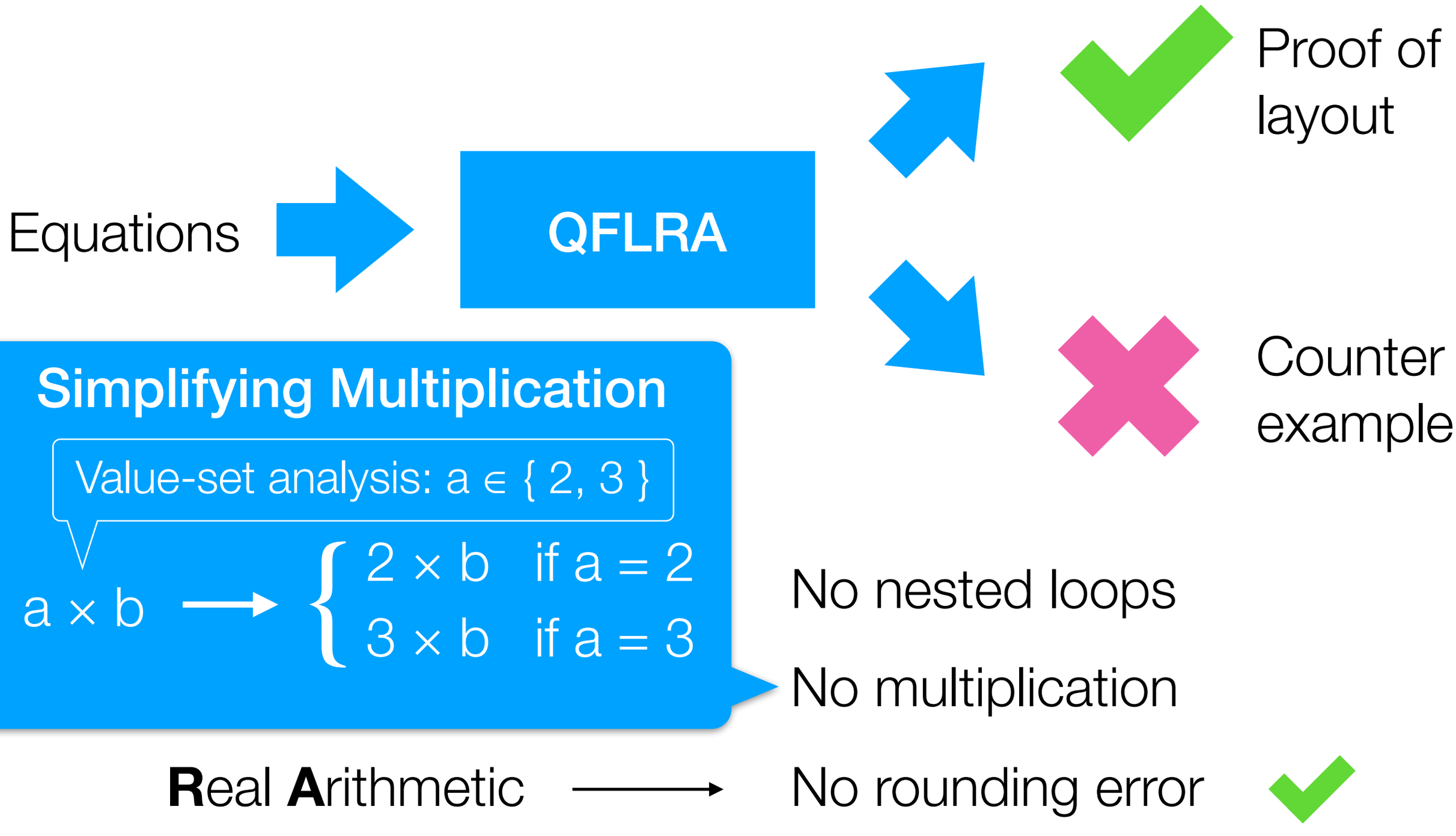
Real **A**rithmetic ⟶

No nested loops

No multiplication

No rounding error

Solving the Equations



Solving the Equations

Equations → QFLRA

QFLRA → Proof of layout

QFLRA → Counter example

Bounded work per element

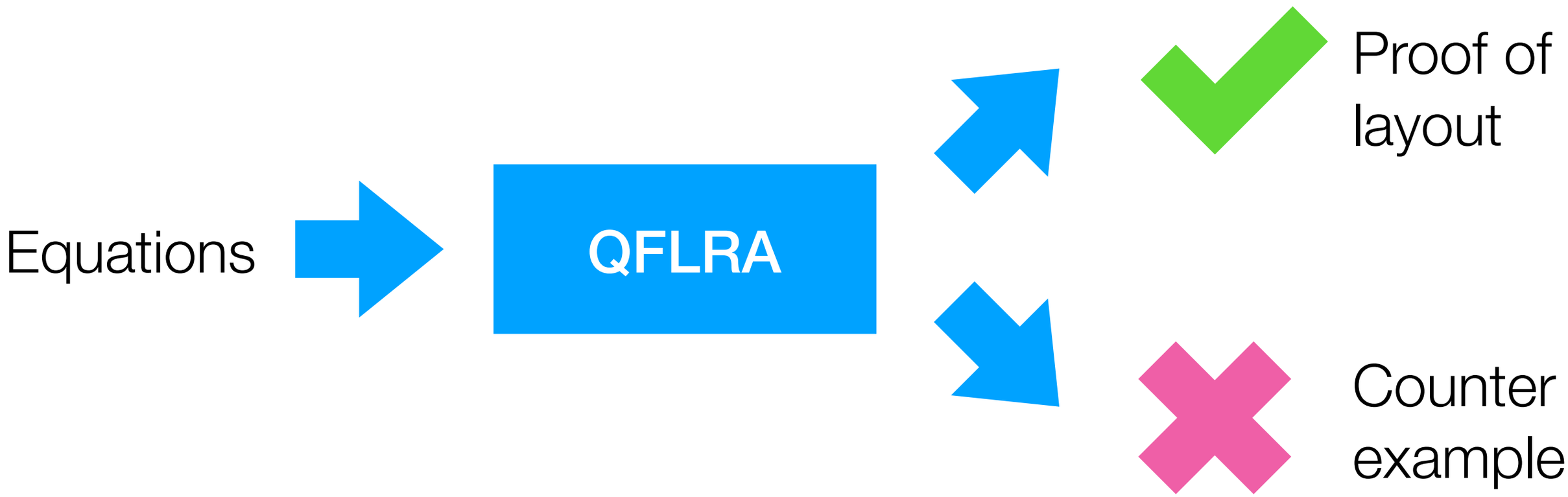
Only one loop: render each element

Quantifier **F**ree ⟶ No nested loops

Linear ⟶ No multiplication ✓

Real **A**rithmetic ⟶ No rounding error ✓

Solving the Equations

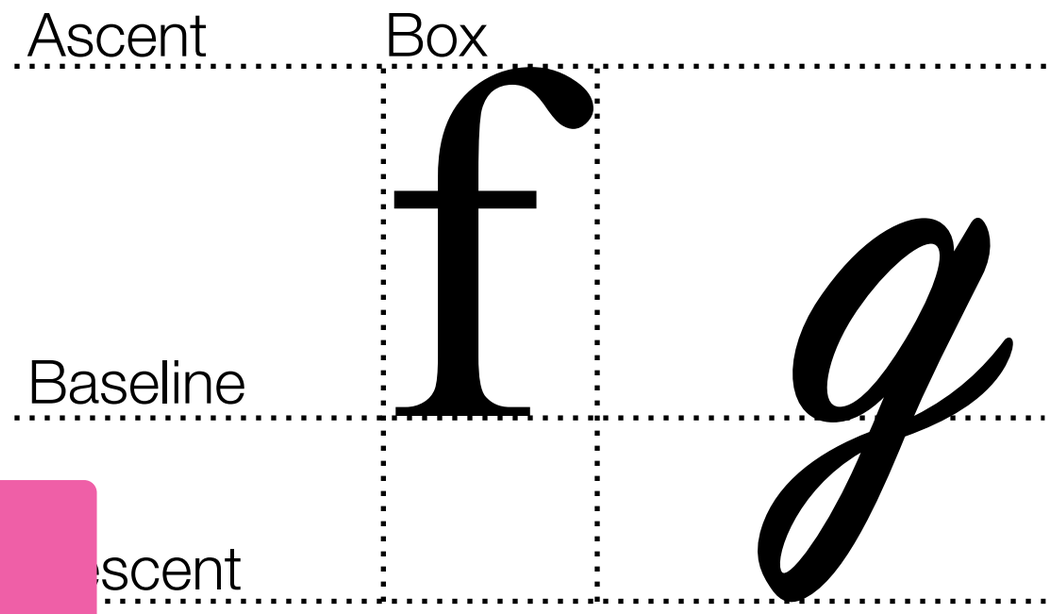


Problem: bounded work per element

Incrementalization, fusion, and unrolling

Inspired by compiler optimizations

Computing Line Height

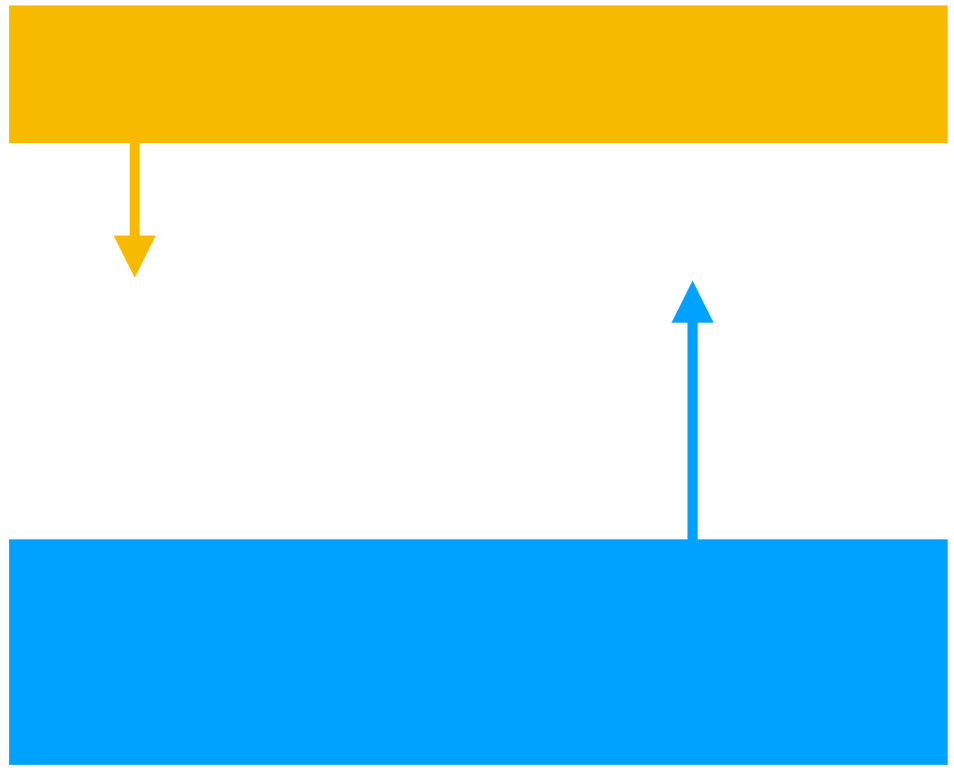


Loop over elements in line

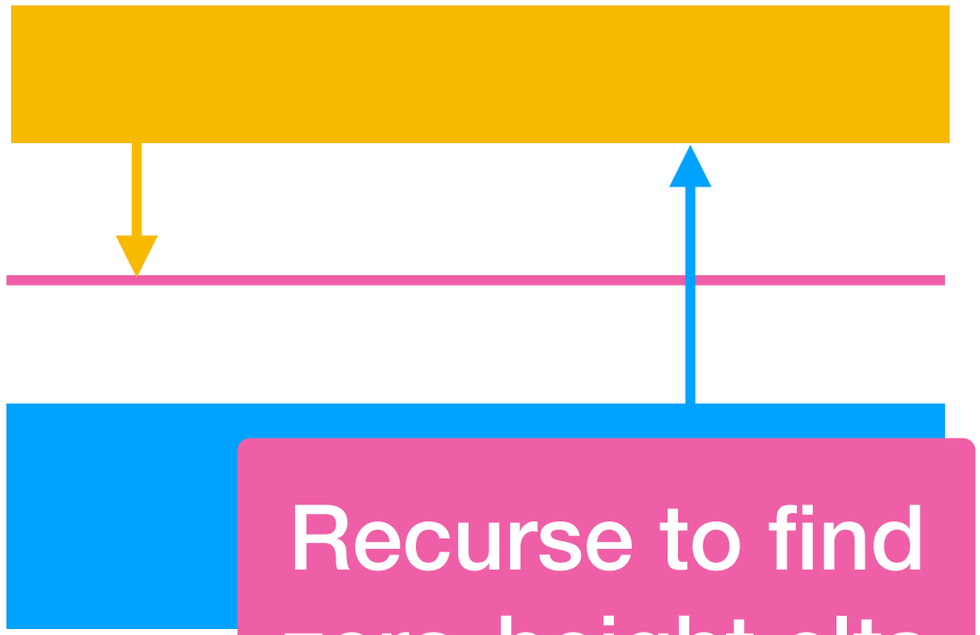
"The line box height is the distance between the **uppermost** box top and the **lowermost** box bottom"

Incrementalization: update running maximum

Computing Margins



Computing Margins



Recurse to find zero-height elts

"Adjoining vertical margins **collapse**;
two margins are **adjoining** if and only if …"

