DOM:
Specification & Client Reasoning

Azalea Raad José Fragoso Santos Philippa Gardner
Imperial College London

APLAS'16
23 November 2016

Document Object Model (DOM)

- Cross-platform, language-independent, XML update library
- Standardised by W3C (and later WHATWG)
  - Written in English (informal, ambiguous)
  - Described in an OO fashion, in an **axiomatic** style
  - Followed by browser vendors

Which DOM?

- DOM Level 1 (Core)
  - Complete model of an XML document
  - Operations for manipulating the document via *interfaces*

Which DOM?

- DOM Level 1 (Core)
  - Complete model of an XML document
  - Operations for manipulating the document via *interfaces*
- DOM Levels 2-4
  - DOM Core (minimally changed from Level 1)
  - Additional features:
    - Event model, XML namespaces, … (Level 2)
    - Keyboard event handling, serialisation, … (Level 3)
    - HTMLCollections, Elements, … (Level 4)

Which DOM?

- DOM Level 1 (Core)
  - Complete model of an XML document
  - Operations for manipulating the document via *interfaces*:
    - The *Node* interface
    - 12 specialised node interfaces
    - 2 interfaces for node collections
    - DOM Exceptions

DOM Specification Wish List

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)
- **local** (minimal operation footprint)

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)
- **local** (minimal operation footprint)
- **compositional** (client program spec from DOM spec)

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)
- **local** (minimal operation footprint)
- **compositional** (client program spec from DOM spec)
- Easily **integrated** with existing program logics

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)
- **local** (minimal operation footprint)
- **compositional** (client program spec from DOM spec)
- Easily **integrated** with existing program logics
  - “Not another program logic!” -- almost everyone in the community

DOM Specification Wish List

- **faithful**; **axiomatic** (not operational); **abstract** (implementation independent)
- **local** (minimal operation footprint)
- **compositional** (client program spec from DOM spec)
- Easily **integrated** with existing program logics
  - “Not another program logic!” -- almost everyone in the community
  - Reason about client programs in different languages (C, JS, Java, …)

Existing Specification (Smith et al. 2008)

- Formally Specified a DOM Core *fragment* in context logic (CL)
- Justified CL over first order logic (scalability)
- Later extended to **full** DOM Core (2011)

Existing Specification (Smith et al. 2008)

- Formally Specified a DOM Core *fragment* in context logic (CL)
  - The *Node* interface
  - 4 (of 12) specialised node interfaces
  - 1 (of 2) Interface for node collections
  - DOM Exceptions modelled as *faults*
- Justified CL over first order logic (scalability)
- Later extended to **full** DOM Core (2011)

Existing Specification (Smith et al. 2008)

- Formally Specified a DOM Core *fragment* in context logic (CL)
  - ✗ (almost) **faithful**
    - restrictive w.r.t. the spec of node collections
  - ✔ **axiomatic**
  - ✔ **abstract**
  - ✗ **local** (acknowledged by the authors)
    - substantial overapproximation of e.g. `appendChild` footprint
  - ✗ **compositional** (not known by authors - claimed otherwise)
    - a simple (4 loc) client program requires **6** specifications!
  - ✗ Easily **integrated** with existing program logics
    - CL model not compatible with separation logic (SL)
    - cannot be integrated into SL-based program logics
- Justified CL over first order logic (scalability)
- Later extended to **full** DOM Core (2011)

Contributions

- Formally Specified the same DOM Core fragment
  - ✔ **faithful**
    - accurately specify the behaviour of node collections
  - ✔ **axiomatic**
  - ✔ **abstract**
  - ✔ **local**
    - minimally capture the footprint of e.g. `appendChild`
  - ✔ **compositional**
    - same simple client program requires only **1** specification
  - ✔ Easily **integrated** with (SL-based) program logics
    - compatible model with SL

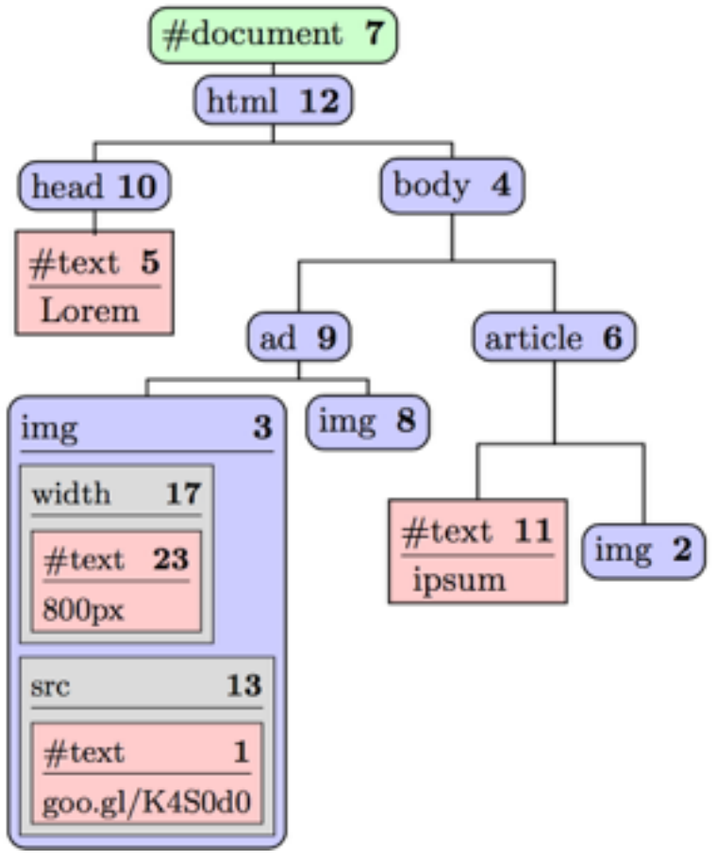
Contributions

- Formally Specified the same DOM Core fragment
  - ✔ **faithful**
    - accurately specify the behaviour of node collections
  - ✔ **axiomatic**
  - ✔ **abstract**
  - ✔ **local**
    - minimally capture the footprint of e.g. `appendChild`
  - ✔ **compositional**
    - same simple client program requires only **1** specification
  - ✔ Easily **integrated** with (SL-based) program logics
    - compatible model with SL
- General methodology for extending SL-based logics with DOM spec
  - Integrated DOM spec with JSLogic (JavaScript Program Logic - POPL'12)
  - Verified multiple ad blocker scripts in JavaScript

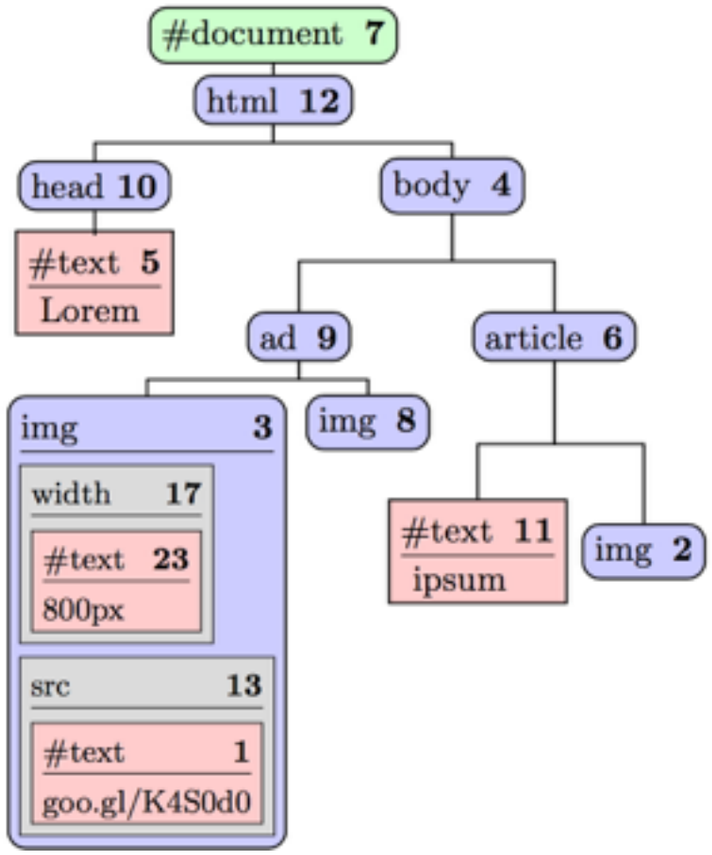
Contributions

- Formally Specified the same DOM Core fragment
  - ✔ **faithful**
    - accurately specify the behaviour of node collections
  - ✔ **axiomatic**
  - ✔ **abstract**
  - ✔ **local**
    - minimally capture the footprint of e.g. `appendChild`
  - ✔ **compositional**
    - same simple client program requires only **1** specification
  - ✔ Easily **integrated** with (SL-based) program logics
    - compatible model with SL
- General methodology for extending SL-based logics with DOM spec
  - Integrated DOM spec with JSLogic (JavaScript Program Logic - POPL'12)
  - Verified multiple ad blocker scripts in JavaScript
- Justified DOM spec w.r.t. an implementation
  - Upcoming thesis (Don't wait for the movie, order your copy now!)

