Revisiting Recency Abstraction for JavaScript

Towards an Intuitive, Compositional, and Efficient Heap Abstraction

Singleton Abstraction

Jihyeok Park — KAIST

Xavier Rival — DIENS, ÉNS, CNRS, PSL Research University and INRIA

Sukyoung Ryu — KAIST

Static Analysis for JavaScript

JavaScript

- *de facto* language for **web programming**
- **static analyzers** based on abstract interpretation
    - SAFE / TAJS / WALA
- precise analysis of **object properties**

Object Properties

```
var o = {a : 1};
```

- dynamic addition and removal of object properties

```
o.b = 2;      // {a : 1, b : 2}
delete o.a; // {b : 2}
```

- first-class property names

```
var v = ‘p’;
o[v+’q’];     // === o.pq
```

- higher-order functions

```
o.f = function() {};
o.f();          // indirect call
```

Weak vs **Strong** Update

```
var o = {};
o.p = 1;
o.p = 2;
```

- **strong update**

```
o = {
   p: 2
}
```

- **weak update**

```
o = {
   p: ∗,1,2
}
```

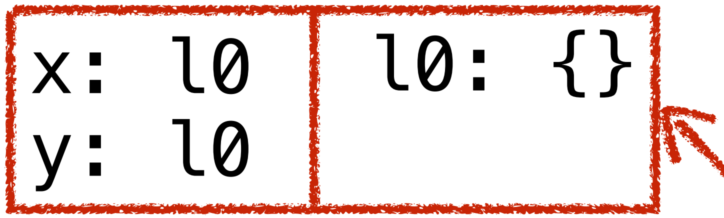
(∗ : absent value)

Allocation-site Abstraction

Allocation Site

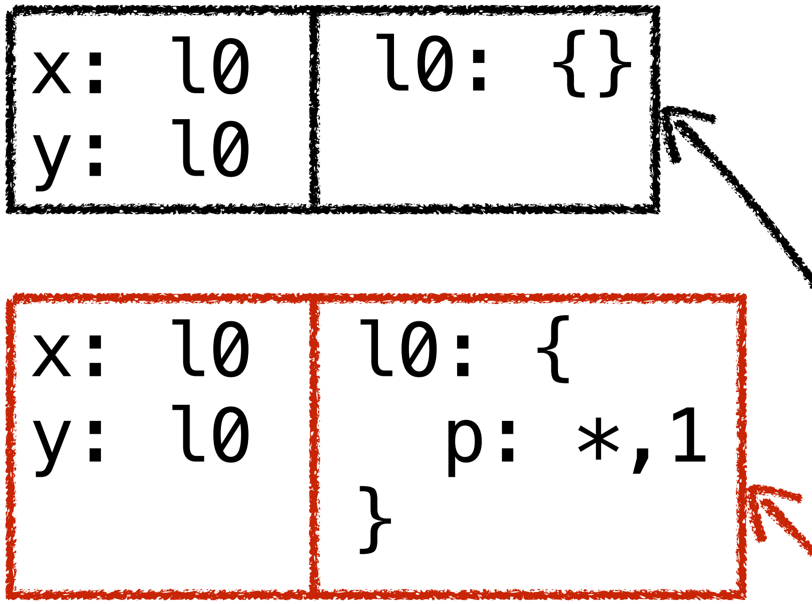
```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Allocation-site Abstraction