Fusion: interleave with outer render loop

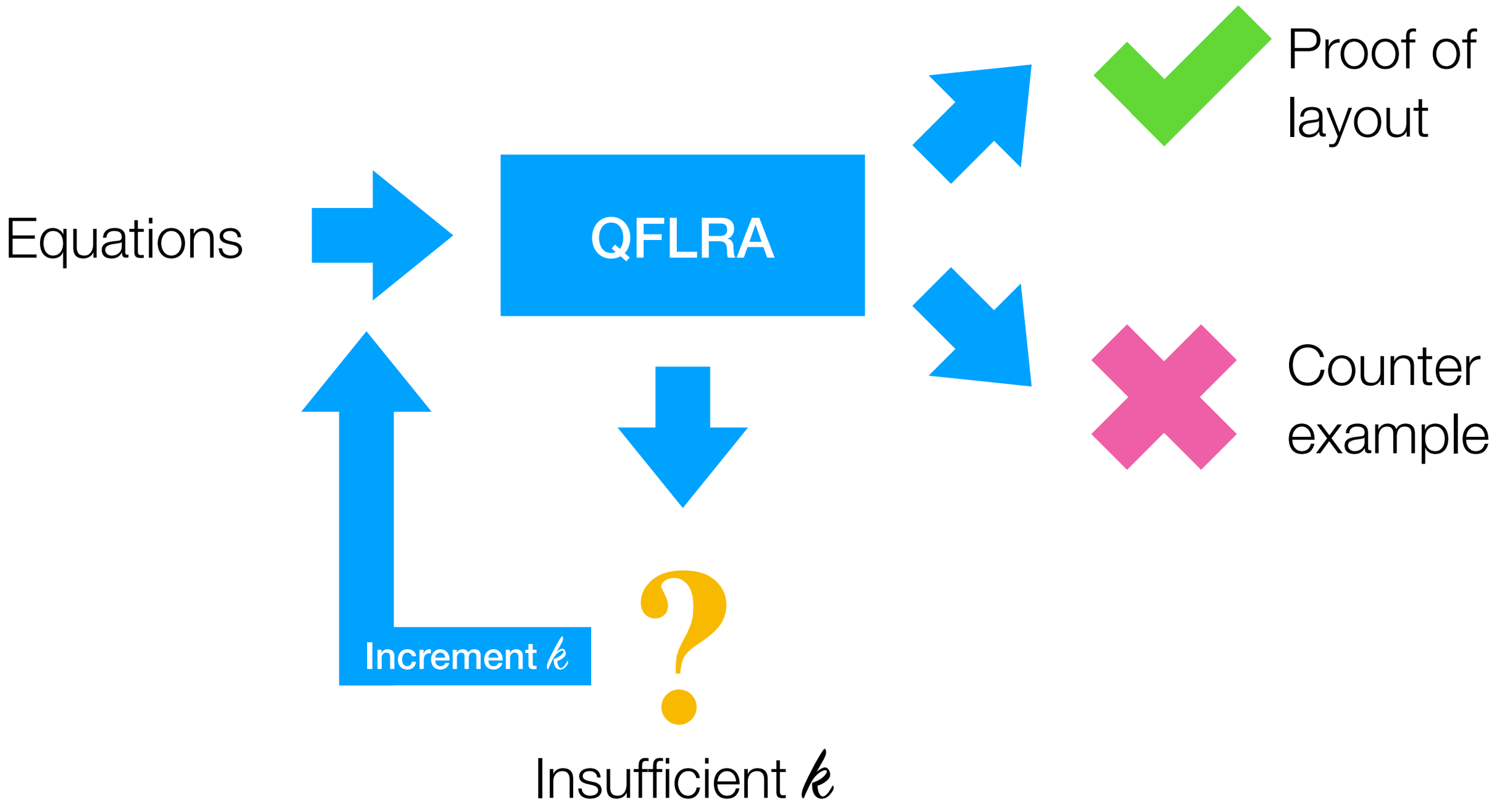
Computing Float Layout

Text wraps around floats on either side of it.

