Artifacts Available acm V1.1

Artifacts Evaluated acm Reusable V1.1

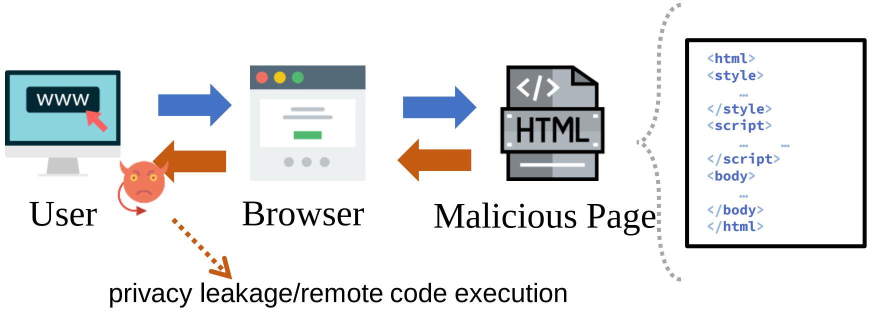
Towards Better Semantics Exploration for Browser Fuzzing

Chijin Zhou[1], Quan Zhang[1], Lihua Guo[1], Mingzhe Wang[1], Yu Jiang[1], Qing Liao[2], Zhiyong Wu[3], Shanshan Li[3] and Bin Gu[4]

[1]Tsinghua University, China
[2]Harbin Institute of Technology, China
[3]National University of Defense Technology, China
[4]Beijing Institute of Control Engineering, China

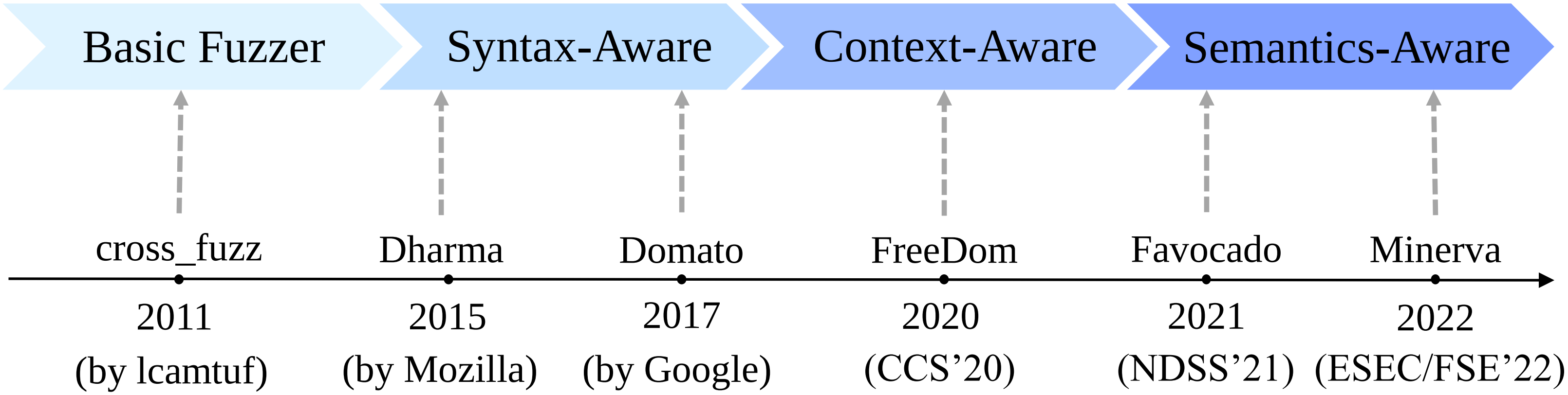
Browser Fuzzing: A Decade of Research

If a browser is vulnerable …



```
<html>
<style>
…
</style>
<script>
…       …
</script>
<body>
…
</body>
</html>
```

Goal of fuzzers: Generate html files that explore **browser states** and, with luck, trigger **bugs**.



Basic Fuzzer → Syntax-Aware → Context-Aware → Semantics-Aware

cross_fuzz, 2011 (by lcamtuf)

Dharma, 2015 (by Mozilla)

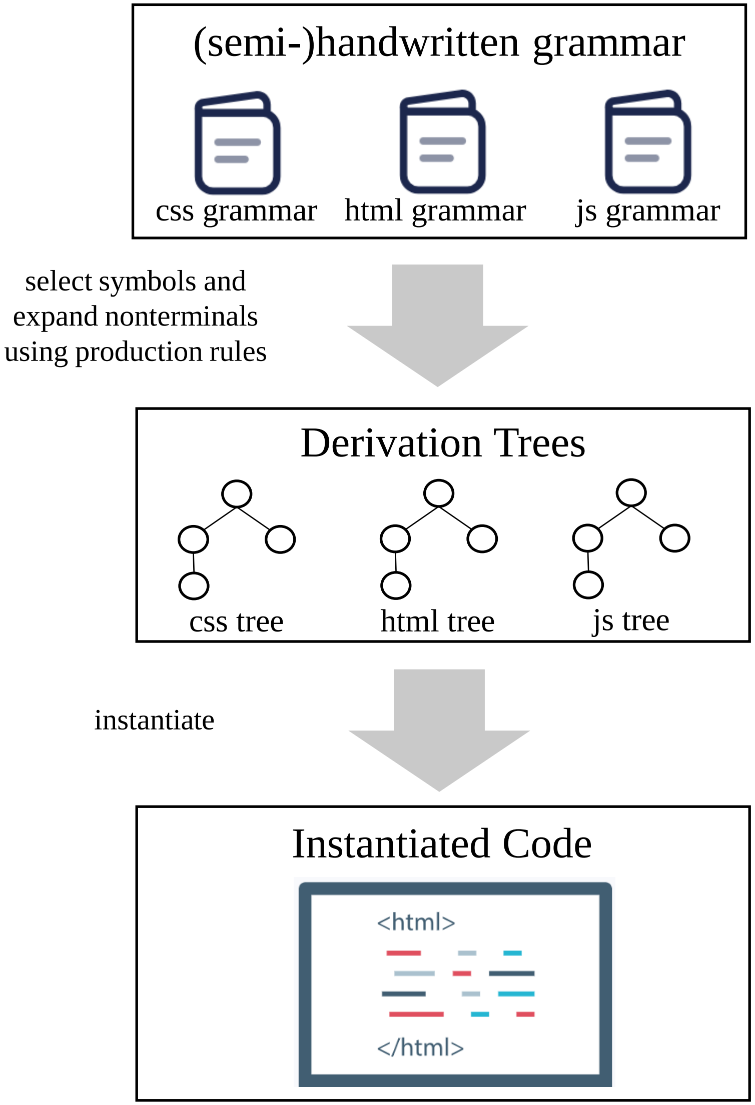
Domato, 2017 (by Google)

FreeDom, 2020 (CCS’20)

Favocado, 2021 (NDSS’21)

Minerva, 2022 (ESEC/FSE’22)

How did browser fuzzers work?



(semi-)handwritten grammar

css grammar html grammar js grammar

select symbols and expand nonterminals using production rules

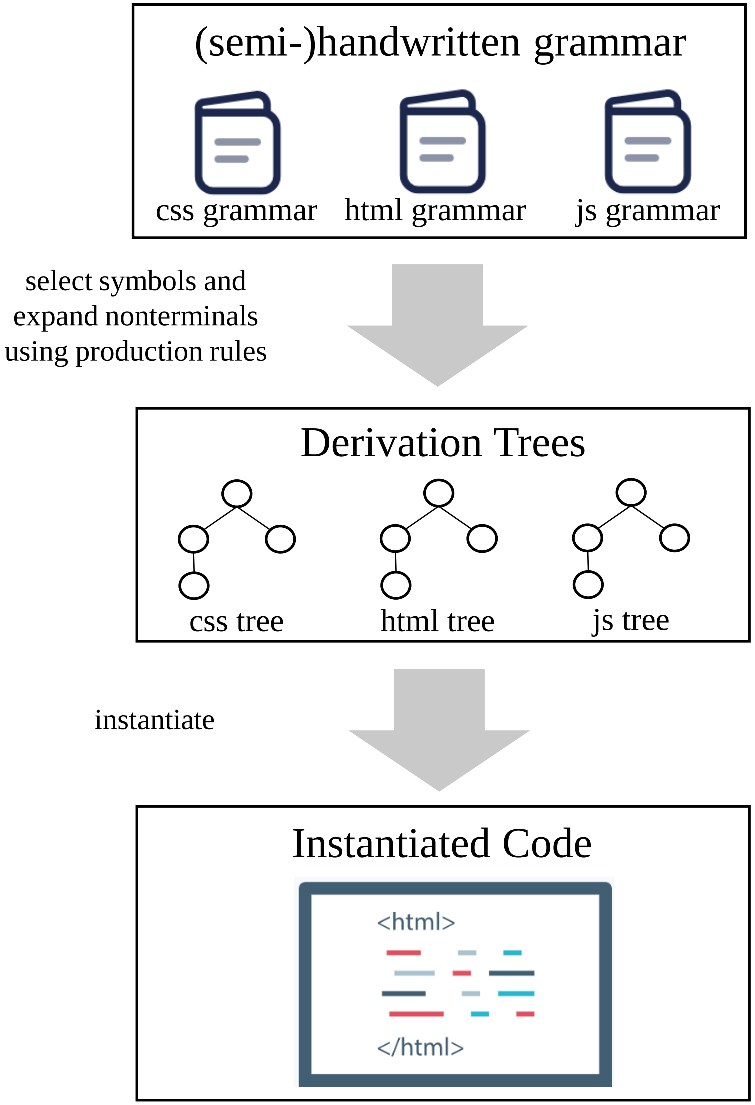
Derivation Trees

css tree html tree js tree

instantiate

Instantiated Code

How did browser fuzzers work?