```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

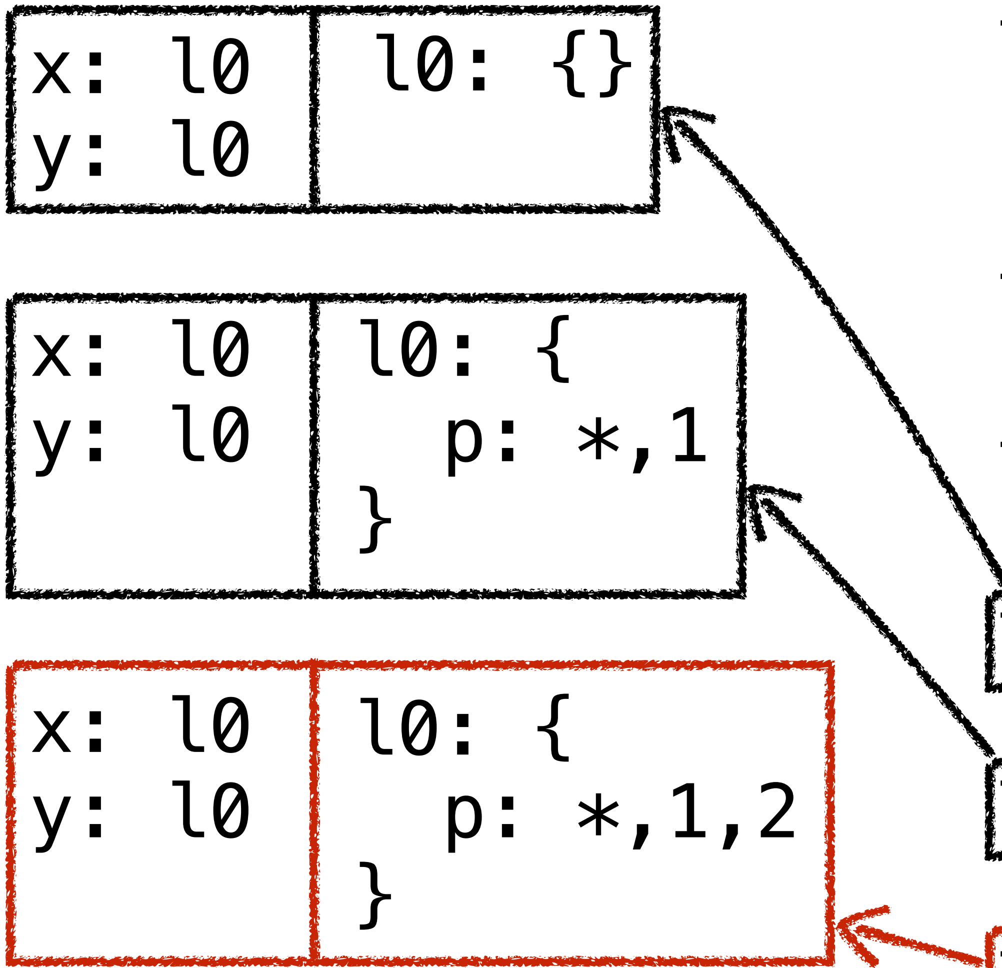
Allocation-site Abstraction



Weak Update

```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Allocation-site Abstraction



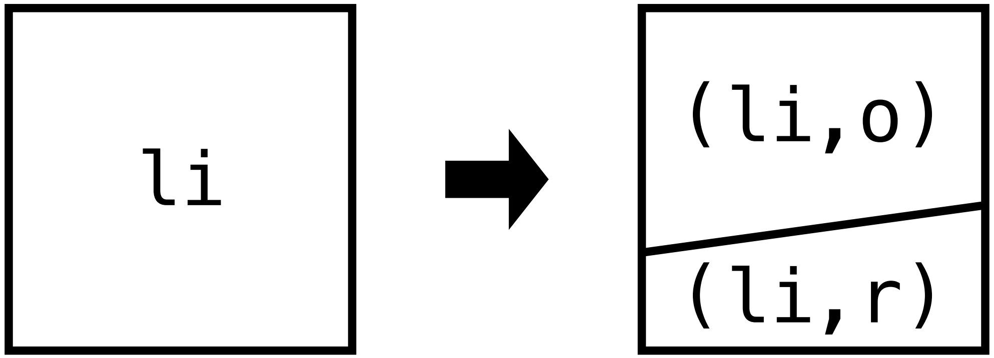

```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Weak Update

Recency Abstraction

Recency Abstraction

- defined on top of the **allocation-site abstraction**
  - **recent** : `(li,r)` with **strong updates**
    - most recently created objects
  - **old** : `(li,o)` with **weak updates**
    - not recent locations



Recency Abstraction

```
x: (l0,o) | (l0, o): {}
y: (l0,r) | (l0, r): {}
```

```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Recency Abstraction