Arbitrary-size data structure

Unroll up to k corners

Solving the Equations



Verifying Real Pages

Webs lates.com

Only use formalized subset of CSS

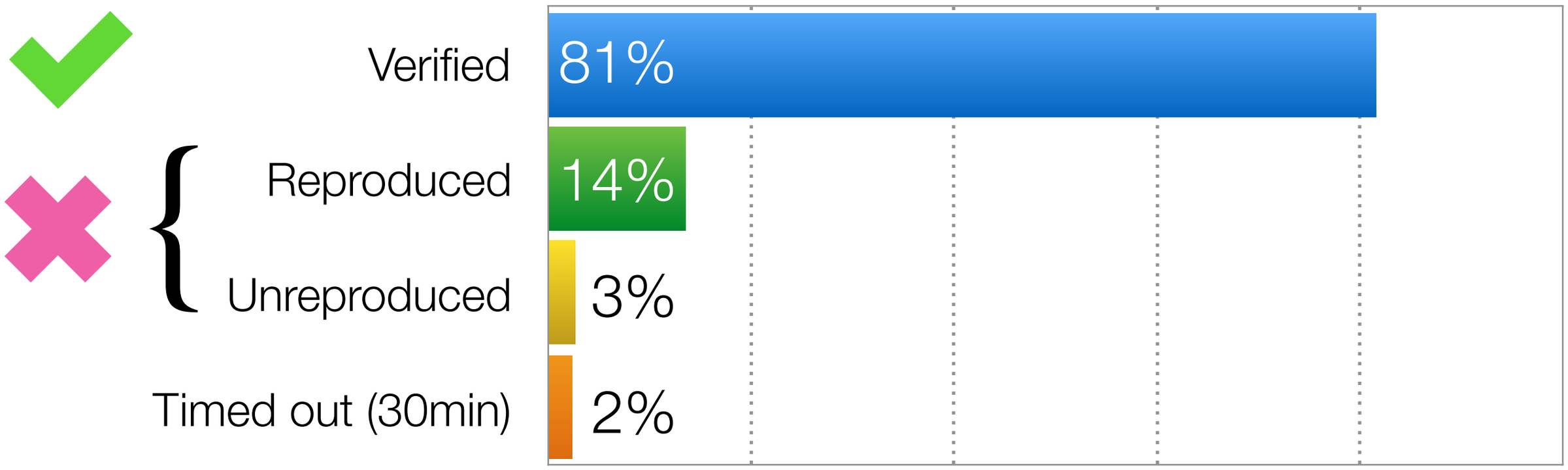
62 web pages

14 properties

} **476** sensible combinations

From Apple, Google, DOJ accessibility guides

Verifying Real Pages



Evaluate: verified majority of real-world inputs

Found many real accessibility bugs

Few false positives and few timeouts

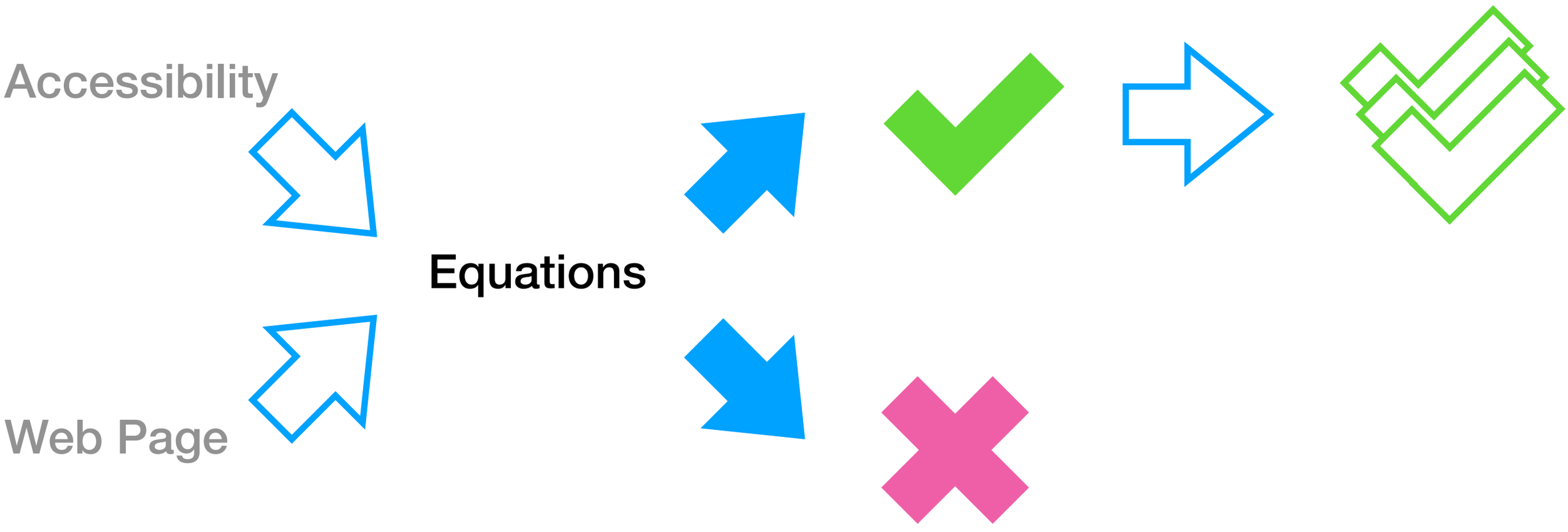
[PLDI'18]