DOM Tree Structure

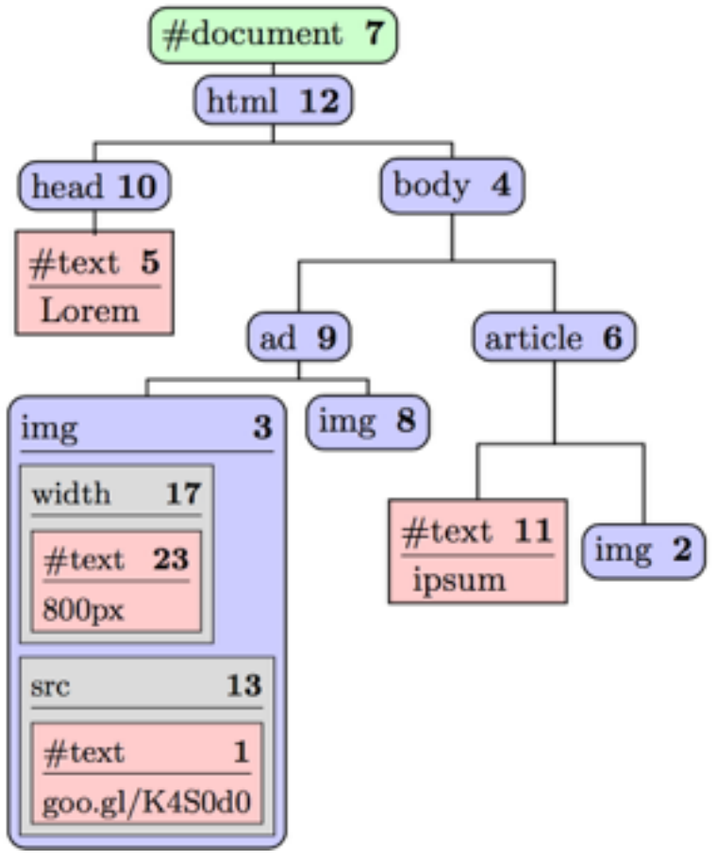


DOM Tree Structure



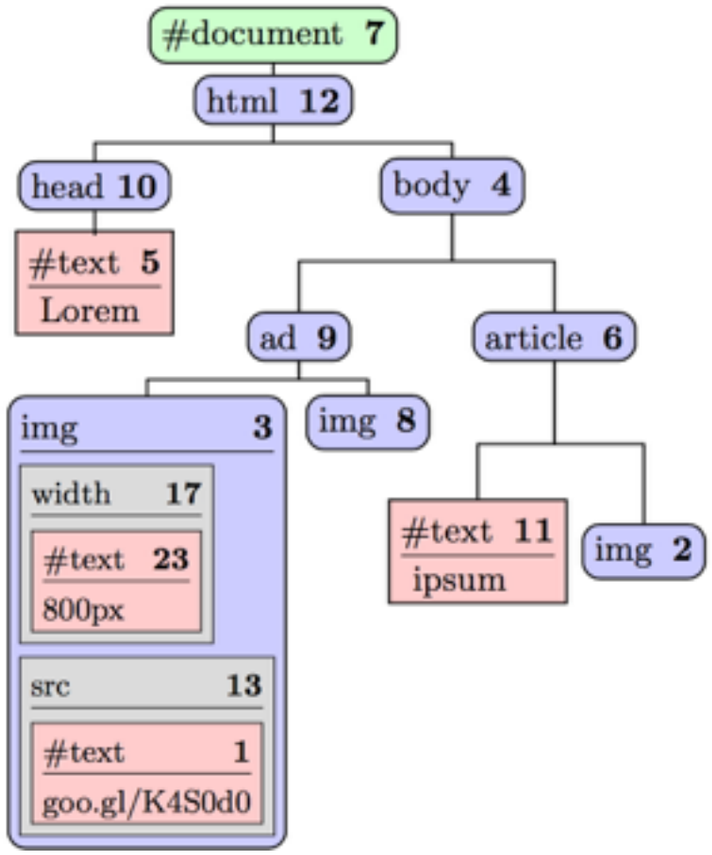
- Each node uniquely identified (an integer identifier)

DOM Tree Structure



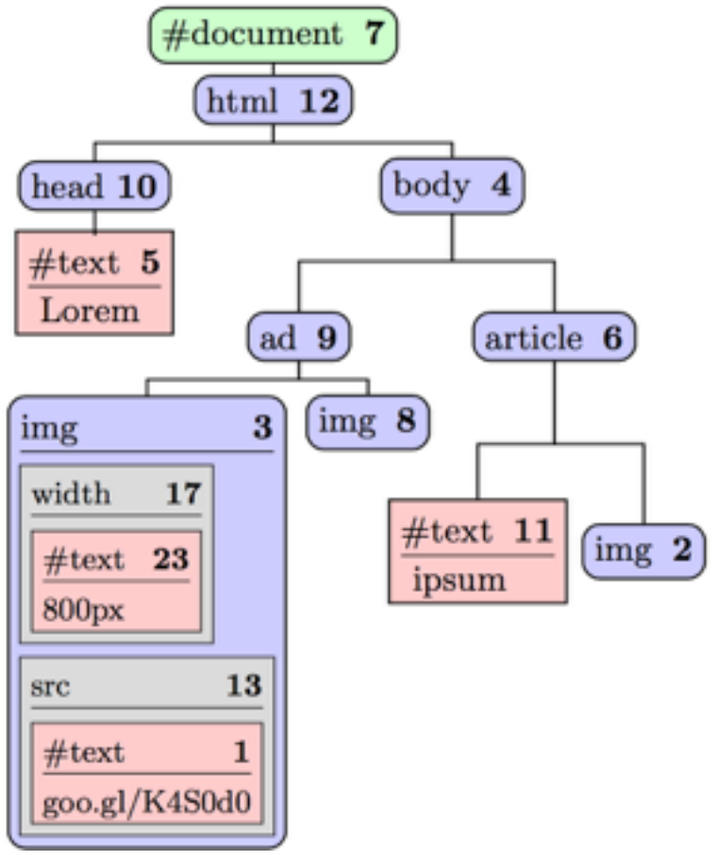
- Each node uniquely identified (an integer identifier)
  - The *Document*
    children: at most one *Element*

DOM Tree Structure



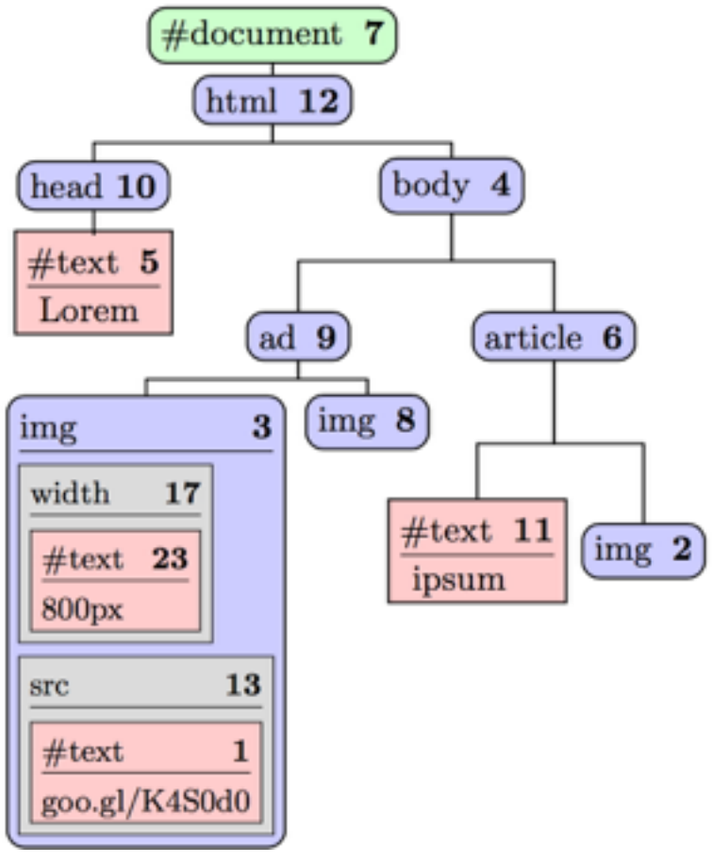
- Each node uniquely identified (an integer identifier)
  - The *Document*
    children: at most one *Element*
  - *Elements*
    children: *Text* and *Element* nodes

DOM Tree Structure



- Each node uniquely identified (an integer identifier)
  - The *Document*
    children: at most one *Element*
  - *Elements*
    children: *Text* and *Element* nodes
  - *Texts*
    children: none, value: arbitrary string
  - *Attributes*
    children: *Text* nodes, value: concatenated values of children

n.getAttribute(s)