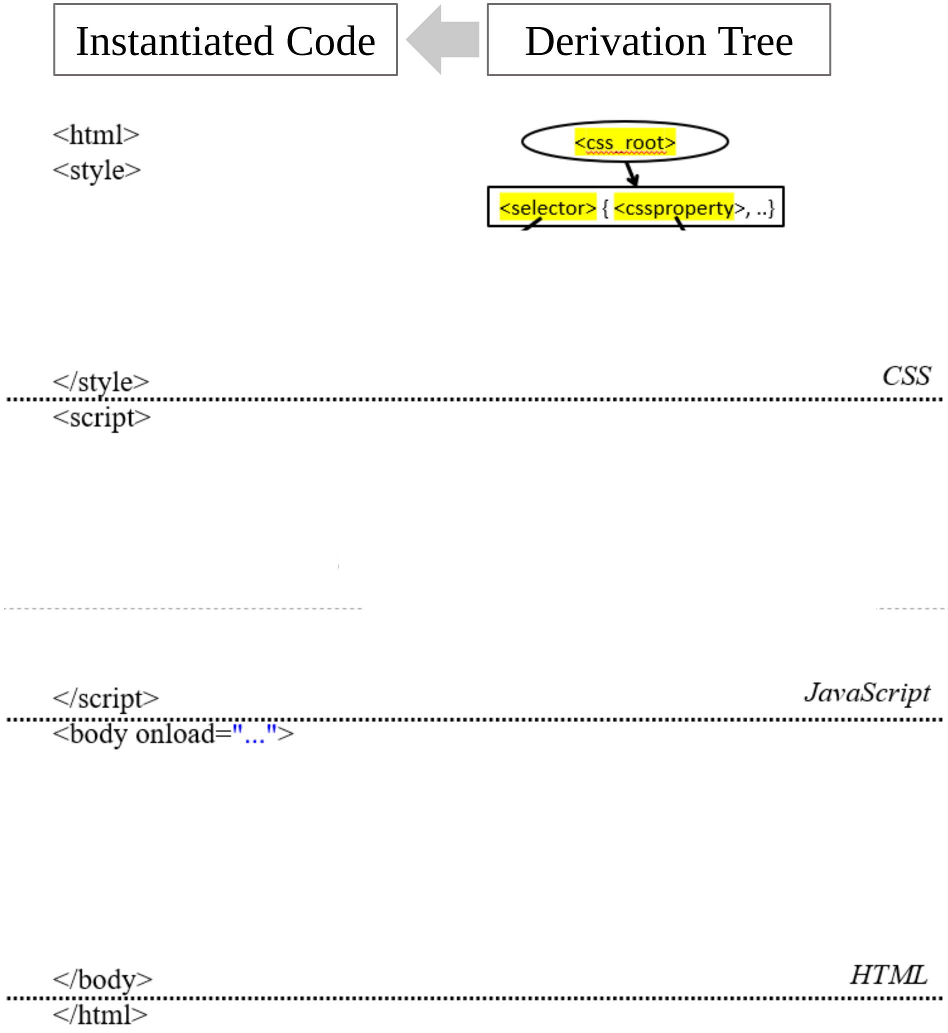
(semi-)handwritten grammar

select symbols and expand nonterminals using production rules

Derivation Trees

instantiate

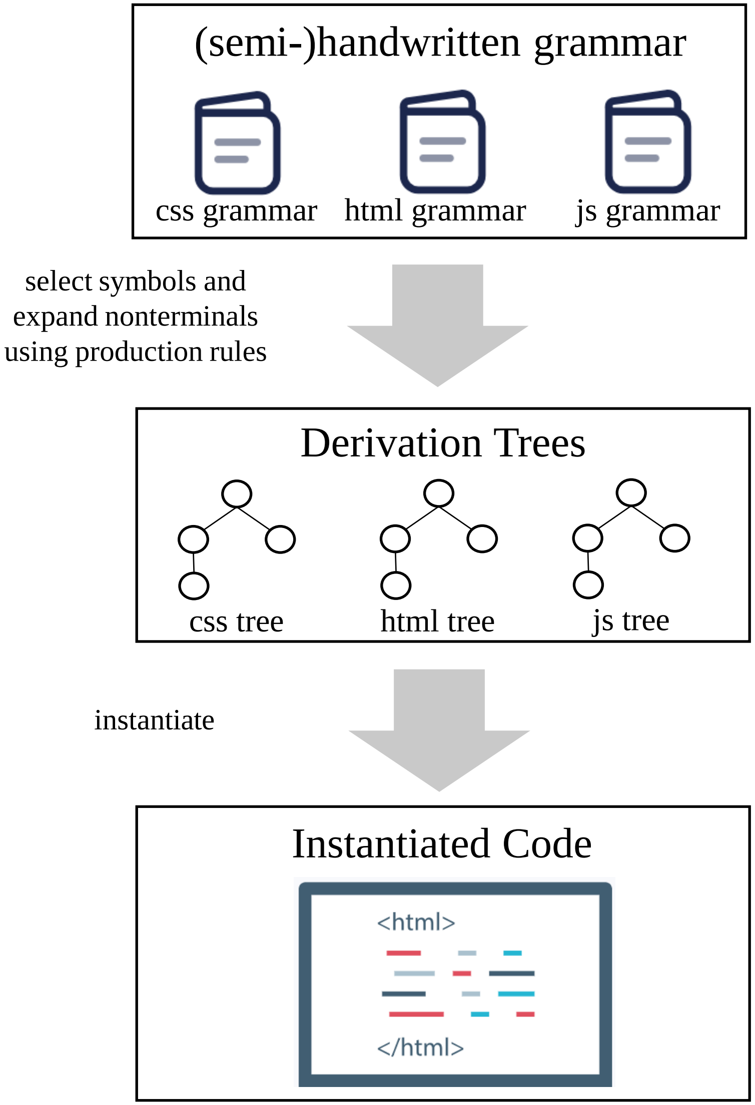
Instantiated Code



Instantiated Code ← Derivation Tree

```
<html>
<style>

</style>
<script>

</script>
<body onload="...">

</body>
</html>
```

How did browser fuzzers work?



(semi-)handwritten grammar

css grammar

html grammar

js grammar

select symbols and expand nonterminals using production rules

Derivation Trees

css tree

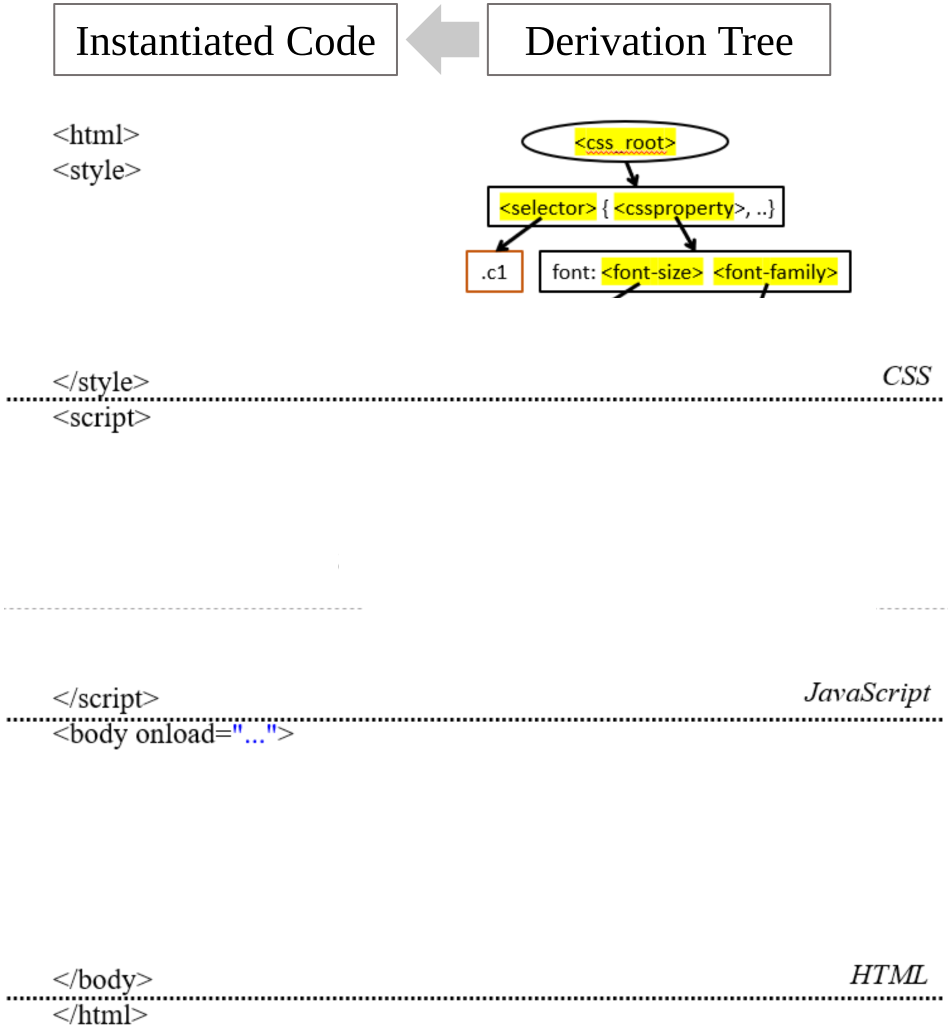
html tree

js tree

instantiate

Instantiated Code

```
<html>
</html>
```



Instantiated Code

Derivation Tree

<css_root>

<selector> { <cssproperty>, ..}

.c1

font: <font-size> <font-family>

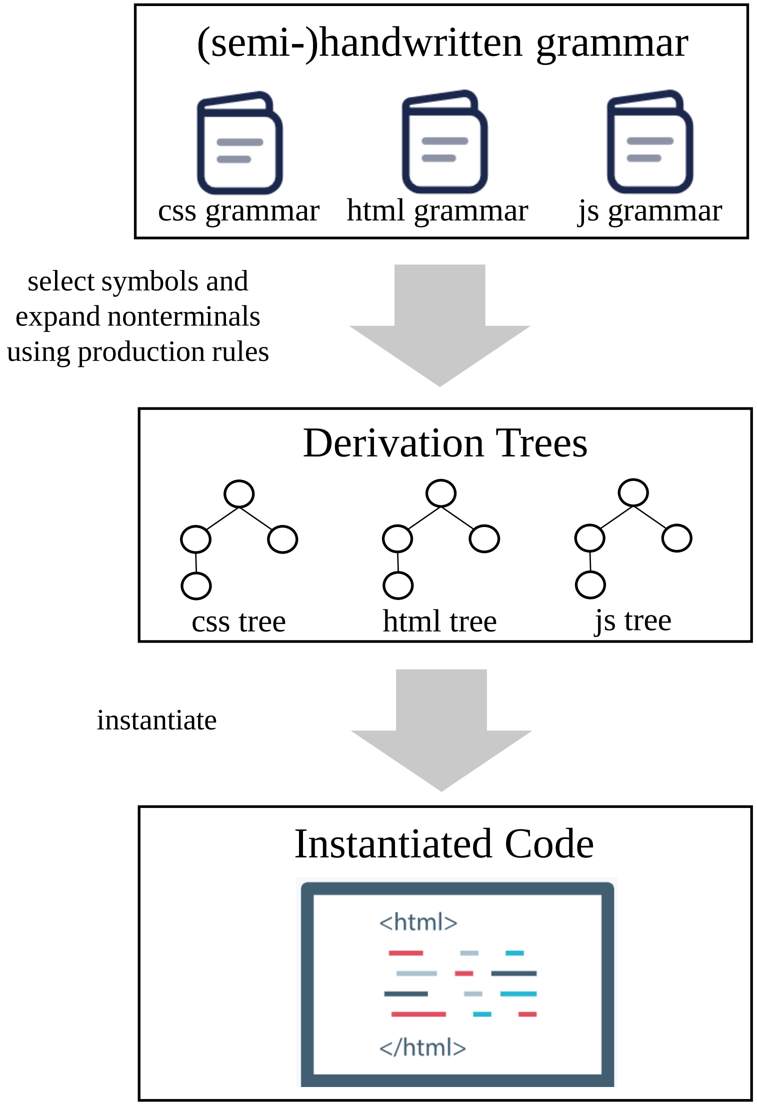
```
<html>
<style>

</style>
<script>

</script>
<body onload="...">

</body>
</html>
```

CSS

JavaScript

HTML

How did browser fuzzers work?