How It Works

Formalize

Solve

Scale Up



Formalizing visual properties

Overcoming solver limitations

How It Works

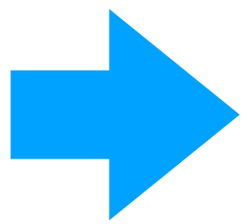
Formalize

Solve

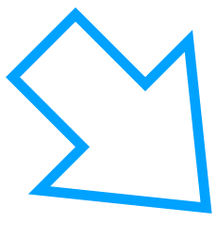
Scale Up

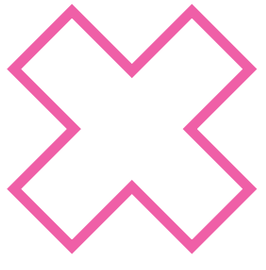
Accessibility

Equations



Web Page

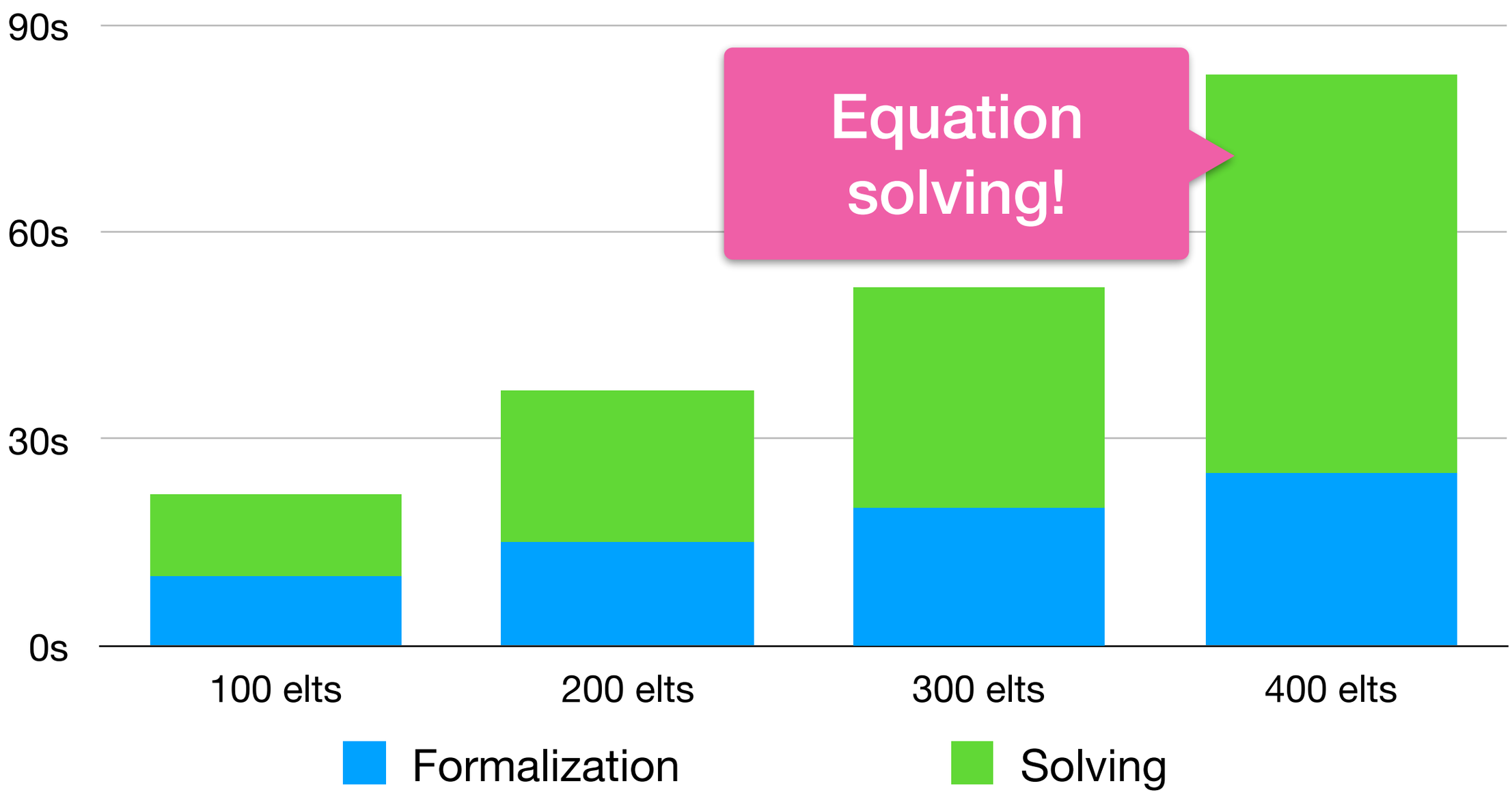




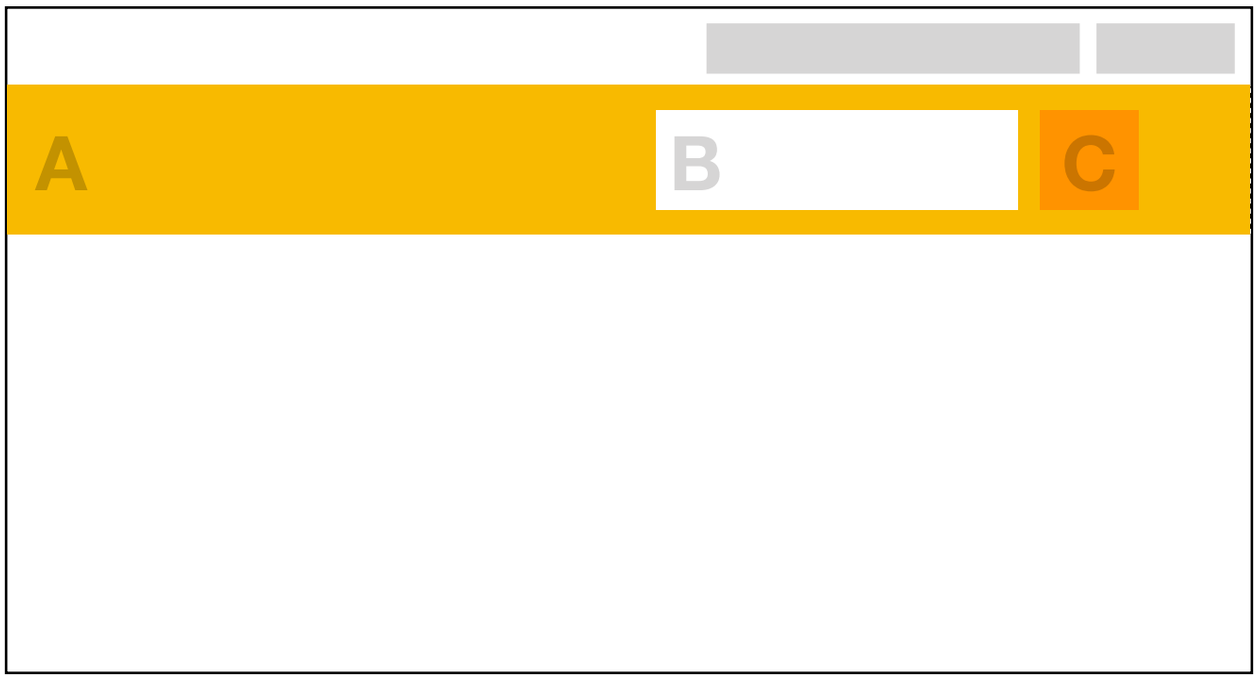
Formalizing visual properties

Overcoming solver limitations

Scaling Verification

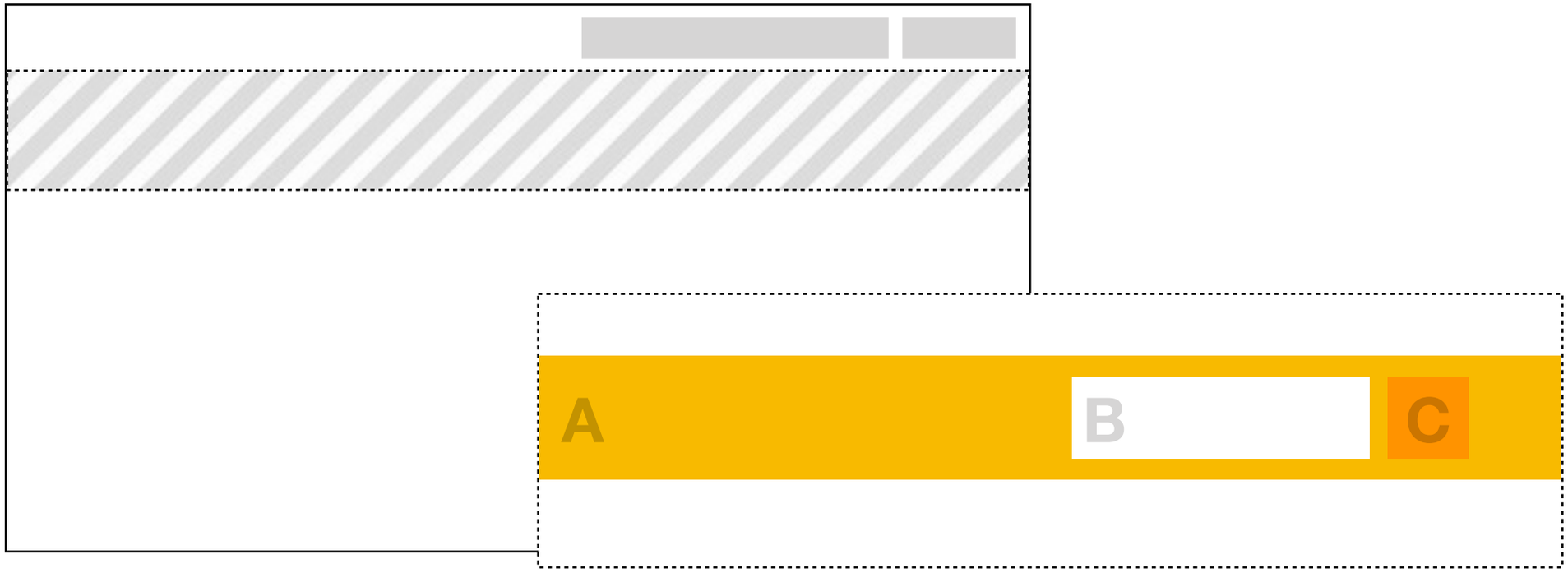


Scaling Verification



Problem: reason about large pages quickly

Scaling Verification