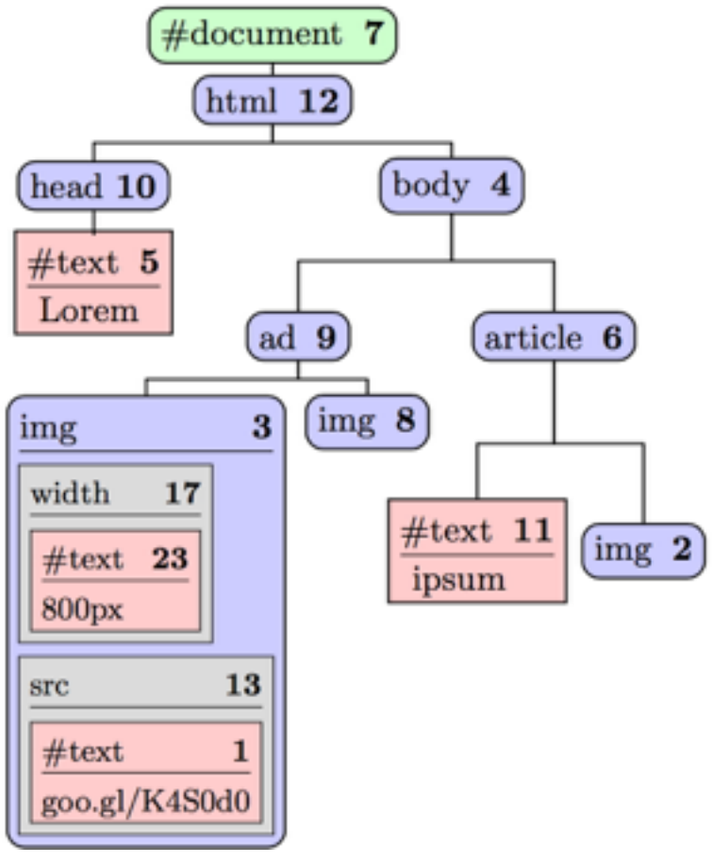


`n.getAttribute(s)`:

When **`n`** identifies an element node, the value of the attribute named **`s`** is returned, if it exists; otherwise “ ” is returned.

e.g. when **`n`**=3 and **`s`**=“src” —> the result is “goo.gl/K4S0d0”

n.getAttribute(s)



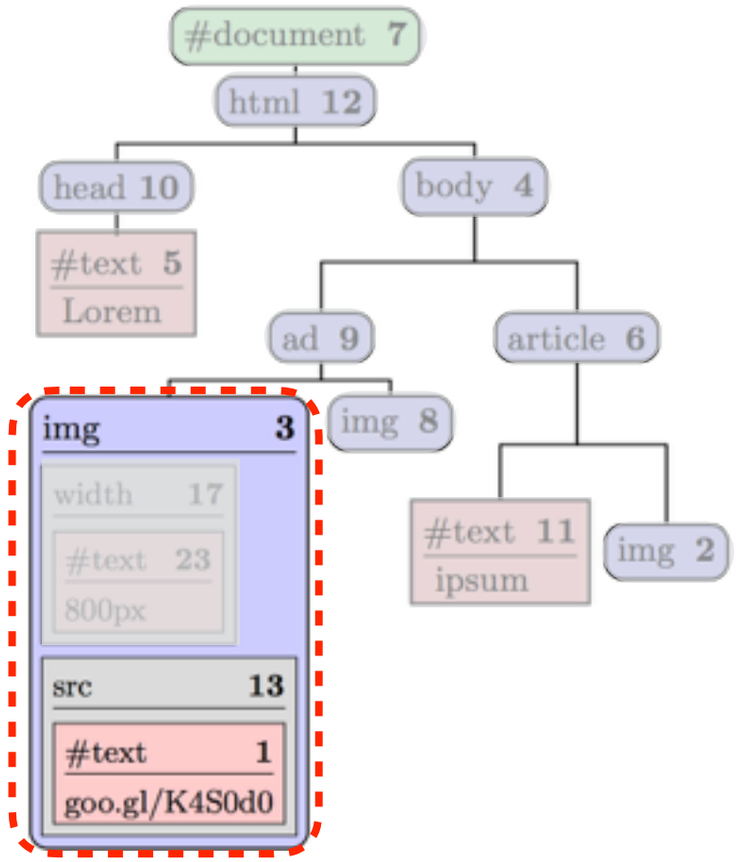
`n.getAttribute(s)`:

When **`n`** identifies an element node, the value of the attribute named **`s`** is returned, if it exists; otherwise " " is returned.

e.g. when **`n`**=3 and **`s`**="src" —> the result is "goo.gl/K4S0d0"

footprint: element node **`n`** and its attribute named **`s`**

n.getAttribute(s)



`n.getAttribute(s)`:

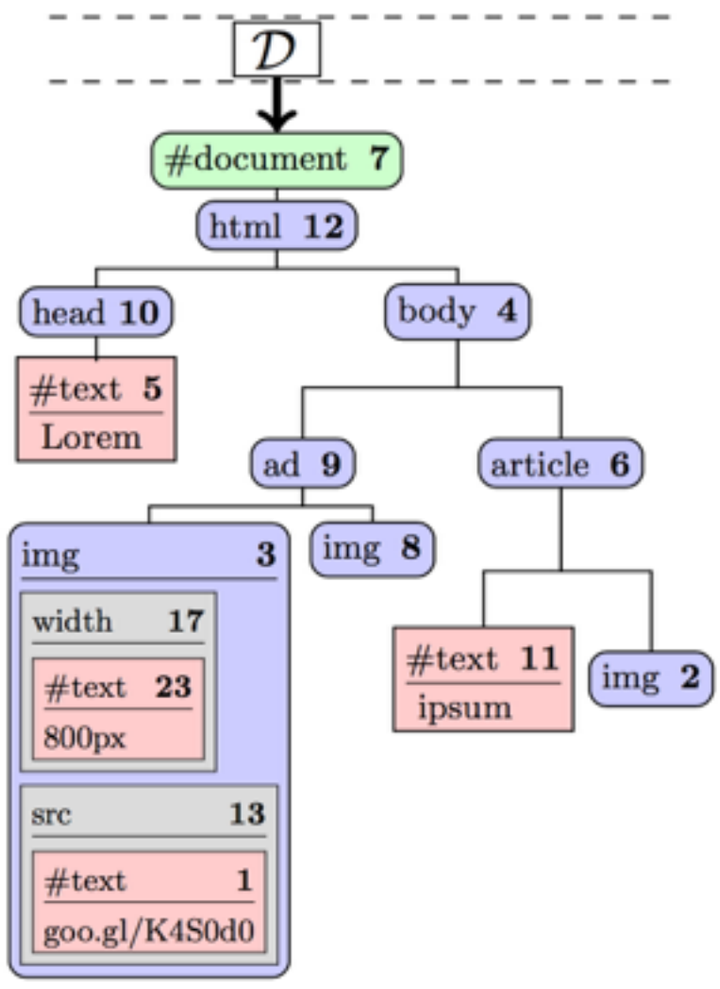
When **`n`** identifies an element node, the value of the attribute named **`s`** is returned, if it exists; otherwise " " is returned.

e.g. when **`n`**=3 and **`s`**="src" —> the result is "goo.gl/K4S0d0"

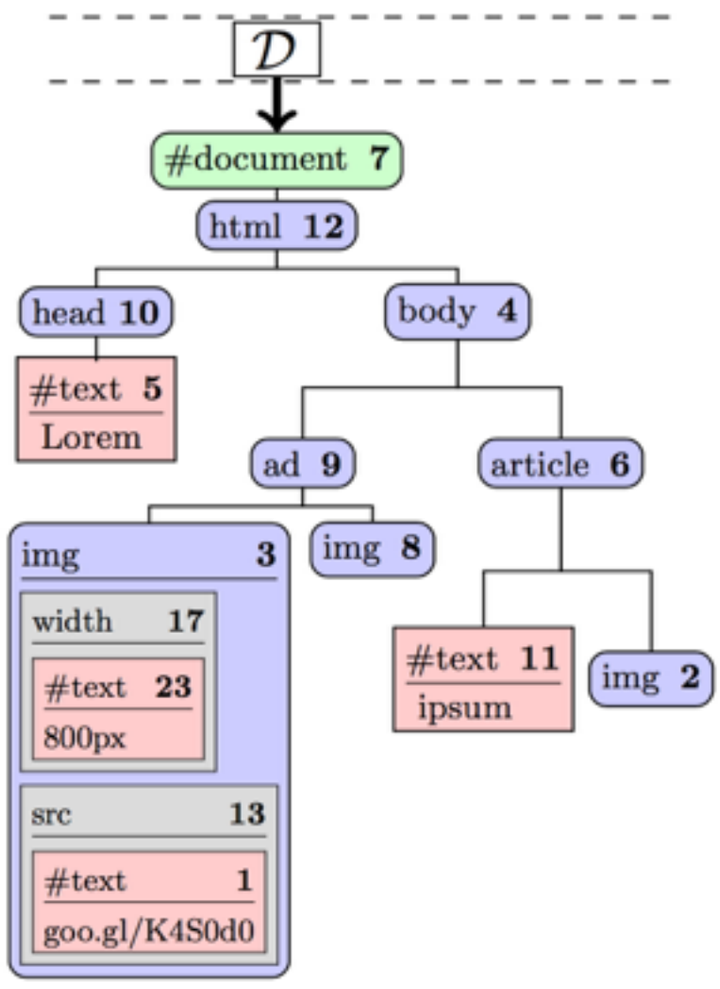
footprint: element node **`n`** and its attribute named **`s`**

e.g. when **`n`**=3 and **`s`**="src" —> footprint in dashed box

Structural Separation Logic (SSL)