(semi-)handwritten grammar

css grammar

html grammar

js grammar

select symbols and expand nonterminals using production rules

Derivation Trees

css tree

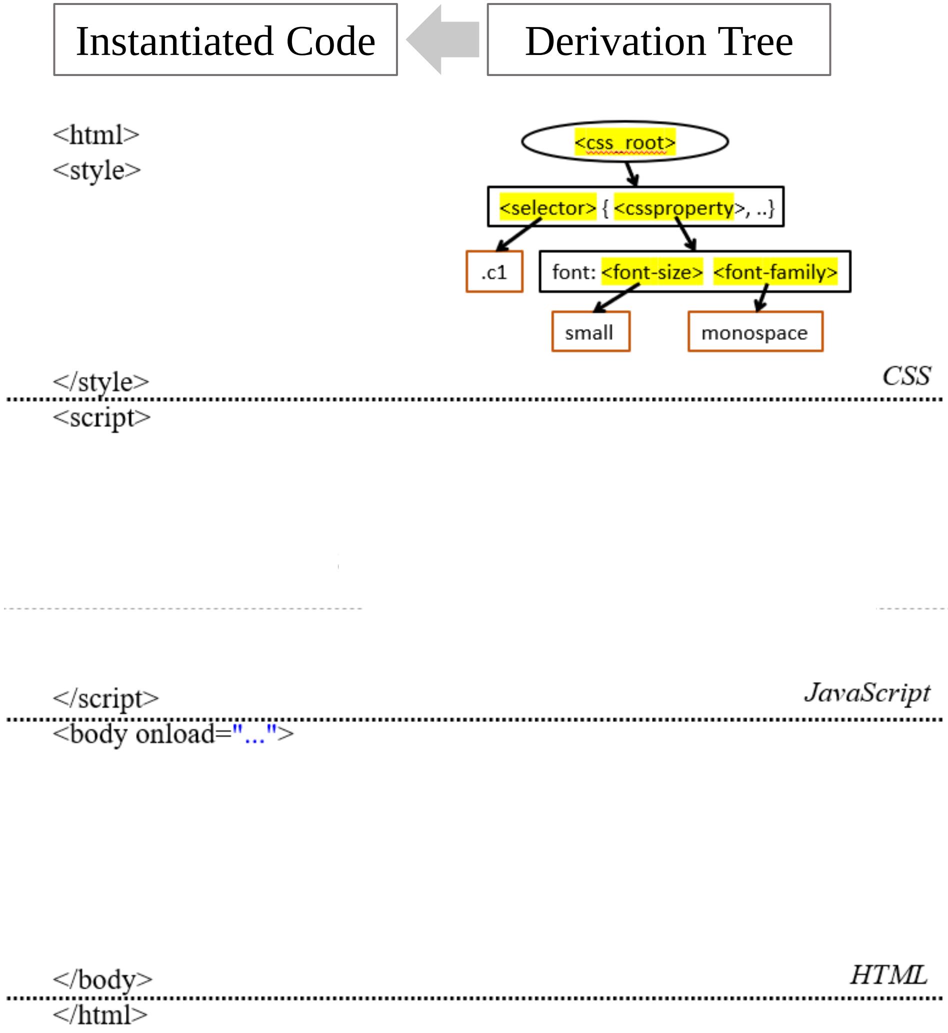
html tree

js tree

instantiate

Instantiated Code

```
<html>
</html>
```



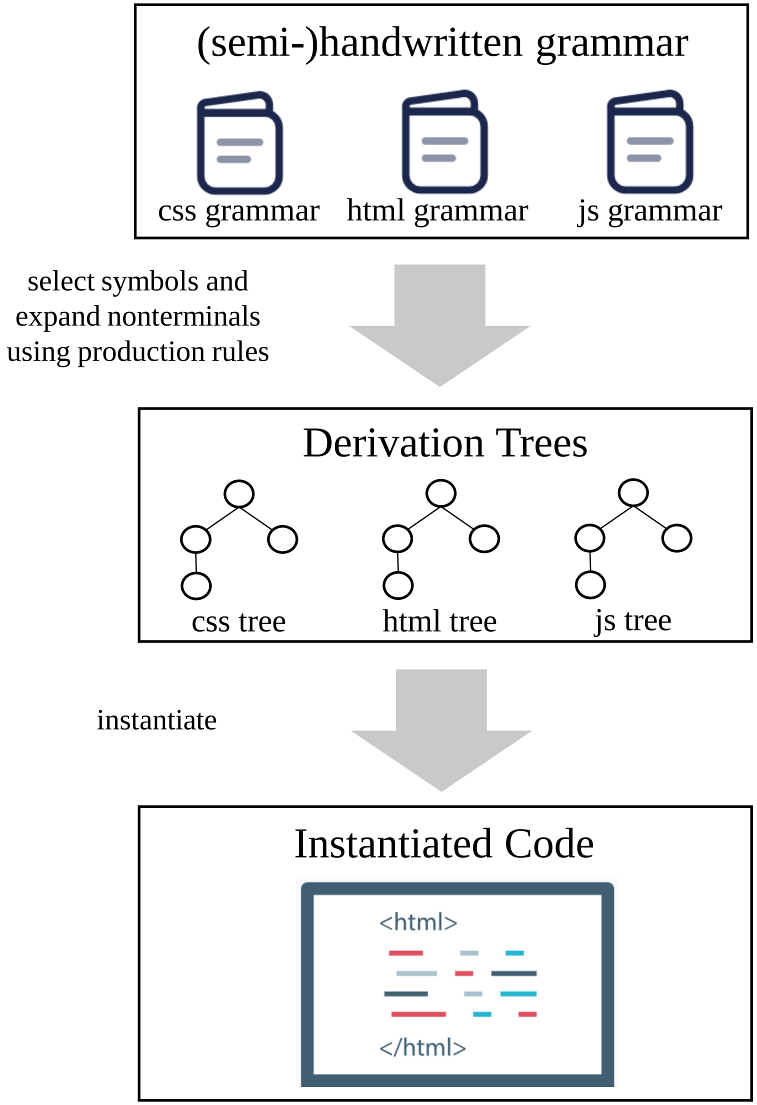
Instantiated Code

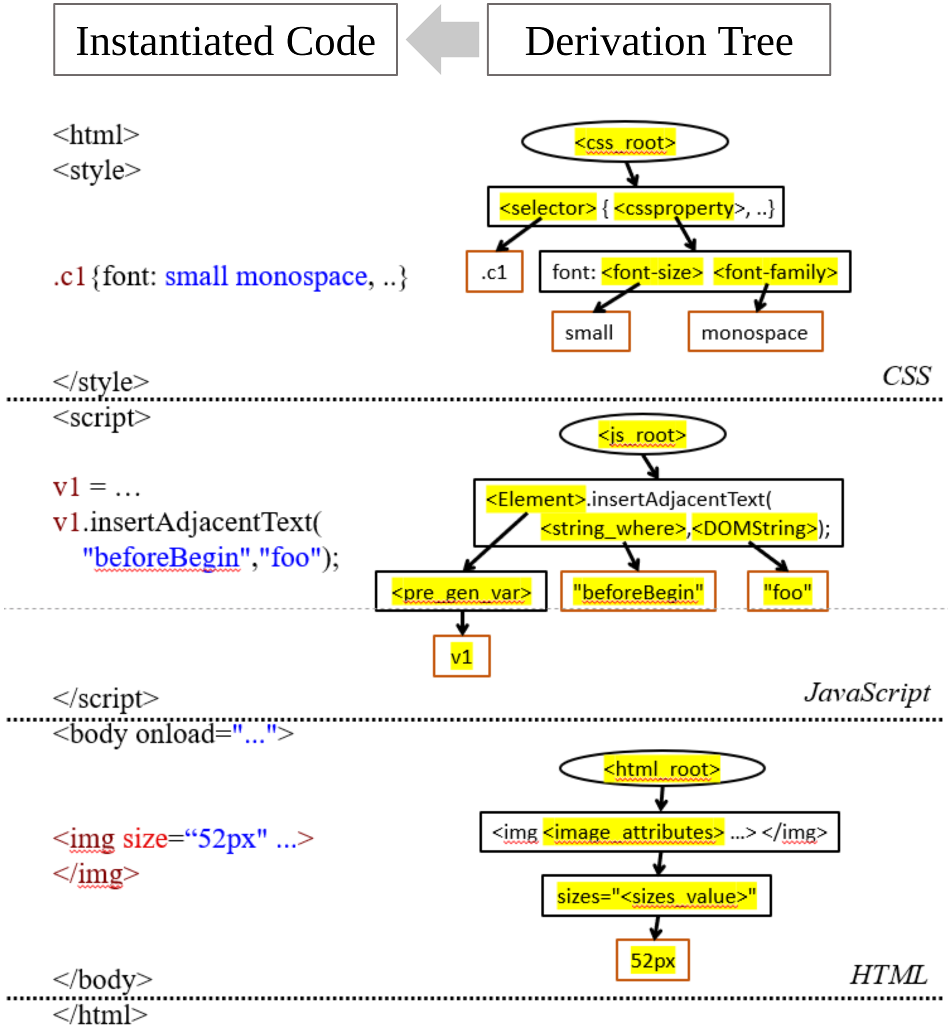
Derivation Tree

```
<html>
<style>
</style>
<script>
</script>
<body onload="...">
</body>
</html>
```

<css_root>
<selector> { <cssproperty>, ..}
.c1
font: <font-size> <font-family>
small
monospace
CSS
JavaScript
HTML

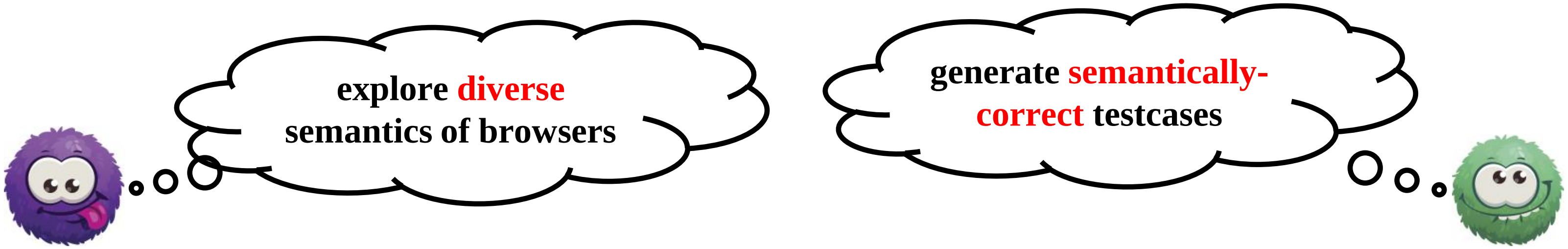
How did browser fuzzers work?




Instantiated Code
Derivation Tree
<css_root>
<selector> { <cssproperty>, ..}
.c1
font: <font-size> <font-family>
small
monospace
CSS
<js_root>
<Element>.insertAdjacentText(
<string_where>,<DOMString>);
<pre_gen_var>
"beforeBegin"
"foo"
v1
JavaScript
<html_root>
<img <image_attributes> ...> </img>
sizes="<sizes_value>"
52px
HTML

```
<html>
<style>

.c1{font: small monospace, ..}

</style>
<script>

v1 = …
v1.insertAdjacentText(
   "beforeBegin","foo");

</script>
<body onload="...">

<img size=“52px" ...>
</img>

</body>
</html>
```

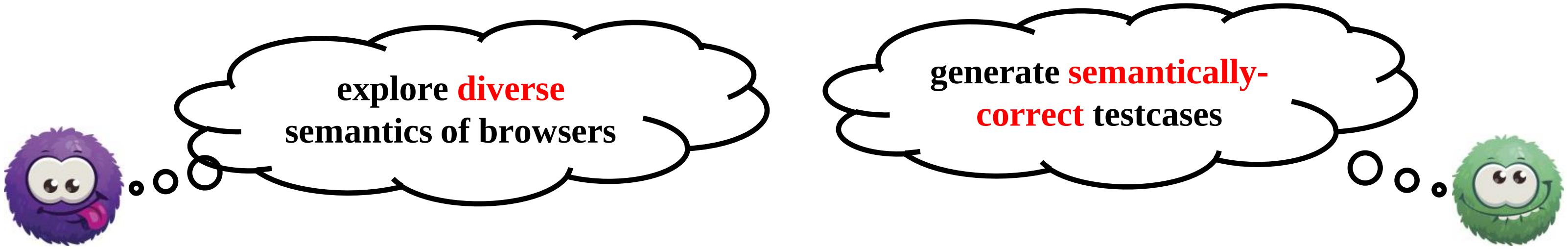
A good browser fuzzer should be able to …



explore **diverse** semantics of browsers

generate **semantically-correct** testcases

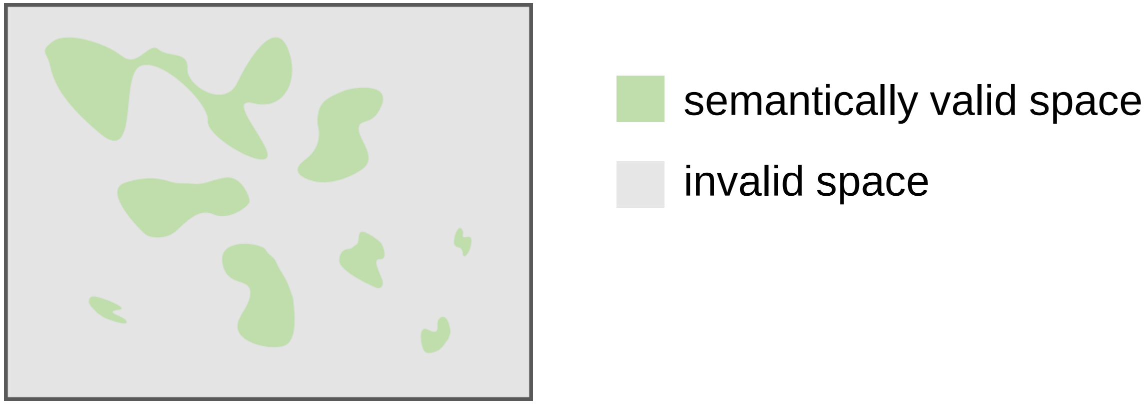
A good browser fuzzer should be able to …



explore **diverse** semantics of browsers

generate **semantically-correct** testcases

Suppose this area presents the input space of a browser ...