```
x: (l0,o) | (l0, o): {}
y: (l0,r) | (l0, r): {}
```

```
x: (l0,o) | (l0,o): {
y: (l0,r) |   p: *,1
          | }
          | (l0,r): {}
```

Weak Update

```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Recency Abstraction

```
x: (l0,o) | (l0, o): {}
y: (l0,r) | (l0, r): {}
```

```
x: (l0,o) | (l0,o): {
y: (l0,r) |    p: *,1
          | }
          | (l0,r): {
          |    p: 2
          | }
```

Strong Update

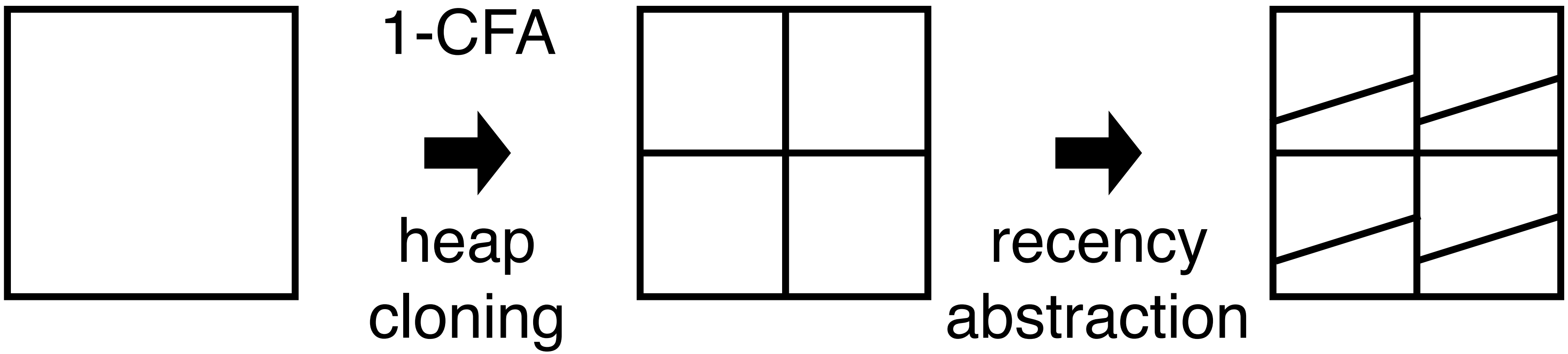
```
l0: function f()
    { return {}; };

l1: var x = f();

l2: var y = f();

l3: x.p = 1;

l4: y.p = 2;

l5: x.p + y.p;
```

Recency Abstraction

- allocation-sites with **heap cloning** (with **sensitivities**)

```
function f(x) {
  return {p: x};
}
```