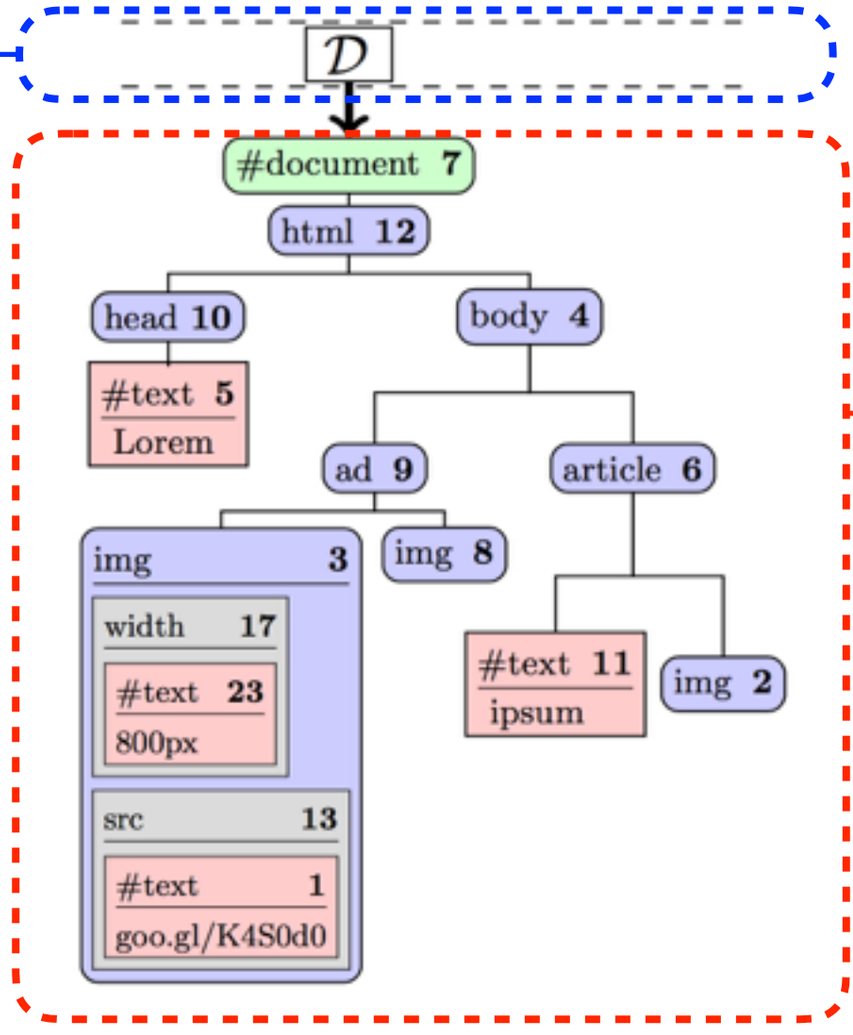
Structural Separation Logic (SSL)



- Program states modelled as ***abstract heaps***

Structural Separation Logic (SSL)

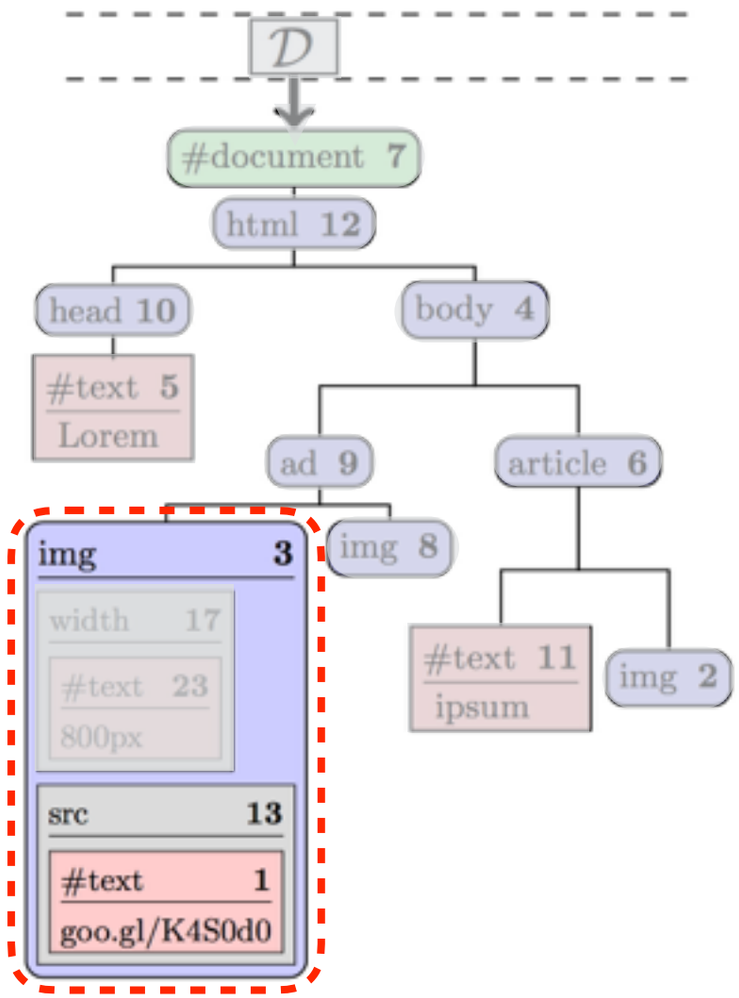
Heap domain: addresses
(DOM root address $\mathcal{D}$)



Heap range: abstract data
(abstract DOM tree)

- Program states modelled as ***abstract heaps***
- Abstract heaps map addresses to abstract data

Structural Separation Logic (SSL)

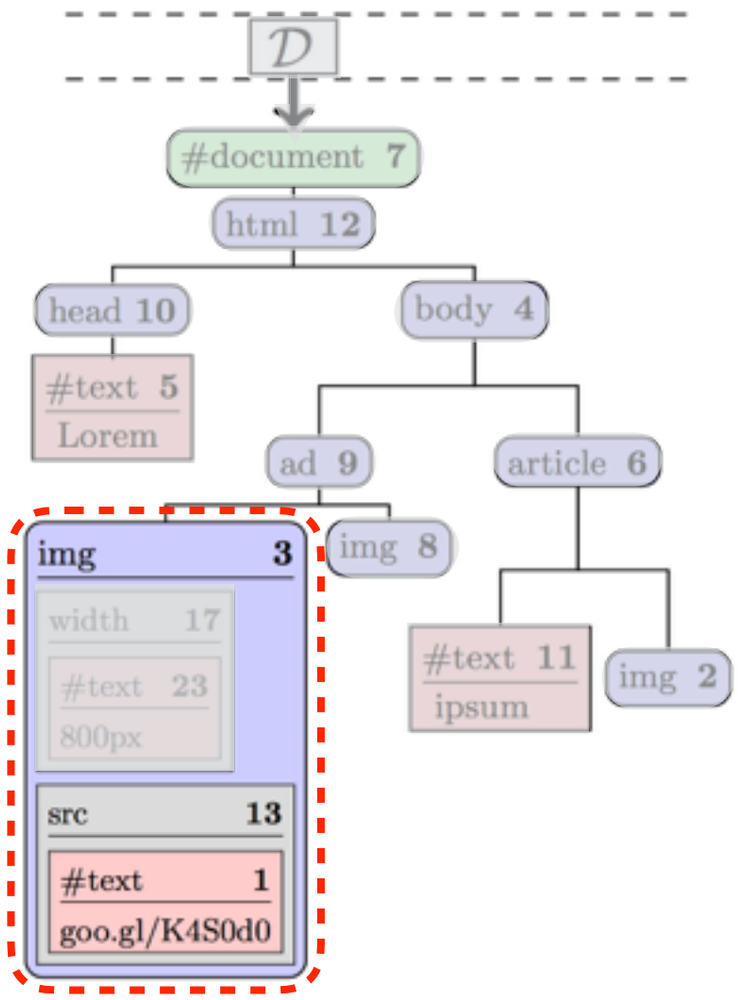


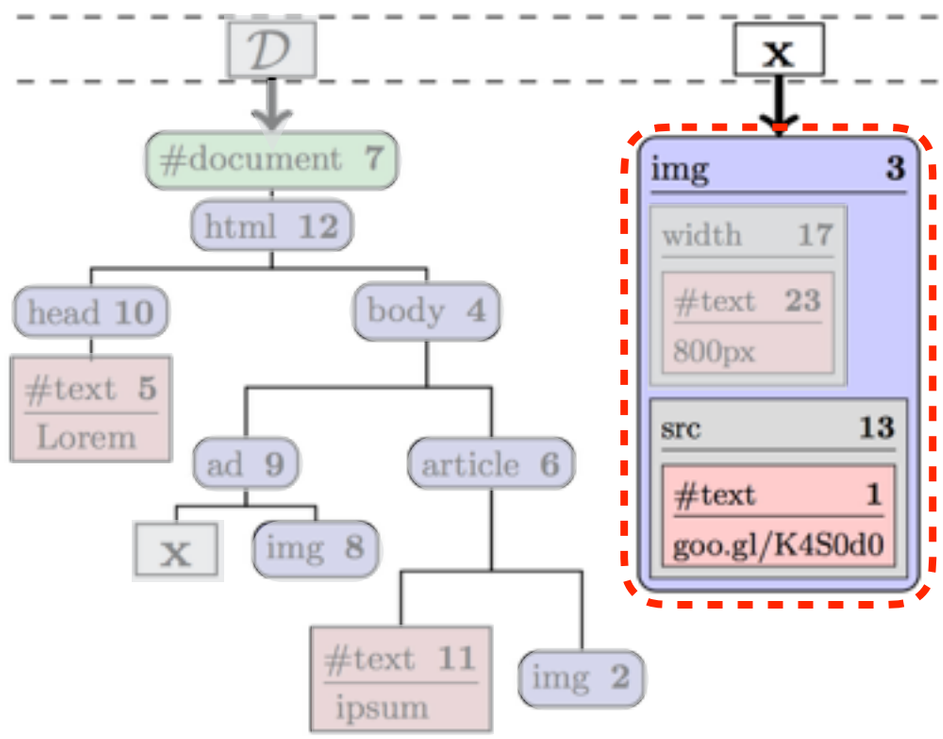
`n.getAttribute(s)`

footprint: element node **`n`** and its attribute named **`s`**

e.g. when **`n`**=3 and **`s`**="src" —> footprint in dashed box

Structural Separation Logic (SSL)