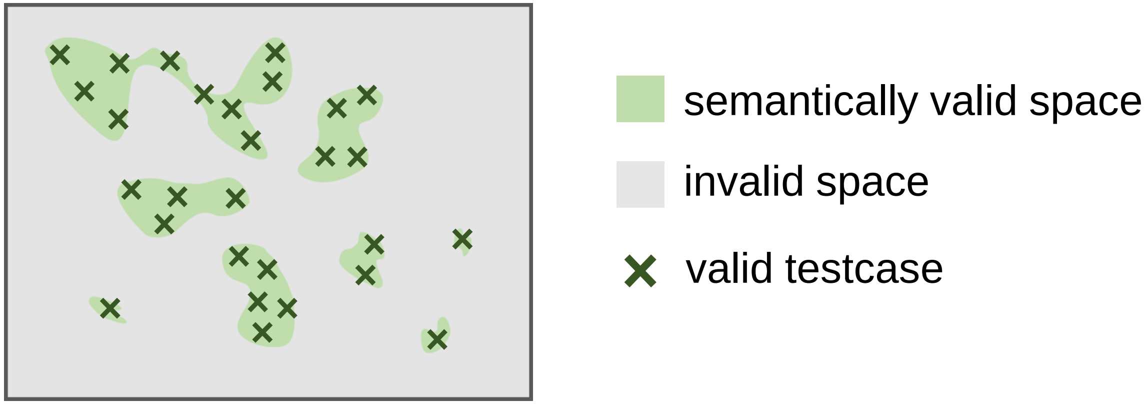


A good browser fuzzer should be able to …

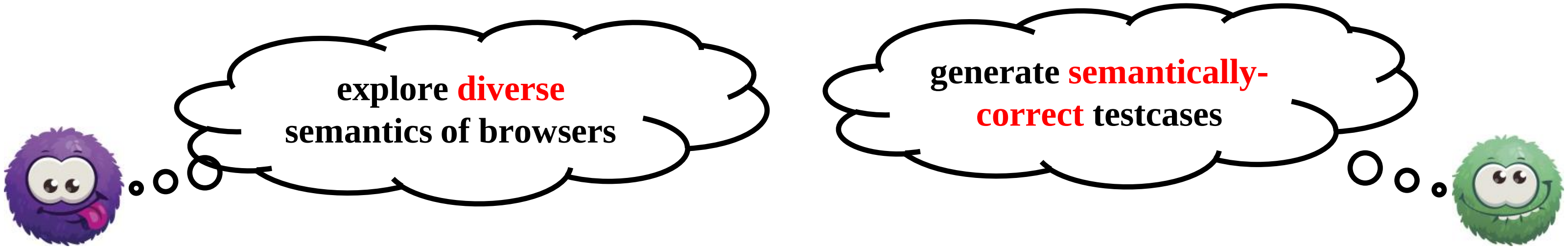
explore diverse semantics of browsers

generate semantically-correct testcases

Ideally, a good fuzzer should generate testcases like …



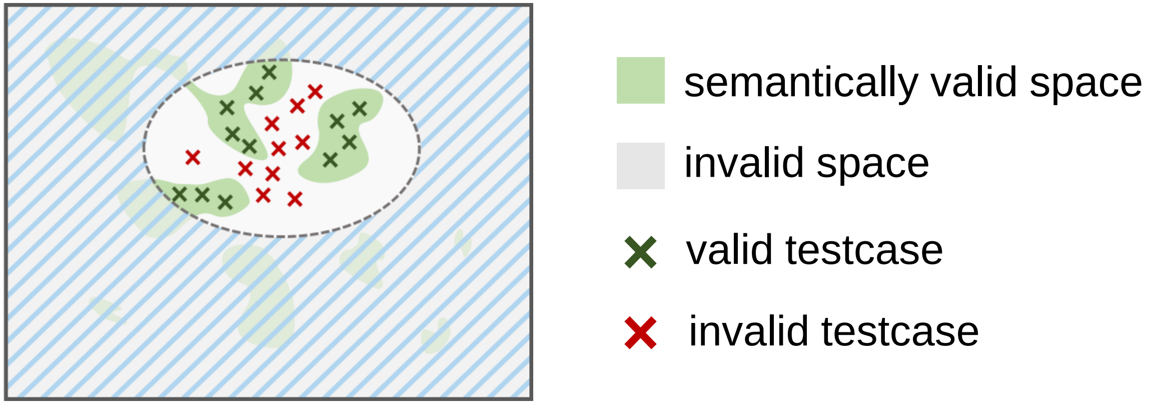
A good browser fuzzer should be able to …



explore **diverse** semantics of browsers

generate **semantically-correct** testcases

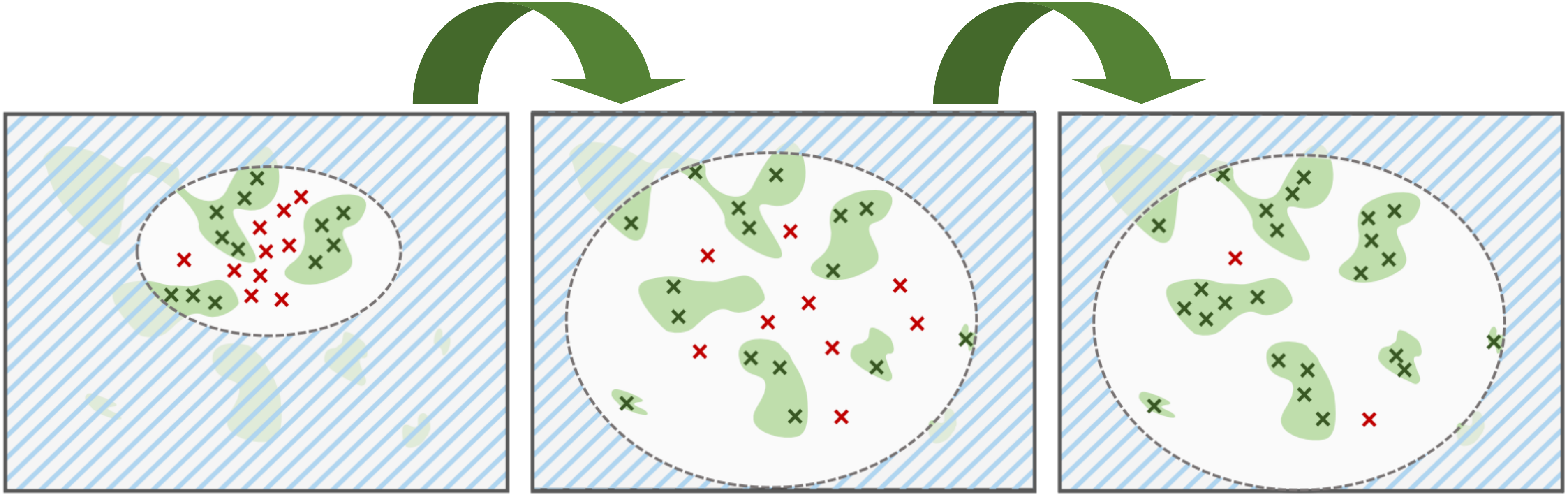
However, existing fuzzers generate testcases like …



Pitfalls: handwritten grammars **limit the semantics exploration** of fuzzers, and still **cannot ensure semantic correctness**

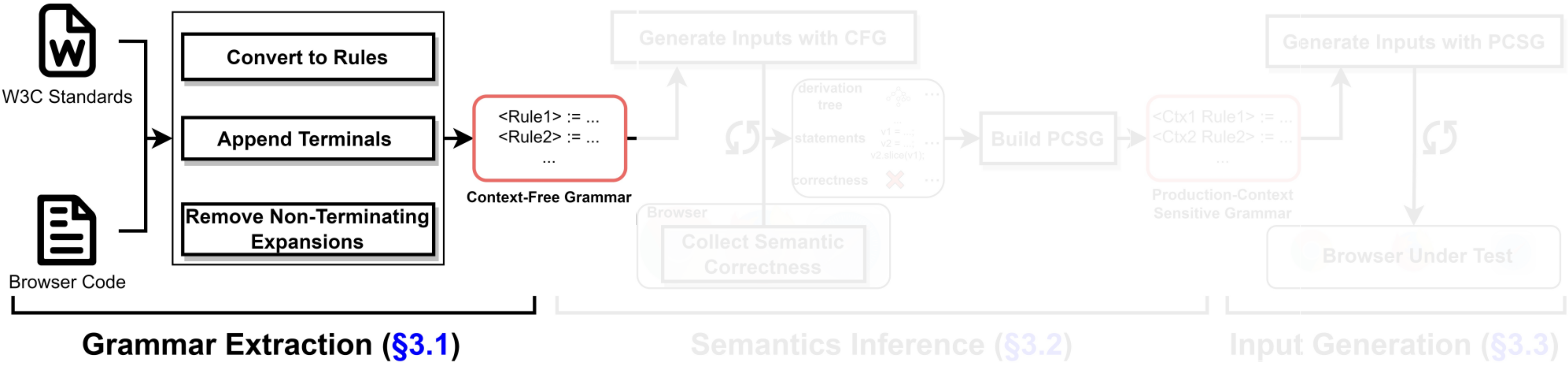
Towards Better Semantics Exploration for Browser Fuzzing

- Goal: automatically generate **quality grammars** to improve browser fuzzing
- workflow:
  - extract a preliminary grammar from ***W3C standards***
  - refine the grammar based on the ***semantic feedback*** of browser executions

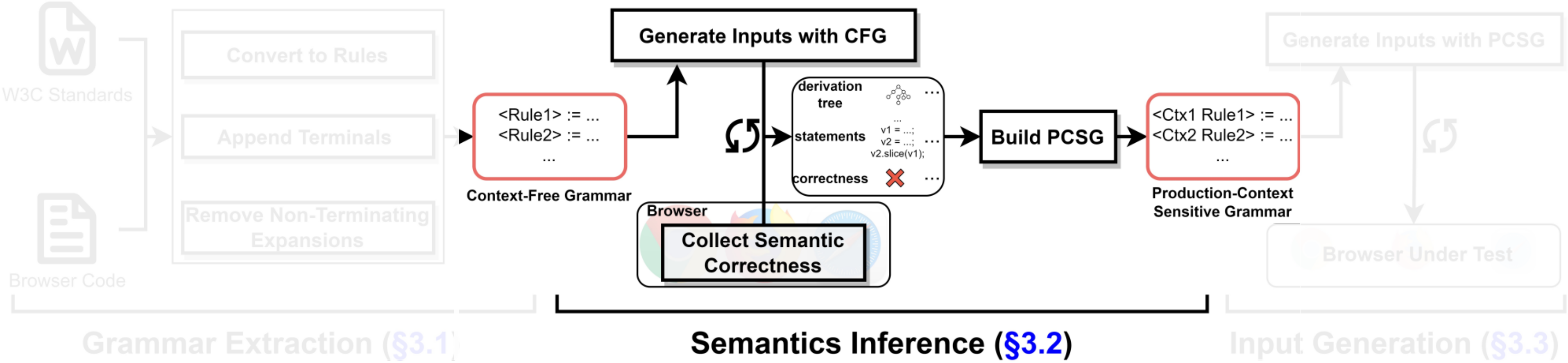


Fuzzing with handwritten grammar Fuzzing with W3C-augmented grammar Fuzzing with refined grammar