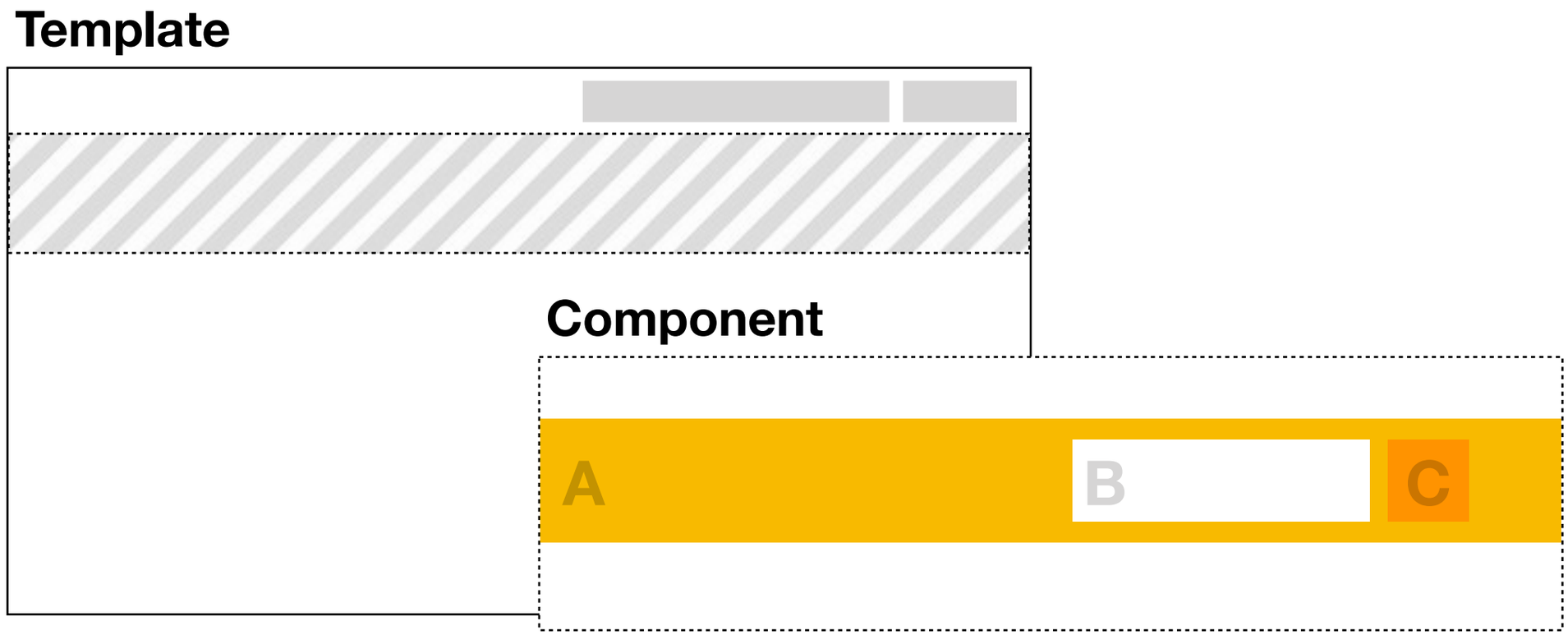


Problem: reason about large pages quickly

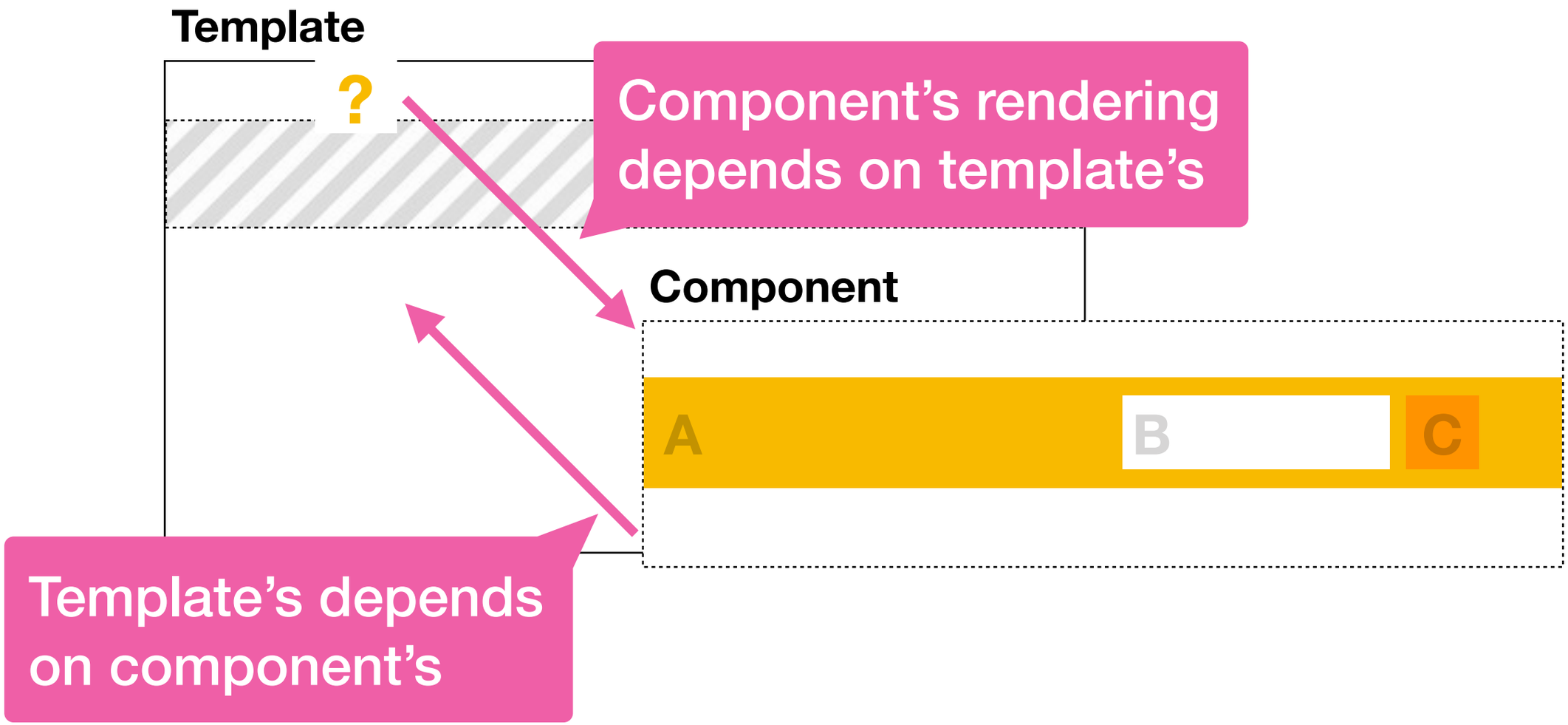
Divide web page into small components

Combine components with rely/guarantee logic

Isolating Components

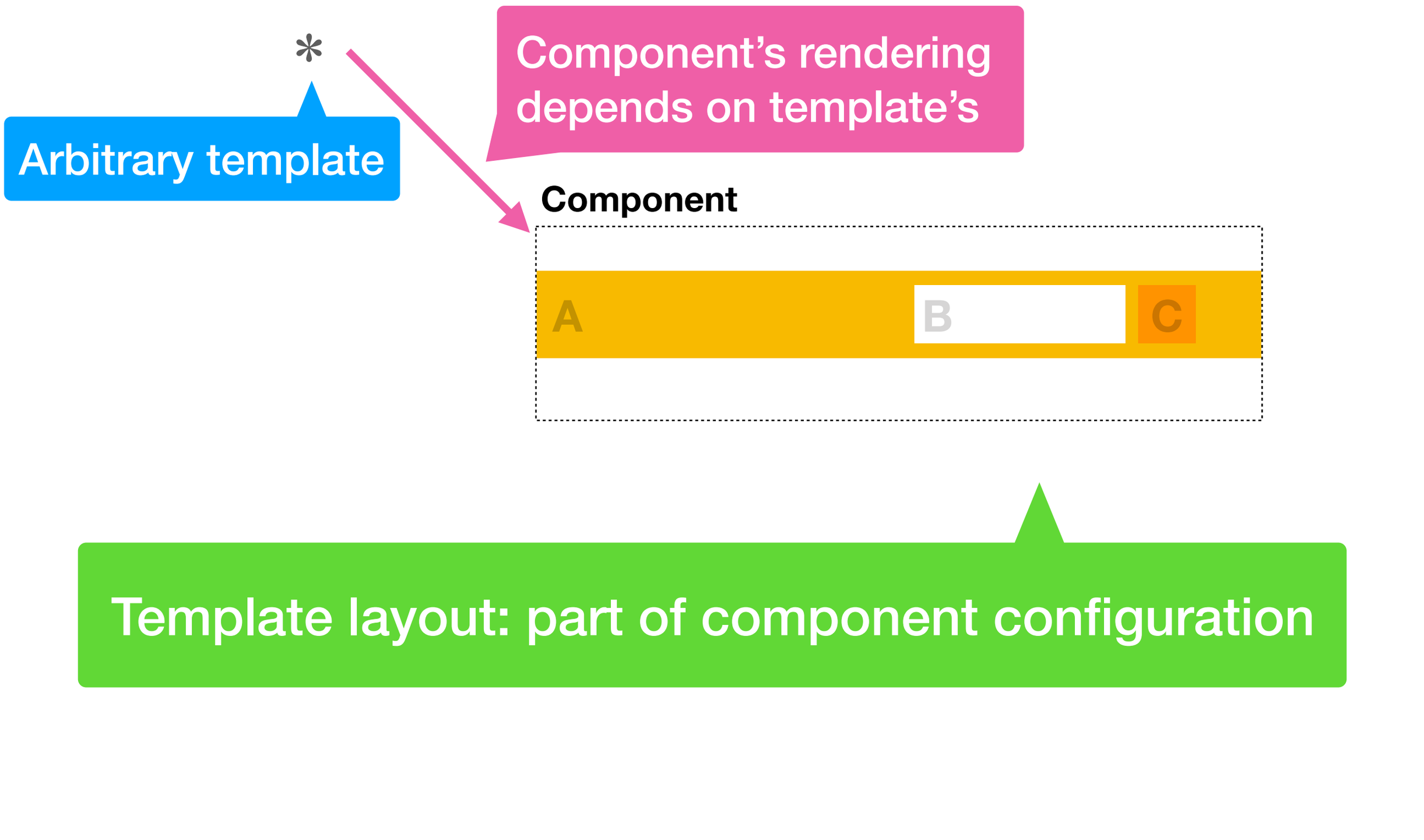


Isolating Components



No module or function boundaries!