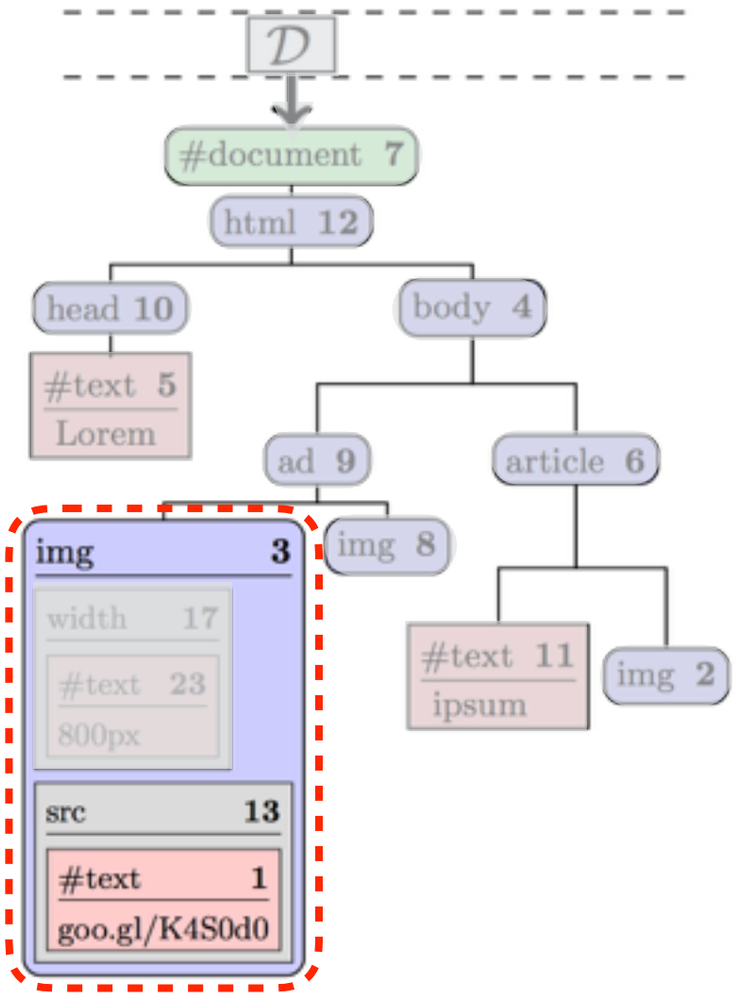
`n.getAttribute(s)`

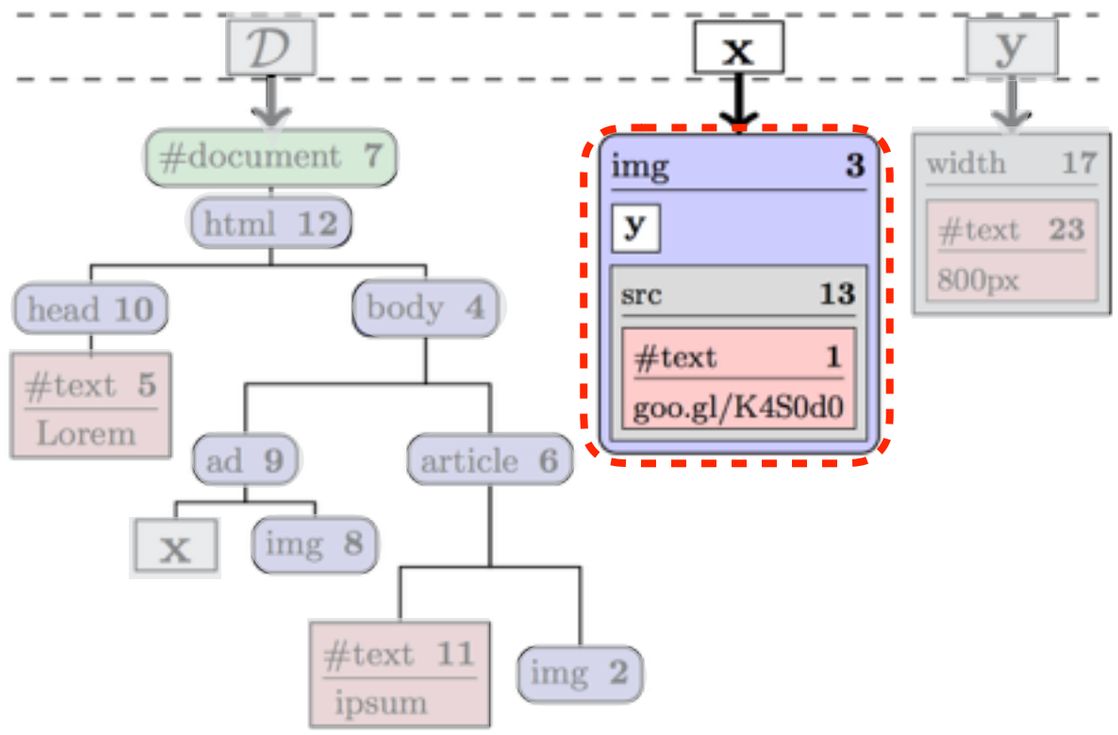
footprint: element node **`n`** and its attribute named **`s`**
e.g. when **`n`**=3 and **`s`**="src" —> footprint in dashed box

abstract allocation

split data, promote it to a fresh *abstract address* **x**, leave behind *context hole* **x**

Structural Separation Logic (SSL)





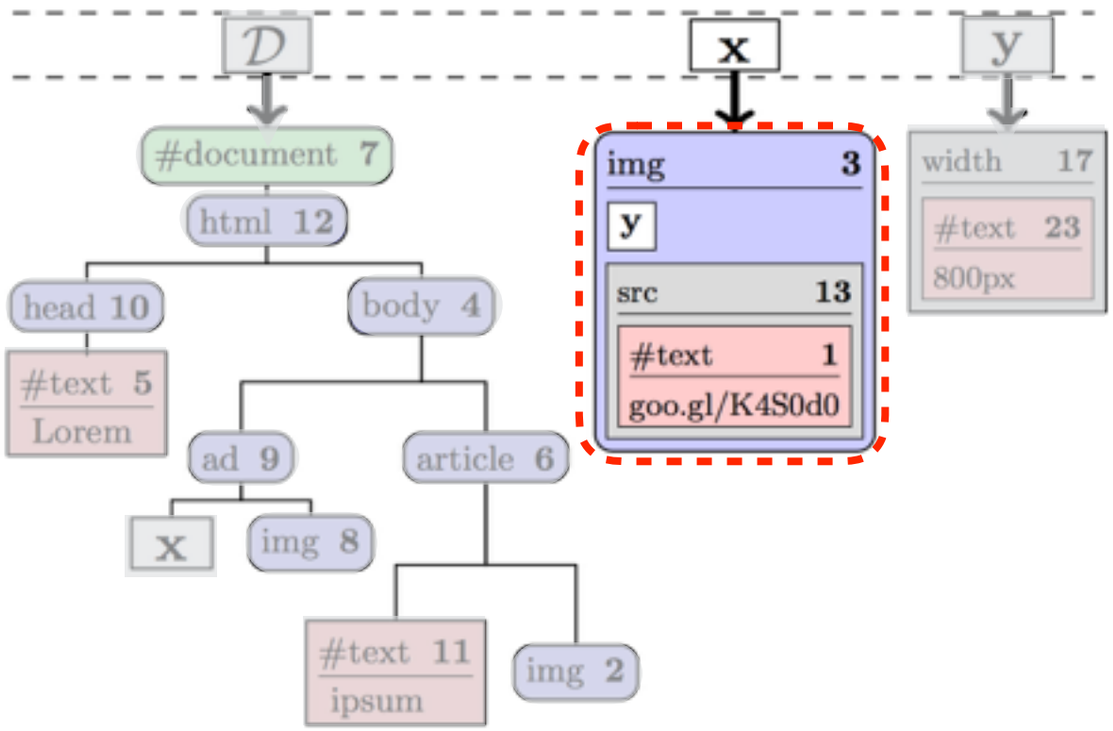
`n.getAttribute(s)`

footprint: element node **`n`** and its attribute named **`s`**
e.g. when **`n`**=3 and **`s`**="src" —> footprint in dashed box

abstract allocation

split data, promote it to a fresh *abstract address* **x**, leave behind *context hole* **x**

Structural Separation Logic (SSL)