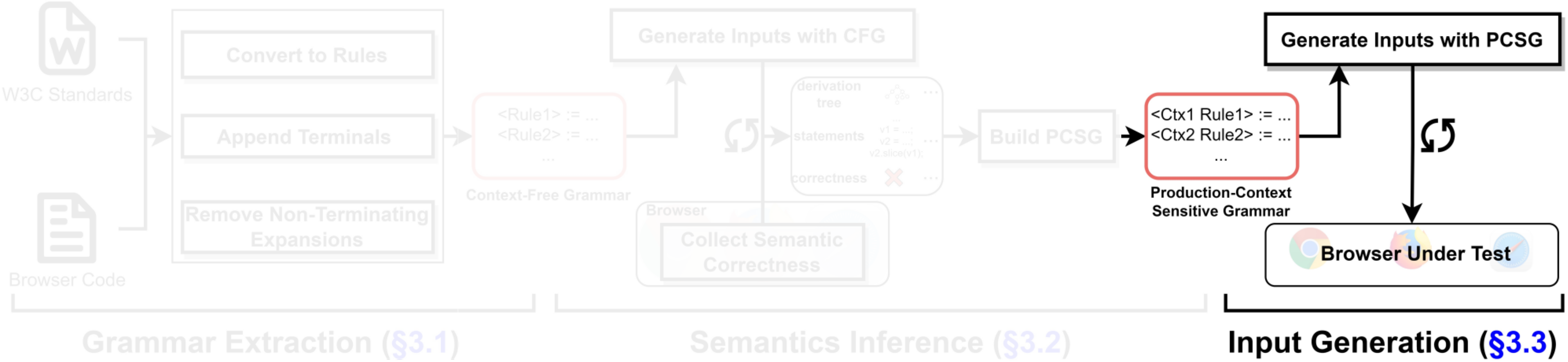
Design Overview



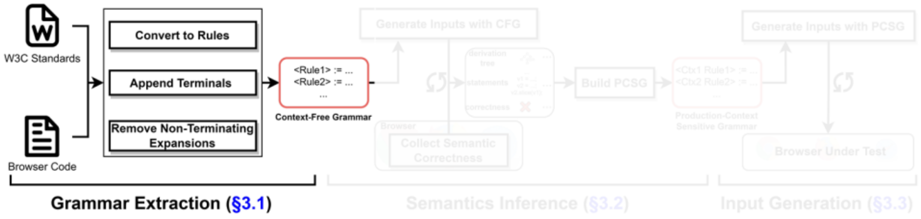
Design Overview



Design Overview



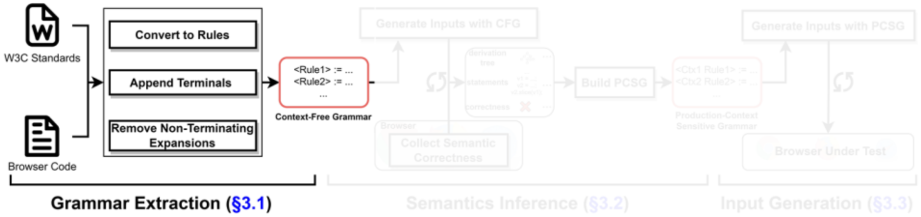
Grammar Extraction



Context-Free Grammar $G = (N, T, P, S)$

N: a set of nonterminals
T: a set of terminals disjoin from N
P: a finite relation in $N \times (N \cup T)^k$, each relation p in the form of $\alpha \rightarrow \beta_1 \beta_2 \cdots \beta_k$
S: a designed start symbol

Grammar Extraction



$$\begin{aligned}\langle cssprop\rangle &\rightarrow \langle fontprop\rangle\\ &\rightarrow \langle alignprop\rangle\\ &\rightarrow \ \ldots\end{aligned}$$

As many semantics as possible

$$\alpha \rightarrow \beta_1\beta_2\ \gamma\ \beta_3 \ldots$$
$$\gamma \rightarrow \beta_4\ \mu\ \beta_5 \ldots \quad ✗$$
$$\mu \ \rightarrow \beta_6\ \alpha \ldots$$

No infinite loops in expansions

$$\begin{aligned}\langle n\_term\rangle &\rightarrow \langle n\_term\rangle \ldots\\ &\rightarrow \cdots\\ &\rightarrow term_1 term_2 \ldots term_n\end{aligned}$$

Expansions to terminal-only expressions

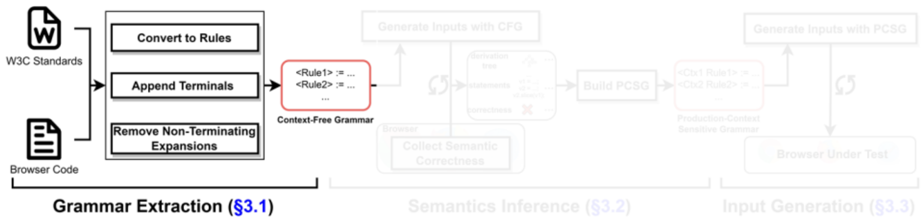
Requirements of the extracted CFG …

□ production rules cover diverse semantics

□ every expansion is not non-terminating

□ every nonterminal can be expanded to a terminal-only expression

Grammar Extraction



Heuristic strategies to convert W3C standards to production rules

Recursively expanding all nonterminals to detect only-loop expansions

Static dataflow analysis on browser source code to find proper terminals

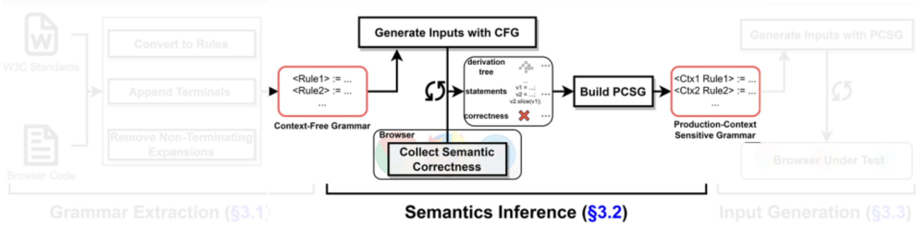
Requirements of the extracted CFG …

☑ production rules cover diverse semantics

☑ every expansion is not non-terminating

☑ every nonterminal can be expanded to a terminal-only expression

Semantics Inference



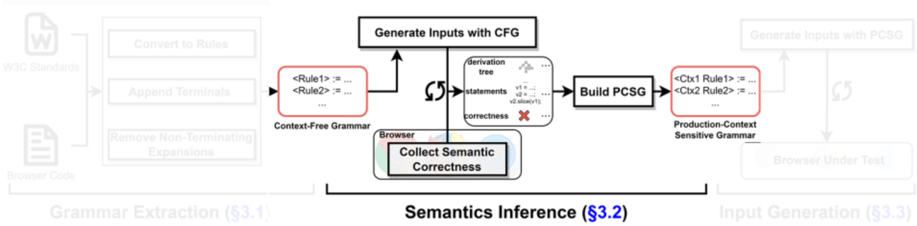
Context-Free Grammar $G = (N, T, P, S)$

Production-Context Sensitive Grammar $G' = (\bar{\mathcal{N}}, \bar{\mathcal{T}}, \bar{\mathcal{P}}, \bar{\mathcal{S}})$

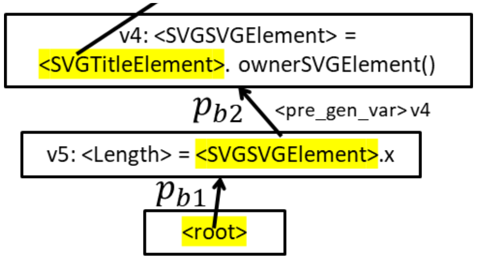
$\bar{\mathcal{N}}$: identical to N
$\bar{\mathcal{T}}$: identical to T
$\bar{\mathcal{P}}$: each relation p in the form of $[\mathbb{C}_p]\alpha \rightarrow \beta_1\beta_2 \cdots \beta_k$
$\bar{\mathcal{S}}$: identical to S

$\mathbb{C}_p$ is a context-checking function for p, $\mathbb{C}_p(ctx) = \begin{cases} \text{true} & \text{likely semantic correct} \quad ✓ \\ \text{false} & \text{unlikely semantic correct} \quad ✗ \end{cases}$

Semantics Inference



Example



v4: <SVGSVGElement> = <SVGTitleElement>. ownerSVGElement()

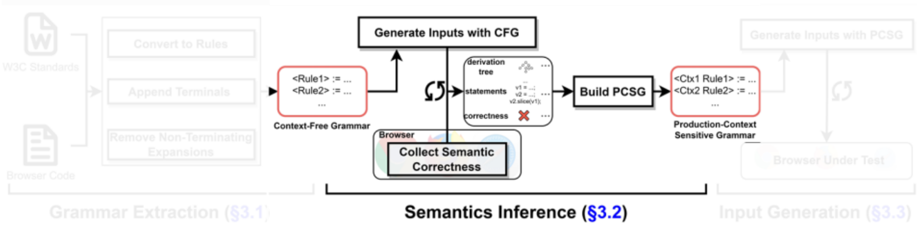
p_{b2} <pre_gen_var> v4

v5: <Length> = <SVGSVGElement>.x

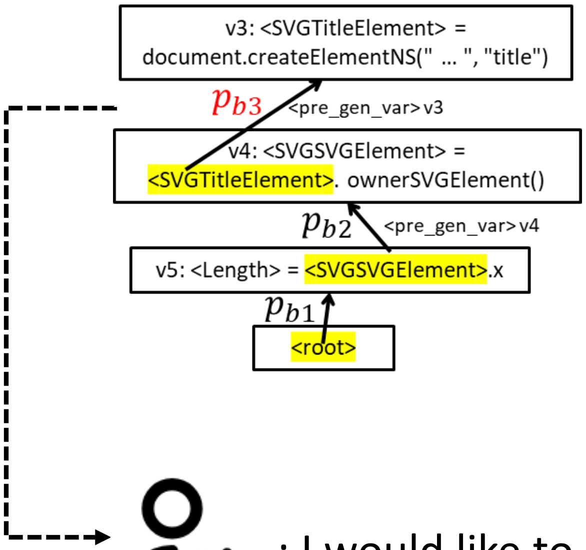
p_{b1}

<root>

Semantics Inference



Example



v3: <SVGTitleElement> = document.createElementNS(" ... ", "title")

v4: <SVGSVGElement> = <SVGTitleElement>. ownerSVGElement()