Isolating Components

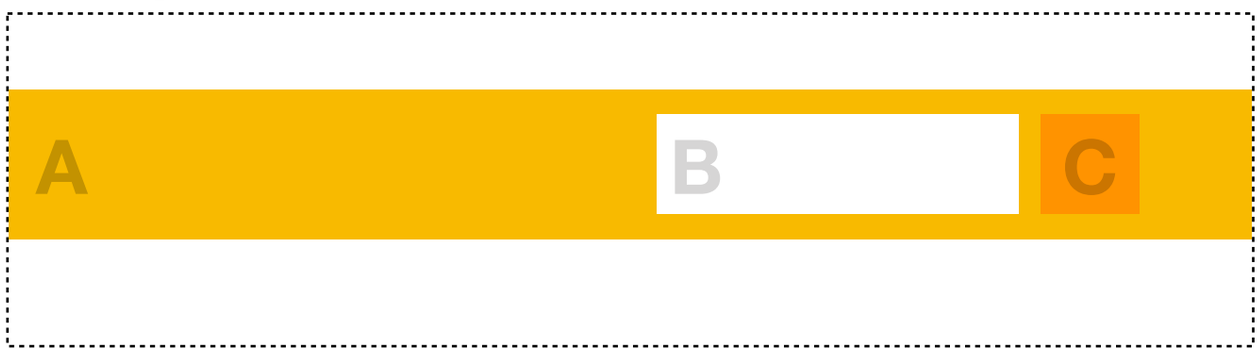


Isolating Components

Precondition

Width available
Current font size
Floating elements

Component's rendering depends on template's



Template layout: part of component configuration

Rely / Guarantee

Component

Precondition ⇒ Specification

Too abstract
- Callout?
- HC.gov example?

Rely / Guarantee

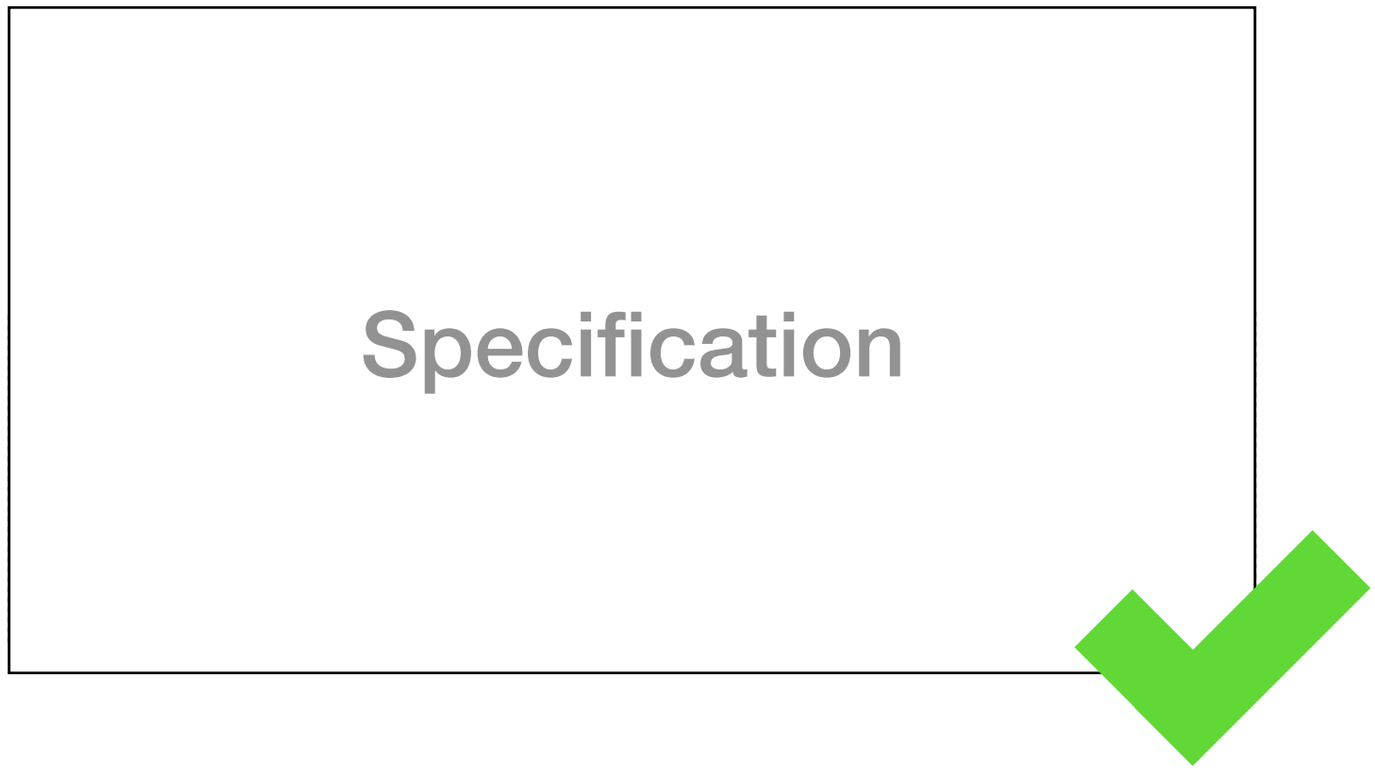
Template

Precondition

Component

Precondition ⇒ Specification

Rely / Guarantee

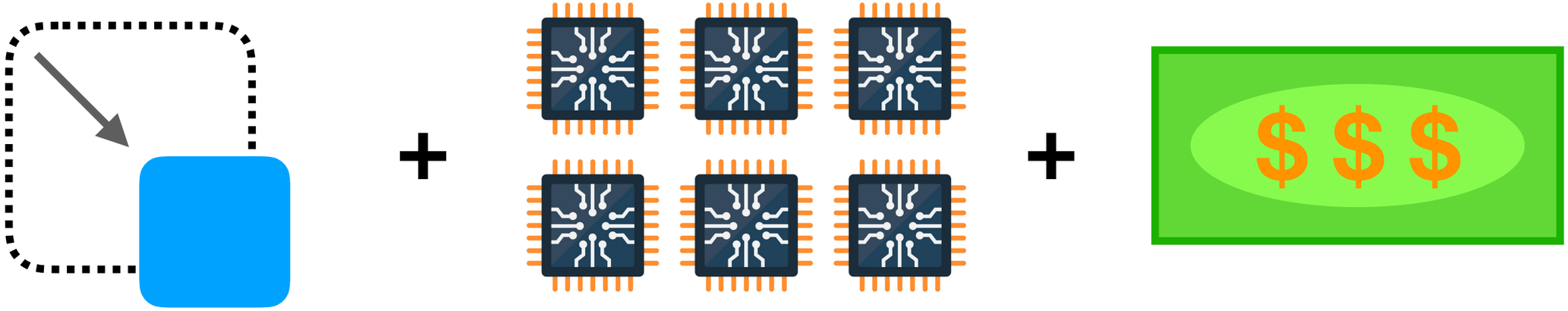


Preconditions checked purely logically

No rendering ⇒ fast

Improving Scale

Problem Size + **Parallelism** + **Caching**



Evaluate: broader class of verified pages

[OOPSLA'19]