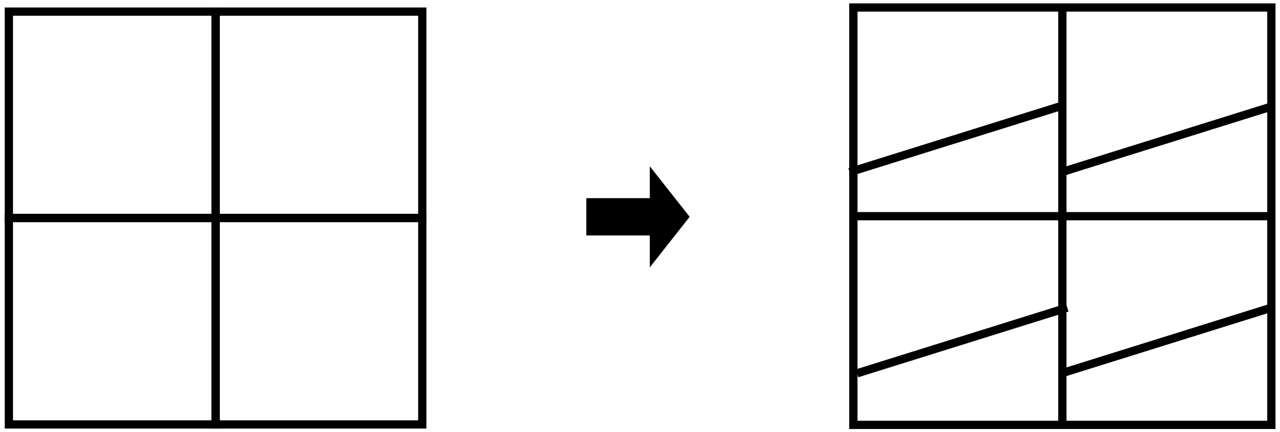
```
f(1); f(2);
f(3); f(4);
```



Recency Abstraction

A given partition

$$\delta : \mathbb{A} \rightarrow \Pi$$



$$\mathbb{A}^{\sharp}_{\delta} = \mathcal{P}(\Pi) \qquad \mathbb{A}^{\sharp}_{\mathbf{r}[\delta]} = \mathcal{P}(\Pi \times \{\mathbf{r}, \mathbf{o}\});$$

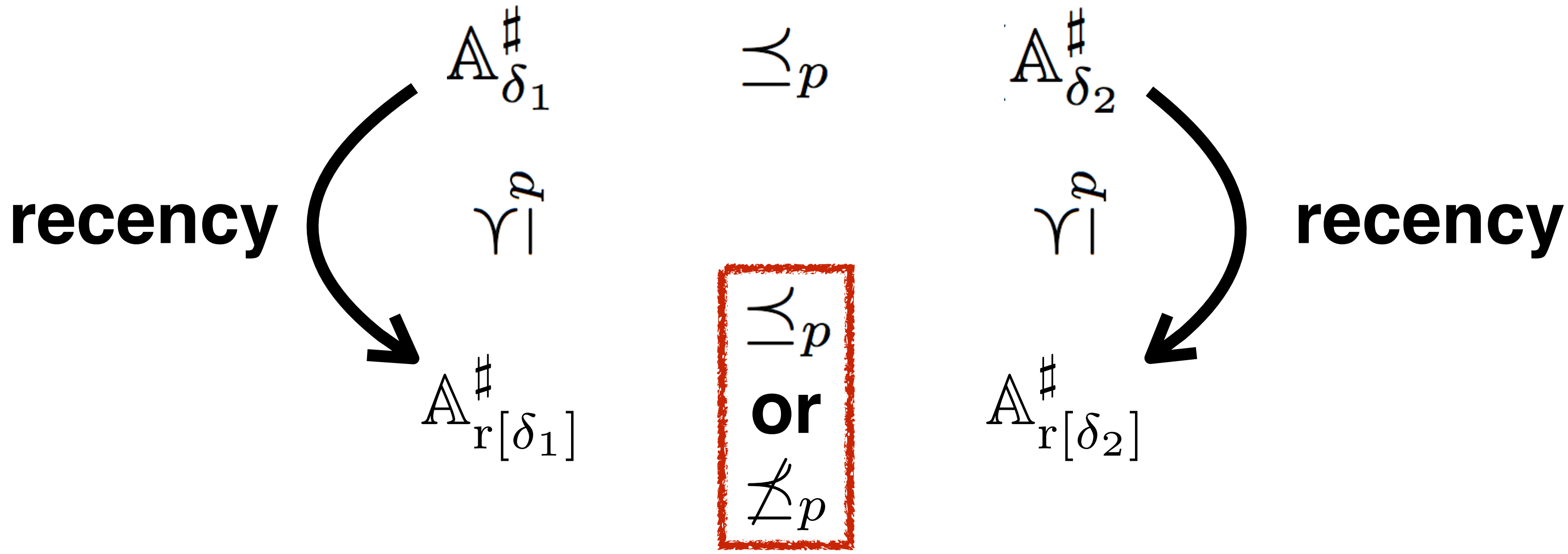
partition-based address abstraction

recency abstraction

Unintuitive Behaviors of Recency Abstraction

Unintuitive Behaviors

- Recency abstraction does **not preserve** the **precision relationship** between given partition-based address abstractions



* $\preceq_p$: precision relationship

Example 1

ℓ_0 : **var** `obj` $= \{\};$
ℓ_1 : **if** $(\ ?\ )\ \{$
ℓ_2 : `obj.a` $= 1;$
ℓ_3 : `obj` $= \{\};$
ℓ_4 : $\}$