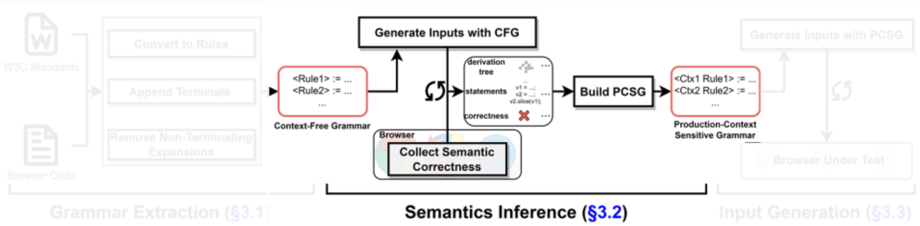
v5: <Length> = <SVGSVGElement>.x

<root>

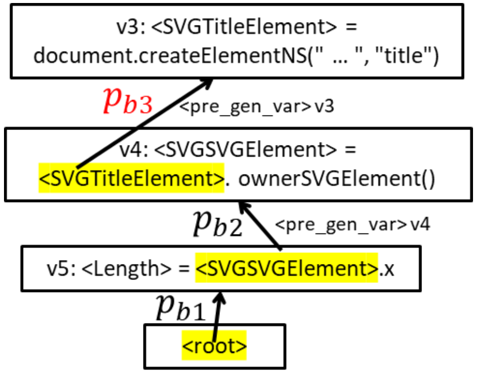
Fuzzer : I would like to know if p_{b3} is a right choice under this context

PCSG : $\mathbb{C}_{p_{b3}}([p_{b1}, p_{b2}])$ returns false, so it is likely to cause a semantic error

Semantics Inference



Example



v3: <SVGTitleElement> = document.createElementNS(" ... ", "title")

p_{b3} <pre_gen_var> v3

v4: <SVGSVGElement> = <SVGTitleElement>. ownerSVGElement()

p_{b2} <pre_gen_var> v4

v5: <Length> = <SVGSVGElement>.x

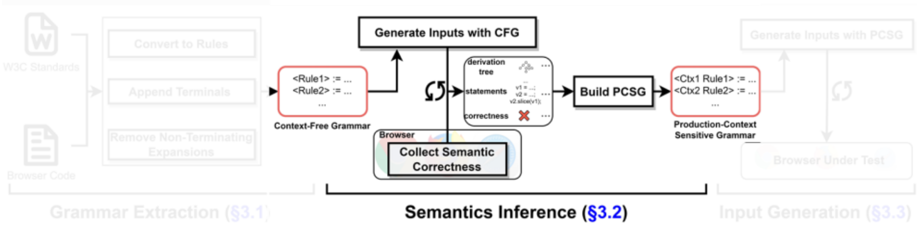
p_{b1}

<root>

```
// v3 is a normal SVGTitleElement element
v3=doc.createElementNS("…", "title");
// v4 is null because v3 is the outermost
v4=v3.ownerSVGElement();
// misuse error because v4 is null
v5=v4.x; 💣
```

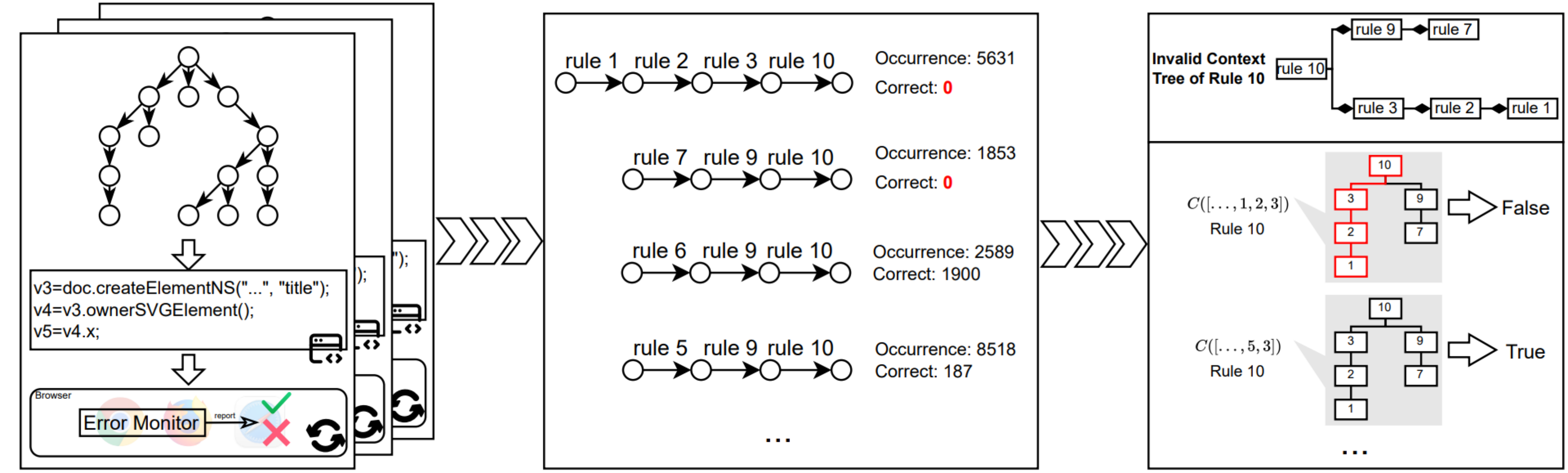
Fact: the generated testcase will triggers a semantic error if the fuzzer selects p_{b3}

Semantics Inference



How can we know $\mathbb{C}_p$ for each production rule p?

- construct a tree-based data structure based on **browsers' feedback**



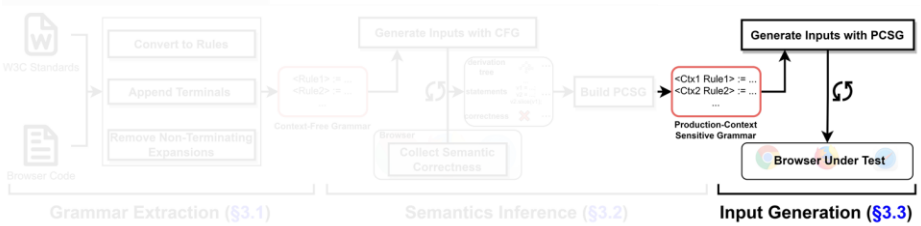
```
v3=doc.createElementNS("...", "title");
v4=v3.ownerSVGElement();
v5=v4.x;
```