Scaled to 11× larger pages, to 1400× faster

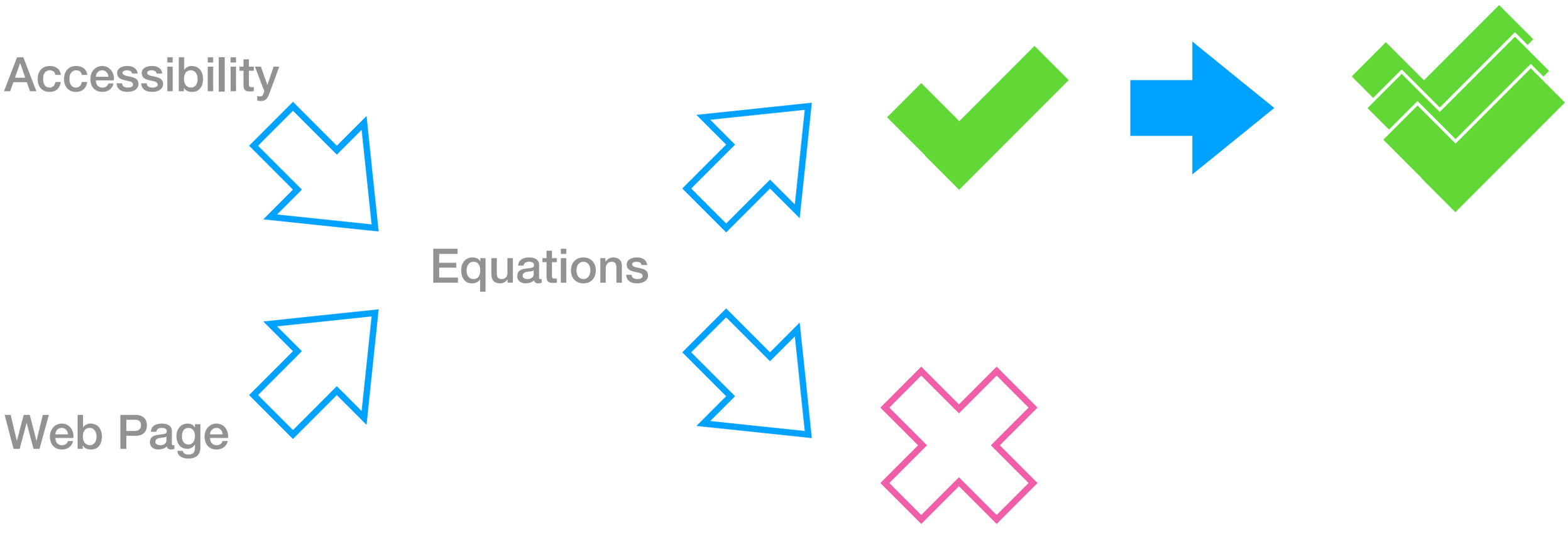
Scaled to multiple pages on one site

How It Works

Formalize

Solve

Scale Up



Formalizing visual properties

Overcoming solver limitations

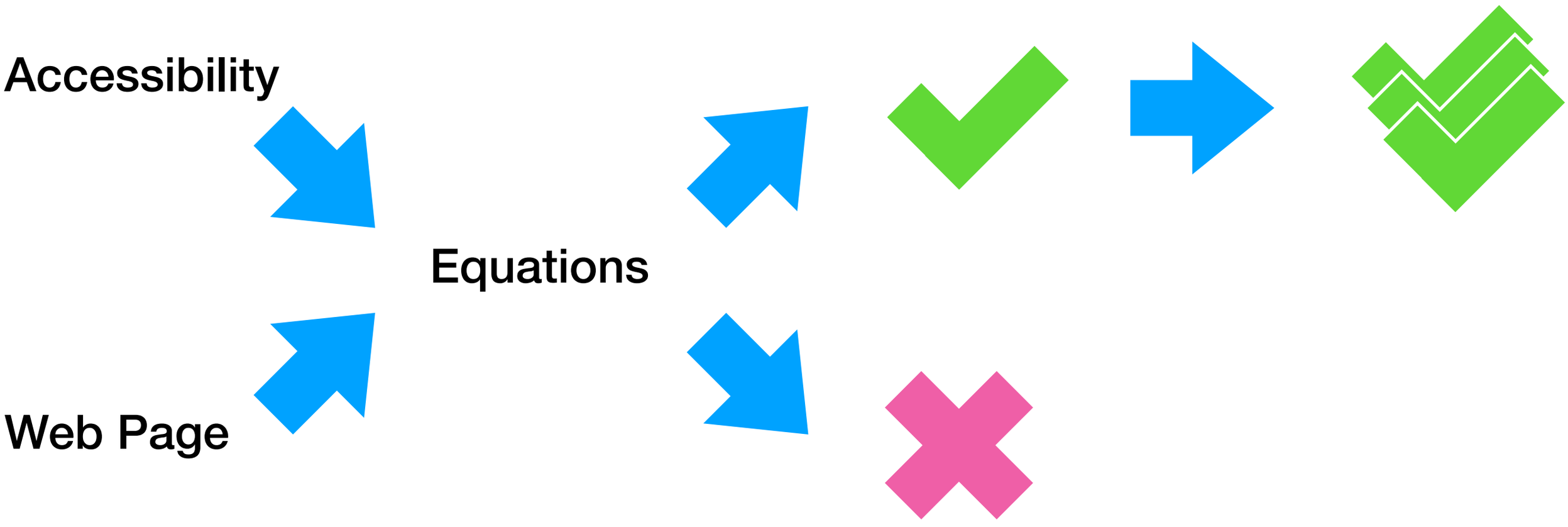
Finding modularity for rely/guarantee

How It Works

Formalize

Solve

Scale Up



Formalizing visual properties

Overcoming solver limitations

Finding modularity for rely/guarantee

Where We're Going

From Logic to Programs

Three Parts

Write a specification

Propagate into program

Scale to systems

Program Reasoning

How to **define / write** a programming language

Syntax, two types of semantics, and interpretation

Symbolic execution of program expressions

Dealing with function calls, conditionals, and feasibility

Changes in state due to program statements

Hoare logic, invariants, and termination

Next class:

Symbolic Execution

To do:

- ☐ Course feedback
- ☐ Project Proposal