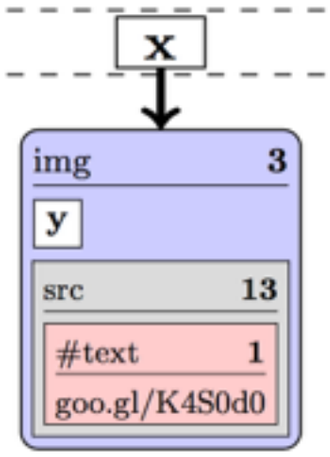
`n.getAttribute(s)`

footprint: element node **n** and its attribute named **s**
e.g. when **n**=3 and **s**="src" —> footprint in dashed box

abstract allocation

split data, promote it to a fresh *abstract address* **x**, leave behind *context hole* **x**

Structural Separation Logic (SSL)



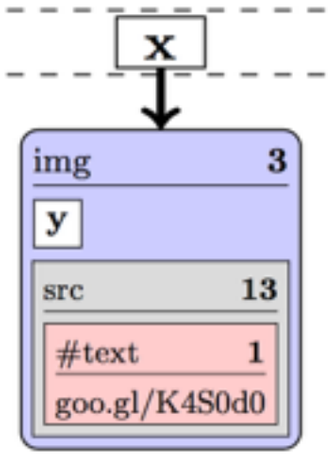
`n.getAttribute(s)`

footprint: element node **`n`** and its attribute named **`s`**
e.g. when **`n`**=3 and **`s`**="src" —> footprint in dashed box

abstract allocation

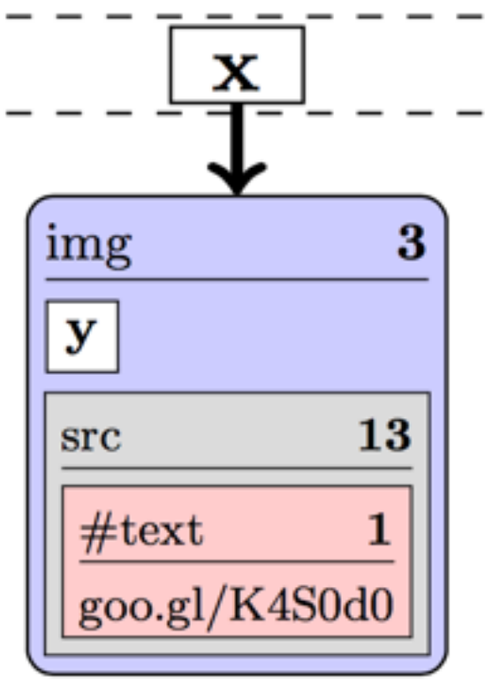
split data, promote it to a fresh *abstract address* **x**, leave behind *context hole* **x**

DOM Assertions in SSL



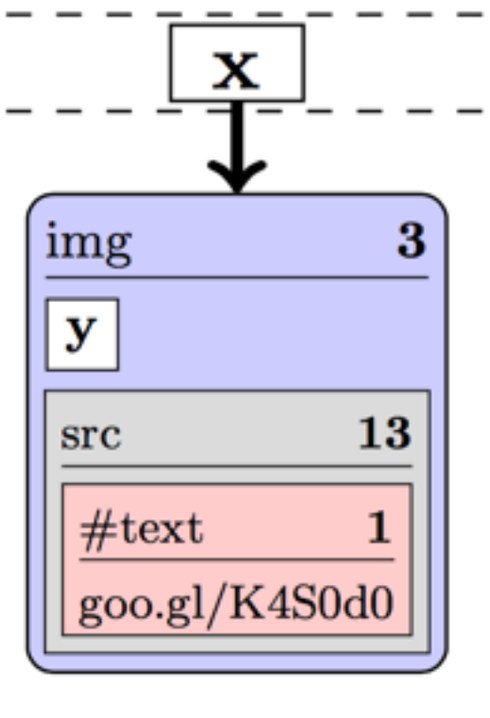
DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



DOM Assertions in SSL

DOM SSL Model
(abstract heaps)

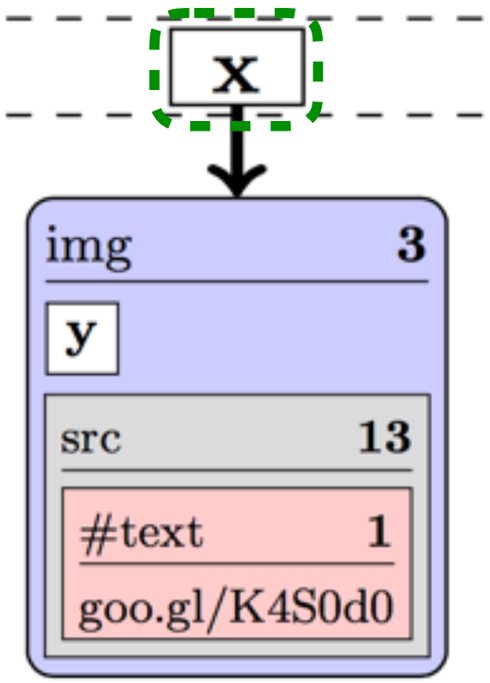


DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



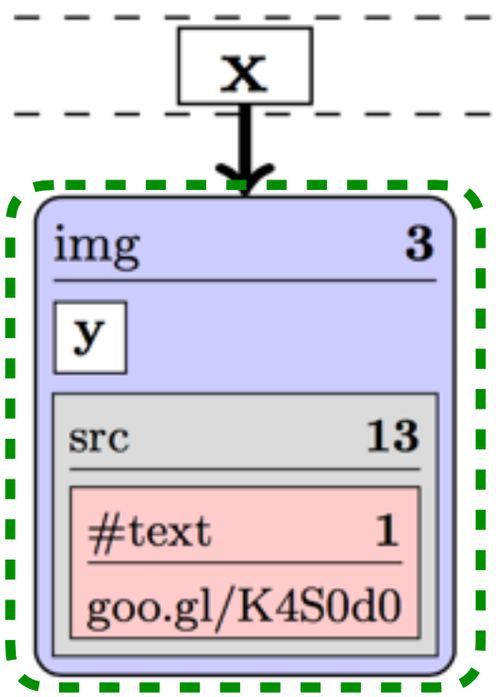
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

logical variable describing abstract address **x**

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



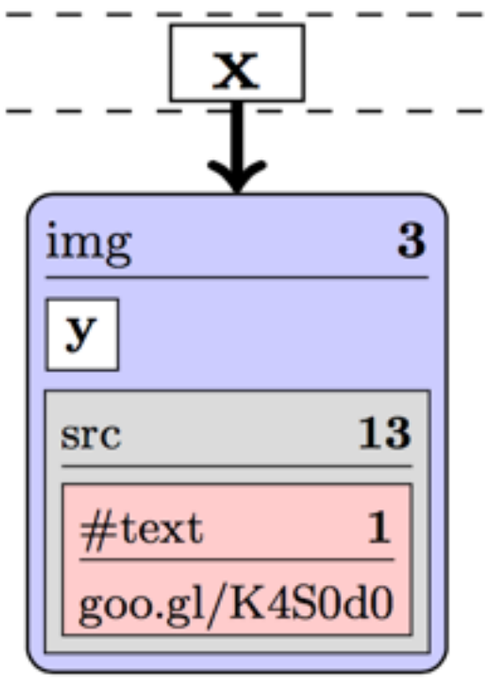
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

element node assertion

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



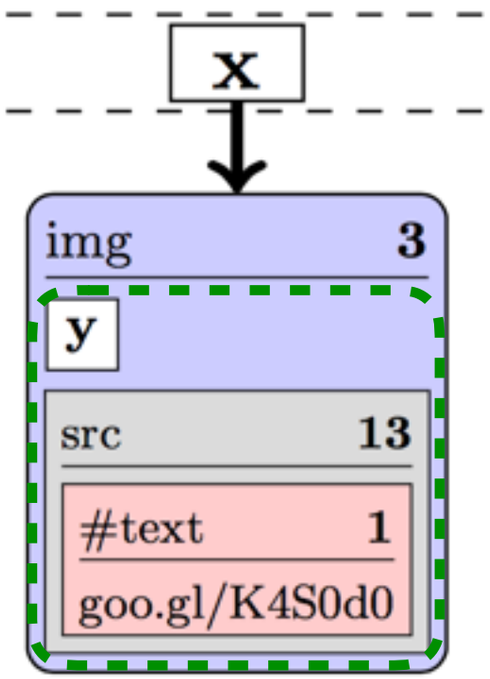
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

empty child forest assertion

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



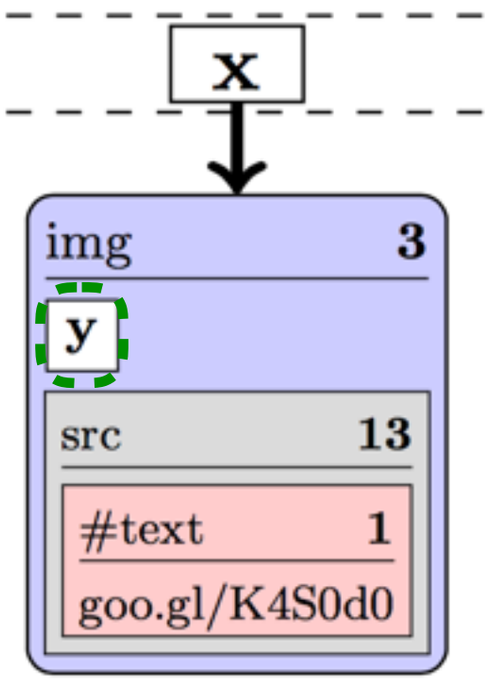
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