Execution Results of generated statements

Occurrence statistics of parent-child rule chains

Construction of context-checking function

Input Generation

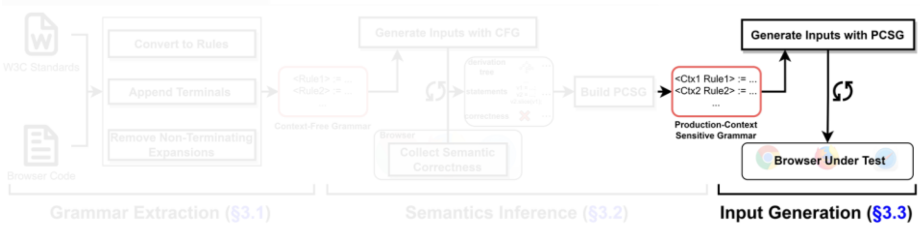


Derivation Tree

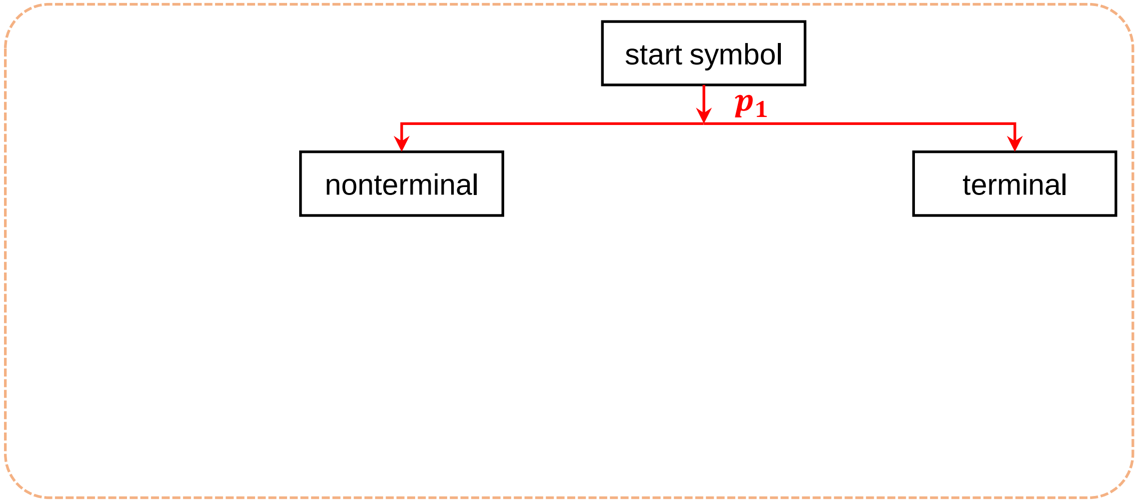
start symbol

CTX = {}

Input Generation



Derivation Tree



start symbol

p_1

nonterminal

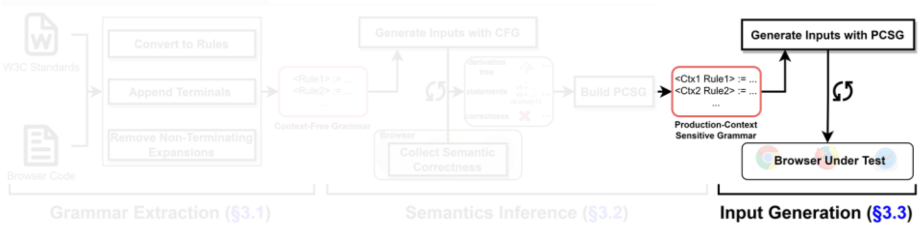
terminal

CTX = {}

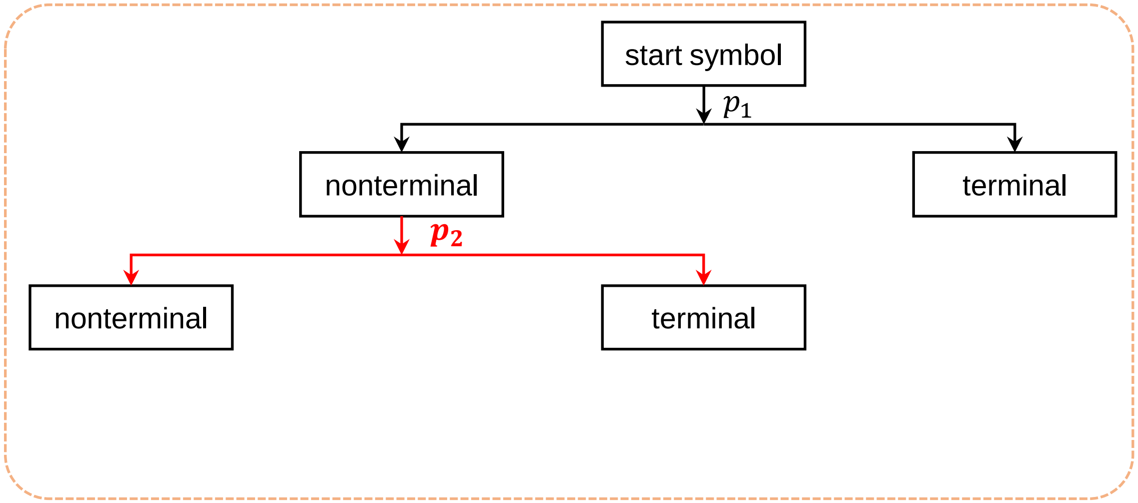
randomly select a rule

$$\mathbb{C}_{p_1}([]) == true?$$

Input Generation



Derivation Tree

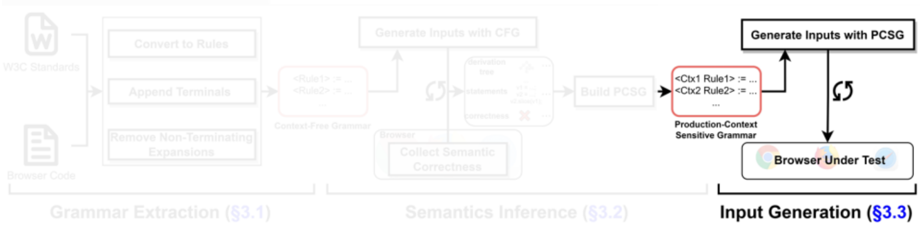


CTX = $\{p_1\}$

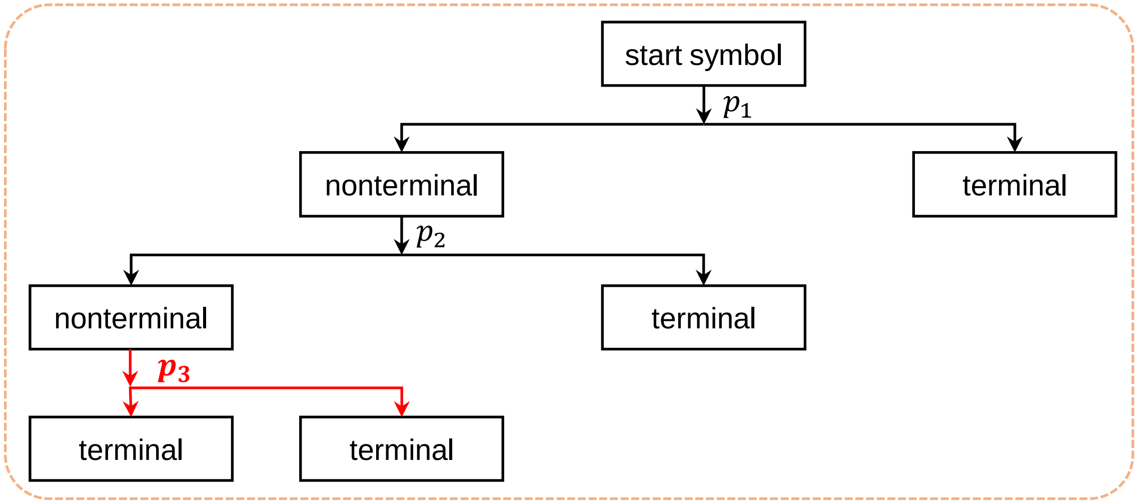
randomly select a rule

$$\mathbb{C}_{p_2}([p_1]) == true?$$

Input Generation



Derivation Tree

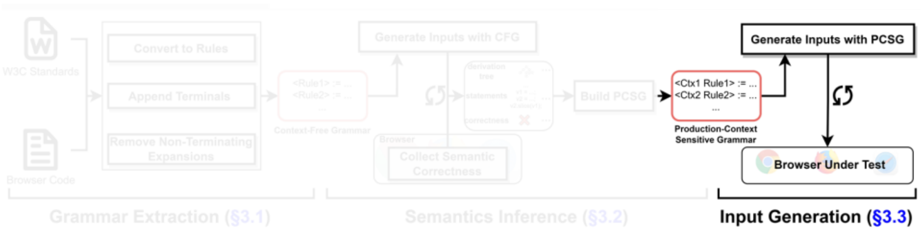


CTX = $\{p_1, p_2\}$

randomly select a rule

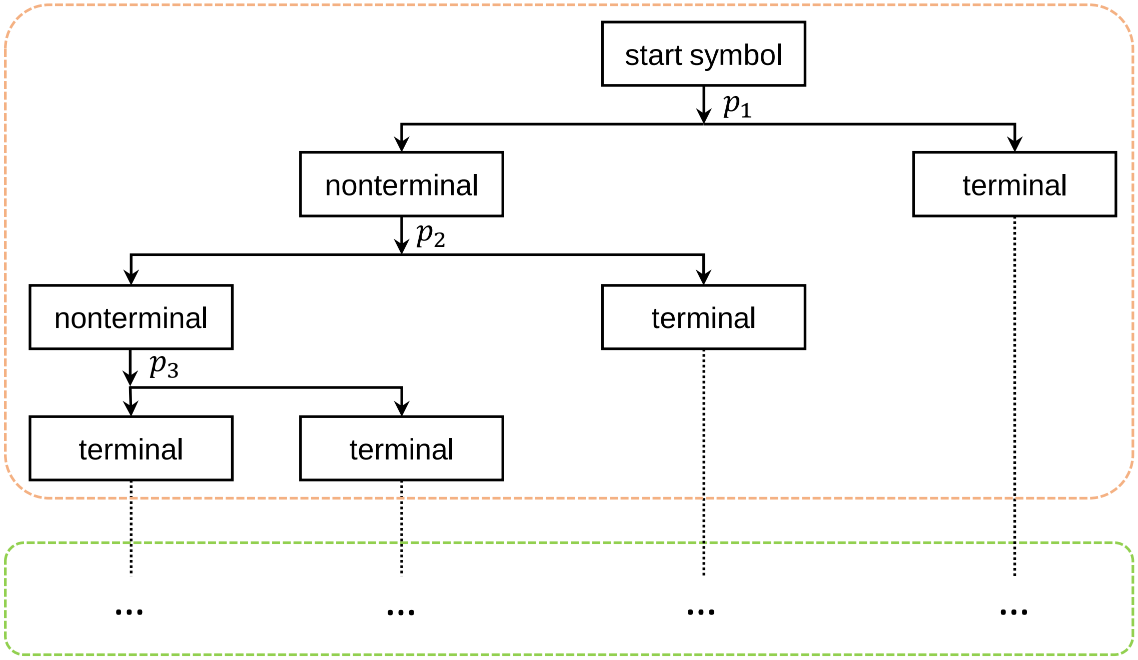
$\mathbb{C}_{p_3}[p_1, p_2] == true?$

Input Generation



Derivation Tree

Testcase

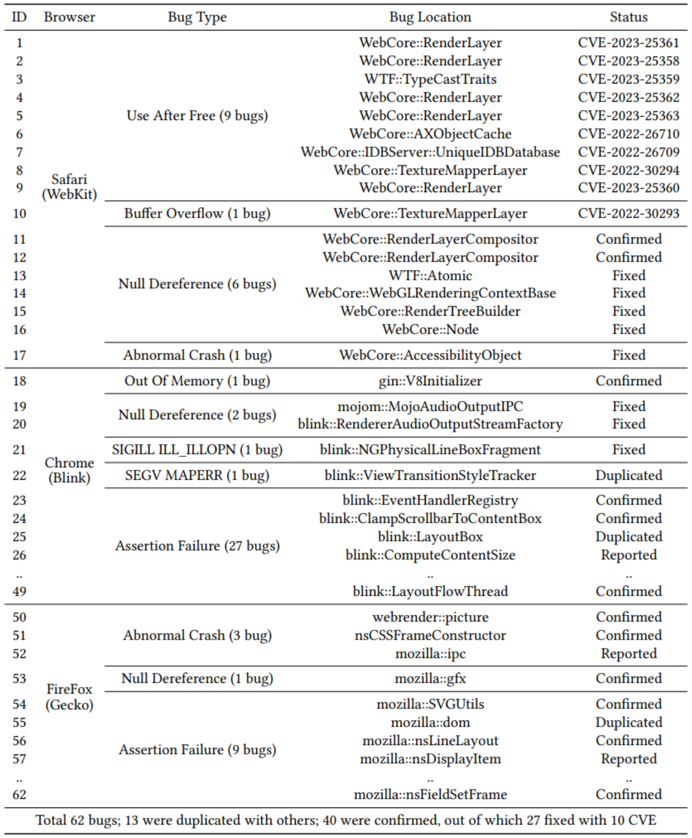


CTX = $\{p_1, p_2, p_3\}$

Evaluation

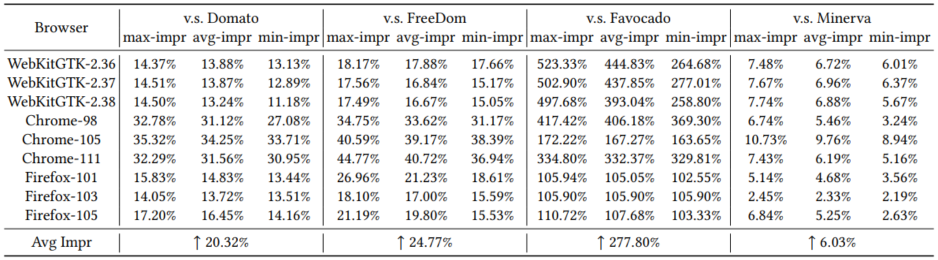
highlight

- found **62** real-world bugs in Safari, Chrome, and Firefox, out of which **40** were confirmed with **10** CVEs
- Compared to existing browser fuzzers
  - 6.03% - 277.80% improvement in branch coverage
  - 3.56% - 160.71% improvement in semantics correctness rate
- Introduced roughly 3.57% overhead during code generation



ID	Browser	Bug Type	Bug Location	Status
1	Safari (WebKit)	Use After Free (9 bugs)	WebCore::RenderLayer	CVE-2023-25361
2			WebCore::RenderLayer	CVE-2023-25358
3			WTF::TypeCastTraits	CVE-2023-25359
4			WebCore::RenderLayer	CVE-2023-25362
5			WebCore::RenderLayer	CVE-2023-25363
6			WebCore::AXObjectCache	CVE-2022-26710
7			WebCore::IDBServer::UniqueIDBDatabase	CVE-2022-26709
8			WebCore::TextureMapperLayer	CVE-2022-30294
9			WebCore::RenderLayer	CVE-2023-25360
10		Buffer Overflow (1 bug)	WebCore::TextureMapperLayer	CVE-2022-30293
11		Null Dereference (6 bugs)	WebCore::RenderLayerCompositor	Confirmed
12			WebCore::RenderLayerCompositor	Confirmed
13			WTF::Atomic	Fixed
14			WebCore::WebGLRenderingContextBase	Fixed
15			WebCore::RenderTreeBuilder	Fixed
16			WebCore::Node	Fixed
17		Abnormal Crash (1 bug)	WebCore::AccessibilityObject	Fixed
18	Chrome (Blink)	Out Of Memory (1 bug)	gin::V8Initializer	Confirmed
19		Null Dereference (2 bugs)	mojom::MojoAudioOutputIPC	Fixed
20			blink::RendererAudioOutputStreamFactory	Fixed
21		SIGILL ILL_ILLOPN (1 bug)	blink::NGPhysicalLineBoxFragment	Fixed
22		SEGV MAPERR (1 bug)	blink::ViewTransitionStyleTracker	Duplicated
23		Assertion Failure (27 bugs)	blink::EventHandlerRegistry	Confirmed
24			blink::ClampScrollbarToContentBox	Confirmed
25			blink::LayoutBox	Duplicated
26			blink::ComputeContentSize	Reported
..			..	..
49			blink::LayoutFlowThread	Confirmed
50	FireFox (Gecko)	Abnormal Crash (3 bug)	webrender::picture	Confirmed
51			nsCSSFrameConstructor	Confirmed
52			mozilla::ipc	Reported
53		Null Dereference (1 bug)	mozilla::gfx	Confirmed
54		Assertion Failure (9 bugs)	mozilla::SVGUtils	Confirmed
55			mozilla::dom	Duplicated
56			mozilla::nsLineLayout	Confirmed
57			mozilla::nsDisplayItem	Reported
..			..	..
62			mozilla::nsFieldSetFrame	Confirmed
Total 62 bugs; 13 were duplicated with others; 40 were confirmed, out of which 27 fixed with 10 CVE				

Table 6. Coverage improvements of SaGe compared to other browser fuzzers in 24 hours over five runs.