attribute set assertion

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



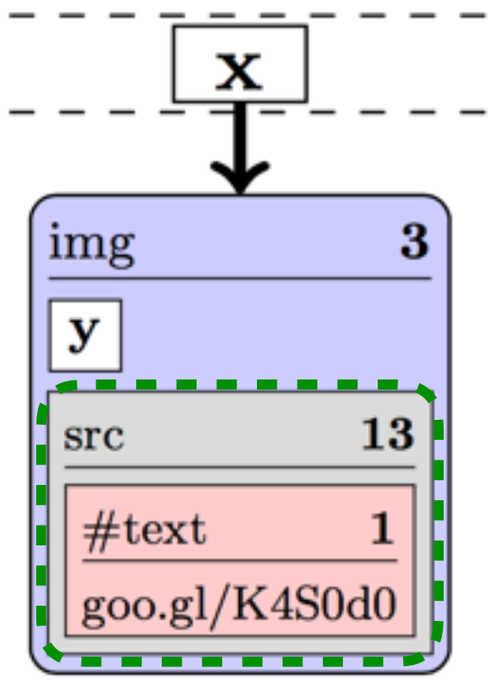
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

logical variable describing context hole **y**

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



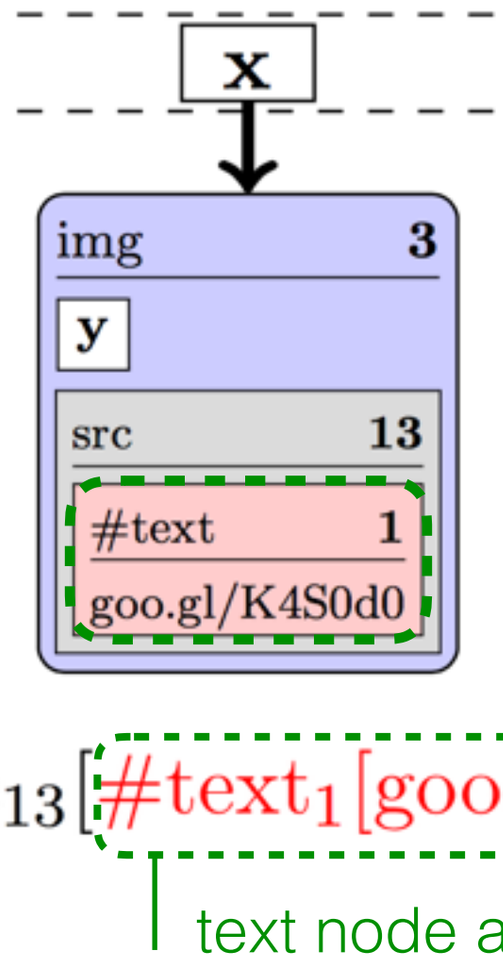
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

attribute node assertion

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



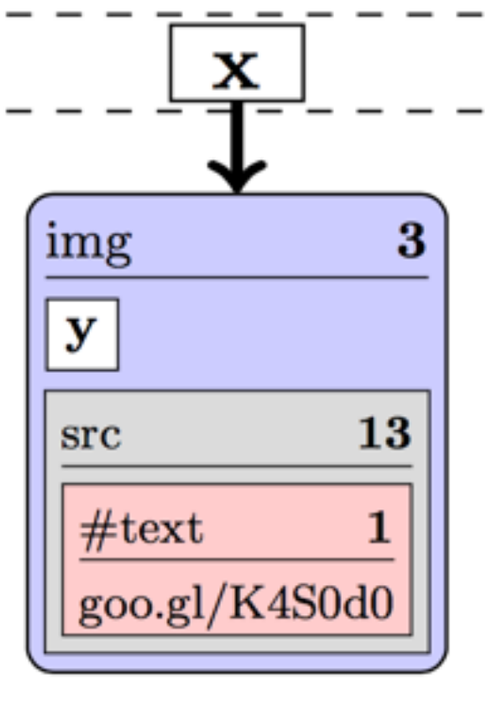
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

text node assertion

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)

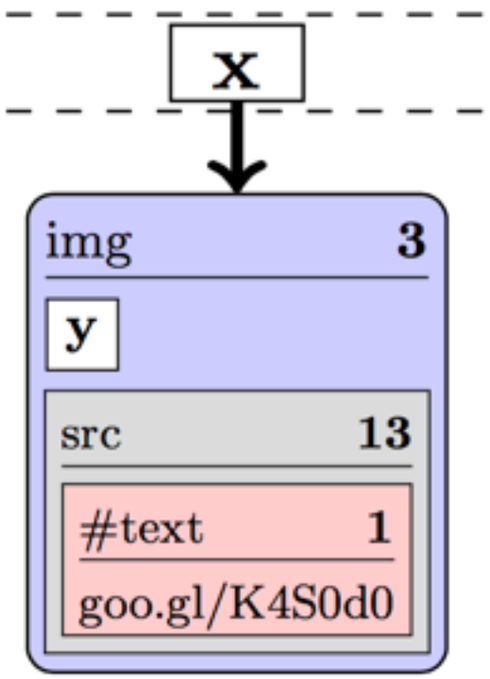


DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)



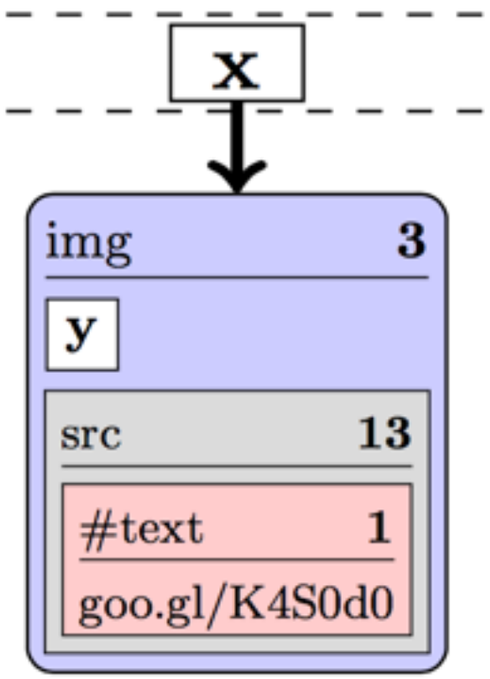
DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\#\mathrm{text}_1[\mathrm{goo.gl/K4S0d0}]], \varnothing]$$

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}], \varnothing] \ * \ \mathsf{val}(\mathrm{T}, \mathrm{goo.gl/K4S0d0})$$

DOM Assertions in SSL

DOM SSL Model
(abstract heaps)

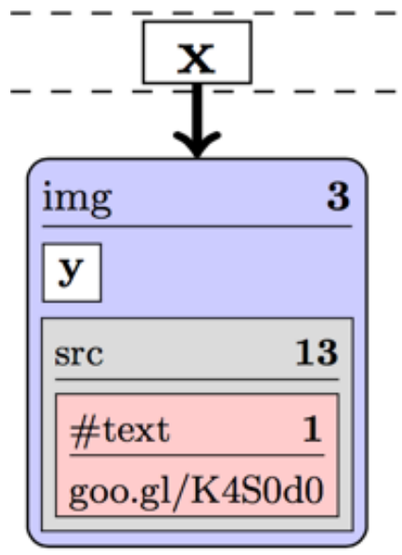


DOM SSL Assertions

$$\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}], \varnothing] \;*\; \mathsf{val}(\mathrm{T}, \mathrm{goo.gl/K4S0d0})$$

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

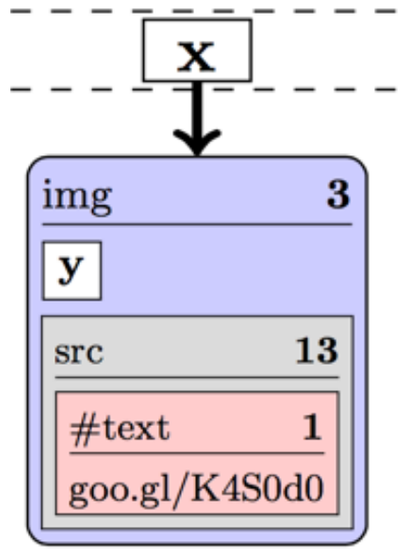
$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\mathrm{R}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

$$\mathtt{r = n.getAttribute(s)}$$

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\text{“goo.gl/K4S0d0”}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

DOM assertion
(operation footprint)

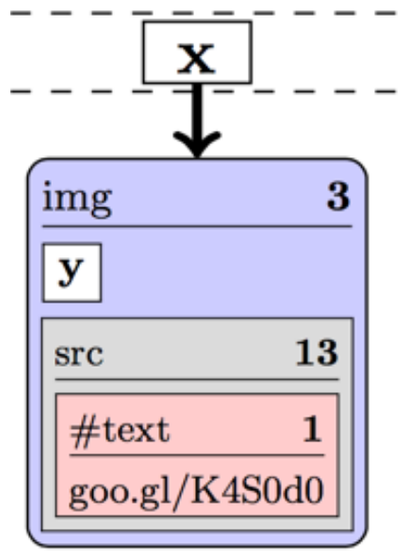
$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\mathrm{R}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

`r=n.getAttribute(s)`

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\text{“goo.gl/K4S0d0”}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

variable store assertion

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\mathrm{R}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

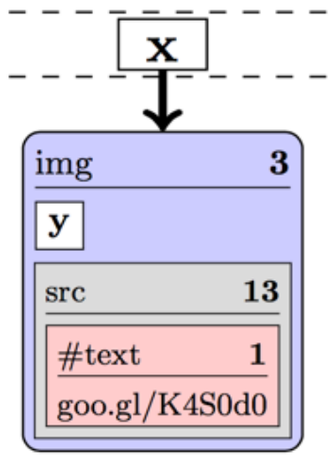
$$\mathtt{r = n.getAttribute(s)}$$

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\text{“goo.gl/K4S0d0”}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

store(…): *black-box* predicate; language-agnostic

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\mathrm{R}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

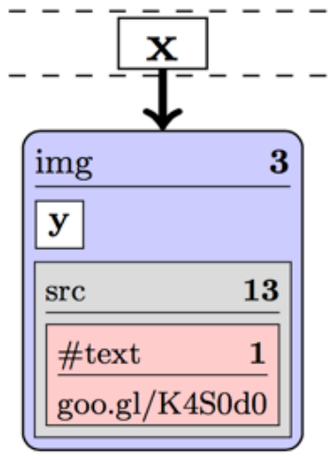
$$\mathtt{r = n.getAttribute(s)}$$

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\text{“goo.gl/K4S0d0”}) \\ *\,\alpha \mapsto \mathrm{img}_3[\beta \odot \mathrm{src}_{13}[\mathrm{T}],\varnothing] * \mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

store(…): *black-box* predicate; language-agnostic

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\mathrm{R})\\ *\,\alpha\mapsto \mathrm{img}_3[\beta\odot \mathrm{src}_{13}[\mathrm{T}],\varnothing]*\mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

$$\mathtt{r=n.getAttribute(s)}$$

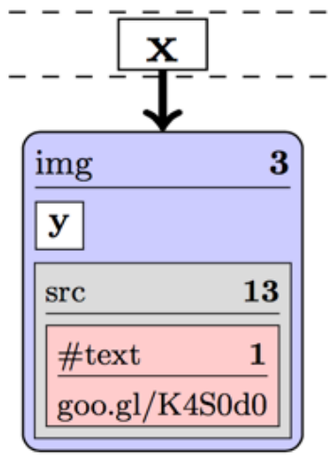
$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:3,\mathtt{s}:\text{“src”},\mathtt{r}:\text{“goo.gl/K4S0d0”})\\ *\,\alpha\mapsto \mathrm{img}_3[\beta\odot \mathrm{src}_{13}[\mathrm{T}],\varnothing]*\mathsf{val}(\mathrm{T},\mathrm{goo.gl/K4S0d0})\end{array}\right\}$$

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:-)\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\beta\odot \mathrm{S}_{\mathrm{M}}[\mathrm{T}],\gamma]\\ *\,\mathsf{val}(\mathrm{T},\mathrm{S}'')\end{array}\right\}\ \mathtt{r=n.getAttribute(s)}\ \left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:\mathrm{S}'')\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\beta\odot \mathrm{S}_{\mathrm{M}}[\mathrm{T}],\gamma]\\ *\,\mathsf{val}(\mathrm{T},\mathrm{S}'')\end{array}\right\}$$

store(…): *black-box* predicate; language-agnostic
interaction point between the language and DOM

DOM Specification in SSL

DOM SSL Model
(abstract heaps)



DOM SSL Specification

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:-)\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\beta\odot \mathrm{S}_{\mathrm{M}}[\mathrm{T}],\gamma]\\ *\,\mathsf{val}(\mathrm{T},\mathrm{S}'')\end{array}\right\}\ \mathtt{r=n.getAttribute(s)}\ \left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:\mathrm{S}'')\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\beta\odot \mathrm{S}_{\mathrm{M}}[\mathrm{T}],\gamma]\\ *\,\mathsf{val}(\mathrm{T},\mathrm{S}'')\end{array}\right\}$$

$$\left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:-)\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\mathrm{A},\gamma]*\mathsf{out}(\mathrm{A},\mathrm{S})\end{array}\right\}\ \mathtt{r=n.getAttribute(s)}\ \left\{\begin{array}{l}\mathsf{store}(\mathtt{n}:\mathrm{N},\mathtt{s}:\mathrm{S},\mathtt{r}:\text{“”})\\ *\,\alpha\mapsto \mathrm{S}'_{\mathrm{N}}[\mathrm{A},\gamma]*\mathsf{out}(\mathrm{A},\mathrm{S})\end{array}\right\}$$

store(…): *black-box* predicate; language-agnostic
interaction point between the language and DOM

DOM Specification Wish List

✔ **faithful**; **axiomatic**; **abstract**

✔ **local**

? **compositional** (client program spec from DOM spec)

? Easily **integrated** with existing program logics

DOM Specification Wish List

✔ **faithful**; **axiomatic**; **abstract**

✔ **local**

? **compositional** (client program spec from DOM spec)

? Easily **integrated** with existing program logics

CL **Non-Compositionality**

```
C  :  r1 = n.getAttribute(“src”);
      r2 = m.getAttribute(“src”);
      r3 = o.getAttribute(“src”);
       r = r1 + r2 + r3;
```

CL **Non-Compositionality**

```
C  :  r1 = n.getAttribute(“src”);
      r2 = m.getAttribute(“src”);
      r3 = o.getAttribute(“src”);
       r = r1 + r2 + r3;
```

Previous work: At least **6** CL specifications needed!

CL **Non-Compositionality**

```
C  :  r1 = n.getAttribute(“src”);
      r2 = m.getAttribute(“src”);
      r3 = o.getAttribute(“src”);
       r = r1 + r2 + r3;
```

Previous work: At least **6** CL specifications needed!

This work: Only **1** SSL specification

DOM Specification Wish List

✔ **faithful**; **axiomatic**; **abstract**

✔ **local**

✔ **compositional** (client program spec from DOM spec)

? Easily **integrated** with existing program logics

DOM Specification Wish List

✔ **faithful**; **axiomatic**; **abstract**

✔ **local**

✔ **compositional** (client program spec from DOM spec)

? Easily **integrated** with existing program logics

DOM (SSL) Specification Integration

- General methodology for extending SL-based logics with DOM spec
  - Given a language PL and its SL-based program logic PLLogic*:
    extend PL to PLDOM (add DOM operations)
    extend the logic to PLDOMLogic:

$$\frac{(P,\ \mathrm{C},\ Q) \in \text{DOMAxiom}}{\{P\}\ \mathrm{C}\ \{Q\}}$$

DOM (SSL) Specification Integration

- General methodology for extending SL-based logics with DOM spec
  - Given a language PL and its SL-based program logic PLLogic*:
    extend PL to PLDOM (add DOM operations)
    extend the logic to PLDOMLogic:

$$\frac{(P,\ \mathrm{C},\ Q) \in \text{DOMAxiom}}{\{P\}\ \mathrm{C}\ \{Q\}}$$

* PLLogic must meet certain conditions (e.g. **store** predicate)

DOM (SSL) Specification Integration

- General methodology for extending SL-based logics with DOM spec
  - Given a language PL and its SL-based program logic PLLogic*:
    extend PL to PLDOM (add DOM operations)
    extend the logic to PLDOMLogic:

$$\frac{(P,\ \mathrm{C},\ Q) \in \text{DOMAxiom}}{\{P\}\ \mathrm{C}\ \{Q\}}$$

  - Can extend PLLogic with *any* SSL-specified library
  - Integrated DOM spec with JSLogic (JavaScript Program Logic - POPL'12)
  - Verified multiple ad blocker scripts in JavaScript

* PLLogic must meet certain conditions (e.g. **store** predicate)

Future/Ongoing Work

Future/Ongoing Work

- Extend to full Core — straightforward as with Smith et al.

Future/Ongoing Work

- Extend to full Core — straightforward as with Smith et al.
- Extend to levels 2-4 — more involved:
  - requires higher-order reasoning (DOM Events)

Future/Ongoing Work

- Extend to full Core — straightforward as with Smith et al.
- Extend to levels 2-4 — more involved:
  - requires higher-order reasoning (DOM Events)
- DOM reasoning tool
  - ongoing work: DOM+JavaScript semi-automatic verification tool

Conclusions

- A DOM specification that is:
  - **faithful**; **axiomatic**; **abstract**
  - **local**
  - **compositional**
  - Easily **integrated** with existing (SL-based) program logics

- General methodology for extending SL-based logics with DOM spec
  - Integrated DOM spec with JSLogic (JavaScript Program Logic - POPL'12)
  - SSL as an add-on
  - Verified multiple ad blocker scripts in JavaScript

Conclusions

- A DOM specification that is:
  - **faithful**; **axiomatic**; **abstract**
  - **local**
  - **compositional**
  - Easily **integrated** with existing (SL-based) program logics

- General methodology for extending SL-based logics with DOM spec
  - Integrated DOM spec with JSLogic (JavaScript Program Logic - POPL'12)
  - SSL as an add-on
  - Verified multiple ad blocker scripts in JavaScript

Thank you for listening!