Browser	v.s. Domato			v.s. FreeDom			v.s. Favocado			v.s. Minerva		
	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr
WebKitGTK-2.36	14.37%	13.88%	13.13%	18.17%	17.88%	17.66%	523.33%	444.83%	264.68%	7.48%	6.72%	6.01%
WebKitGTK-2.37	14.51%	13.87%	12.89%	17.56%	16.84%	15.17%	502.90%	437.85%	277.01%	7.67%	6.96%	6.37%
WebKitGTK-2.38	14.50%	13.24%	11.18%	17.49%	16.67%	15.05%	497.68%	393.04%	258.80%	7.74%	6.88%	5.67%
Chrome-98	32.78%	31.12%	27.08%	34.75%	33.62%	31.17%	417.42%	406.18%	369.30%	6.74%	5.46%	3.24%
Chrome-105	35.32%	34.25%	33.71%	40.59%	39.17%	38.39%	172.22%	167.27%	163.65%	10.73%	9.76%	8.94%
Chrome-111	32.29%	31.56%	30.95%	44.77%	40.72%	36.94%	334.80%	332.37%	329.81%	7.43%	6.19%	5.16%
Firefox-101	15.83%	14.83%	13.44%	26.96%	21.23%	18.61%	105.94%	105.05%	102.55%	5.14%	4.68%	3.56%
Firefox-103	14.05%	13.72%	13.51%	18.10%	17.00%	15.59%	105.90%	105.90%	105.90%	2.45%	2.33%	2.19%
Firefox-105	17.20%	16.45%	14.16%	21.19%	19.80%	15.53%	110.72%	107.68%	103.33%	6.84%	5.25%	2.63%
Avg Impr	↑ 20.32%			↑ 24.77%			↑ 277.80%			↑ 6.03%		

Summary

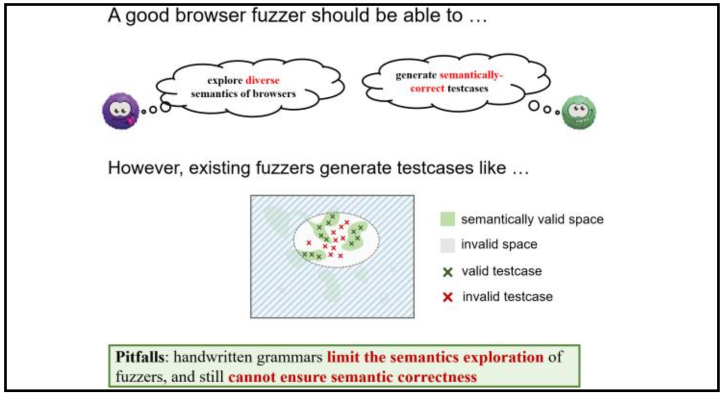
Artifacts: https://zenodo.org/records/8328742

Artifacts Available acm V1.1

Artifacts Evaluated acm Reusable V1.1

Prototype: https://github.com/ChijinZ/SaGe-Browser-Fuzzer

Goal: better semantics exploration



A good browser fuzzer should be able to …

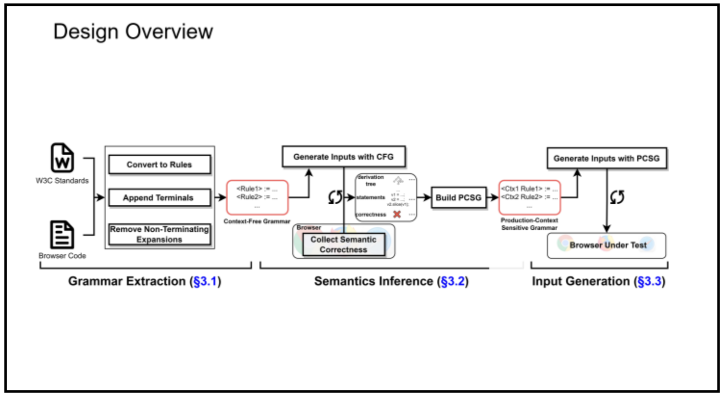
explore diverse semantics of browsers
generate semantically-correct testcases

However, existing fuzzers generate testcases like …

🟩 semantically valid space
▨ invalid space
✓ valid testcase
✗ invalid testcase

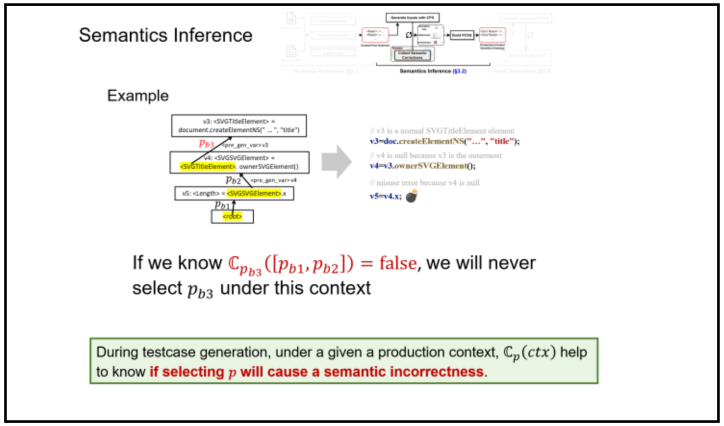
Pitfalls: handwritten grammars **limit the semantics exploration** of fuzzers, and still **cannot ensure semantic correctness**

Method: learn grammars from specs and source code



Design Overview

Insight: infer semantics for production rules



Semantics Inference

Semantics Inference (§3.2)
v3: <SVGTitleElement> = document.createElementNS(" … ", "title")
p_{b3}
<pre_gen_var>v3
v4: <SVGSVGElement> = <SVGTitleElement>. ownerSVGElement()
p_{b2}
<pre_gen_var>v4
v5: <Length> = <SVGSVGElement>.x
p_{b1}
<root>

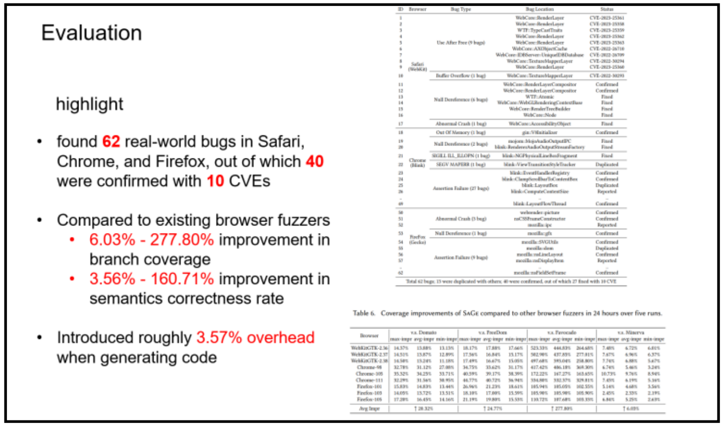
Example

```
// v3 is a normal SVGTitleElement element
v3=doc.createElementNS("…", "title");
// v4 is null because v3 is the outermost
v4=v3.ownerSVGElement();
// misuse error because v4 is null
v5=v4.x; 💣
```

If we know $\mathbb{C}_{p_{b3}}([p_{b1}, p_{b2}]) = \text{false}$, we will never select p_{b3} under this context

During testcase generation, under a given a production context, $\mathbb{C}_p(ctx)$ help to know **if selecting p will cause a semantic incorrectness**.

Evaluation: perform well and can find real-world bugs



Evaluation

highlight

- found **62** real-world bugs in Safari, Chrome, and Firefox, out of which **40** were confirmed with **10** CVEs
- Compared to existing browser fuzzers
  - 6.03% - 277.80% improvement in branch coverage
  - 3.56% - 160.71% improvement in semantics correctness rate
- Introduced roughly 3.57% overhead when generating code

ID	Browser	Bug Type	Bug Location	Status
1	Safari (WebKit)	Use After Free (9 bugs)	WebCore::RenderLayer	CVE-2023-25361
2			WebCore::RenderLayer	CVE-2023-25358
3			WTF::TypeCastTraits	CVE-2023-25359
4			WebCore::RenderLayer	CVE-2023-25362
5			WebCore::RenderLayer	CVE-2023-25363
6			WebCore::AXObjectCache	CVE-2022-26710
7			WebCore::IDBServer::UniqueIDBDatabase	CVE-2022-26709
8			WebCore::TextureMapperLayer	CVE-2022-30294
9			WebCore::RenderLayer	CVE-2023-25360
10		Buffer Overflow (1 bug)	WebCore::TextureMapperLayer	CVE-2022-30293
11		Null Dereference (6 bugs)	WebCore::RenderLayerCompositor	Confirmed
12			WebCore::RenderLayerCompositor	Confirmed
13			WTF::Atomic	Fixed
14			WebCore::WebGLRenderingContextBase	Fixed
15			WebCore::RenderTreeBuilder	Fixed
16			WebCore::Node	Fixed
17		Abnormal Crash (1 bug)	WebCore::AccessibilityObject	Fixed
18	Chrome (Blink)	Out Of Memory (1 bug)	gin::V8Initializer	Confirmed
19		Null Dereference (2 bugs)	mojom::MojoAudioOutputIPC	Fixed
20			blink::RendererAudioOutputStreamFactory	Fixed
21		SIGILL ILL_ILLOPN (1 bug)	blink::NGPhysicalLineBoxFragment	Fixed
22		SEGV MAPERR (1 bug)	blink::ViewTransitionStyleTracker	Duplicated
23		Assertion Failure (27 bugs)	blink::EventHandlerRegistry	Confirmed
24			blink::ClampScrollBarToContentBox	Confirmed
25			blink::LayoutBox	Duplicated
26			blink::ComputeContentSize	Reported
...			...	...
49			blink::LayoutFlowThread	Confirmed
50	FireFox (Gecko)	Abnormal Crash (3 bug)	webrender::picture	Confirmed
51			nsCSSFrameConstructor	Confirmed
52			mozilla::ipc	Reported
53		Null Dereference (1 bug)	mozilla::gfx	Confirmed
54		Assertion Failure (9 bugs)	mozilla::SVGUtils	Confirmed
55			mozilla::dom	Duplicated
56			mozilla::nsLineLayout	Confirmed
57			mozilla::nsDisplayItem	Reported
...			...	...
62			mozilla::nsFieldSetFrame	Confirmed
Total 62 bugs; 15 were duplicated with others; 40 were confirmed, out of which 27 fixed with 10 CVE				

Table 6. Coverage improvements of SAGE compared to other browser fuzzers in 24 hours over five runs.

Browser	v.s. Domato			v.s. FreeDom			v.s. Favocado			v.s. Minerva		
	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr	max-impr	avg-impr	min-impr
WebKitGTK-2.36	14.37%	13.88%	13.13%	18.17%	17.88%	17.66%	523.33%	444.83%	264.68%	7.48%	6.72%	6.01%
WebKitGTK-2.37	14.51%	13.87%	12.89%	17.56%	16.84%	15.17%	502.90%	437.85%	277.01%	7.67%	6.96%	6.37%
WebKitGTK-2.38	14.58%	13.24%	11.18%	17.49%	16.67%	15.05%	497.68%	393.04%	258.80%	7.74%	6.88%	5.67%
Chrome-98	32.78%	31.12%	27.08%	34.75%	33.62%	31.17%	417.42%	406.18%	369.30%	6.74%	5.46%	3.24%
Chrome-105	35.32%	34.25%	33.71%	40.59%	39.17%	38.39%	172.22%	167.27%	163.65%	10.73%	9.76%	8.94%
Chrome-111	32.29%	31.56%	30.95%	44.77%	40.72%	36.94%	334.80%	332.37%	329.81%	7.45%	6.19%	5.16%
Firefox-101	15.83%	14.83%	13.44%	26.96%	21.23%	18.61%	105.94%	105.05%	102.55%	5.14%	4.68%	3.56%
Firefox-103	14.05%	13.72%	13.51%	18.10%	17.00%	15.59%	105.90%	105.90%	105.90%	2.45%	2.33%	2.19%
Firefox-105	17.20%	16.45%	14.16%	21.19%	19.80%	15.53%	110.72%	107.68%	103.33%	6.84%	5.25%	2.63%
Avg Impr	↑ 28.32%			↑ 24.77%			↑ 277.80%			↑ 6.03%		

This research is sponsored in part by Deng Feng Fund