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$ **where** $\delta_\top : \mathbb{A} \rightarrow \{\pi\}$

	$e^\sharp$	$h^\sharp$
true branch	`obj` $\mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\mathsf{a} \mapsto \{1\}\}$
false branch	`obj` $\mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$
join	`obj` $\mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\mathsf{a} \mapsto \{1\}\}$

$$\mathbb{A}^\sharp_{\delta_{id}} \preceq_p \mathbb{A}^\sharp_{\delta_\top}$$

$$\preceq_p \qquad \preceq_p$$

$$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]} \not\preceq_p \mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$$

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]}$ **where** $\delta_{id} : \mathbb{A} \rightarrow \mathbb{L}$

	$e^\sharp$	$h^\sharp$
true branch	`obj` $\mapsto \{(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\mathsf{a} \mapsto \{1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$
false branch	`obj` $\mapsto \{(\ell_0, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\}$
join	`obj` $\mapsto \{(\ell_0, \mathbf{r}),$ $(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\mathsf{a} \mapsto \{\circledast, 1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$

Example 1

ℓ_0 : **var** obj $= \{\}$;
ℓ_1 : **if** (?) {
ℓ_2 : obj.a $= 1$;
ℓ_3 : obj $= \{\}$;
ℓ_4 : }

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$ where $\delta_\top : \mathbb{A} \to \{\pi\}$

	$e^\sharp$	$h^\sharp$
true branch	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\texttt{a} \mapsto \{1\}\}$
false branch	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$
join	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\texttt{a} \mapsto \{1\}\}$

$$\mathbb{A}^\sharp_{\delta_{id}} \preceq_p \mathbb{A}^\sharp_{\delta_\top}$$

$$\preceq_p \qquad \preceq_p$$

$$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]} \not\preceq_p \mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$$

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]}$ where $\delta_{id} : \mathbb{A} \to \mathbb{L}$

	$e^\sharp$	$h^\sharp$
true branch	$\texttt{obj} \mapsto \{(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\texttt{a} \mapsto \{1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$
false branch	$\texttt{obj} \mapsto \{(\ell_0, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\}$
join	$\texttt{obj} \mapsto \{(\ell_0, \mathbf{r}),$ $(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\texttt{a} \mapsto \{\circledast, 1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$

Example 1

```
ℓ0 :  var obj = {};
ℓ1 :  if ( ? ) {
ℓ2 :      obj.a = 1;
ℓ3 :      obj = {};
ℓ4 :  }
```

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$ where $\delta_\top : \mathbb{A} \to \{\pi\}$

	$e^\sharp$	$h^\sharp$
true branch	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\texttt{a} \mapsto \{1\}\}$
false branch	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$
join	$\texttt{obj} \mapsto \{(\pi, \mathbf{r})\}$	$(\pi, \mathbf{r}) \mapsto \{\}$ $(\pi, \mathbf{o}) \mapsto \{\texttt{a} \mapsto \{1\}\}$

$$\mathbb{A}^\sharp_{\delta_{id}} \preceq_p \mathbb{A}^\sharp_{\delta_\top}$$

$$\preceq_p \qquad\qquad \preceq_p$$

$$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]} \not\preceq_p \mathbb{A}^\sharp_{\mathrm{r}[\delta_\top]}$$

$\mathbb{A}^\sharp_{\mathrm{r}[\delta_{\mathrm{id}}]}$ where $\delta_{id} : \mathbb{A} \to \mathbb{L}$

	$e^\sharp$	$h^\sharp$
true branch	$\texttt{obj} \mapsto \{(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\texttt{a} \mapsto \{1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$
false branch	$\texttt{obj} \mapsto \{(\ell_0, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\}$
join	$\texttt{obj} \mapsto \{(\ell_0, \mathbf{r}),$ $(\ell_3, \mathbf{r})\}$	$(\ell_0, \mathbf{r}) \mapsto \{\texttt{a} \mapsto \{\circledast, 1\}\}$ $(\ell_3, \mathbf{r}) \mapsto \{\}$

Example 2

ℓ_0 :	`function g(z){`
ℓ_1 :	`    var result = z.p;`
ℓ_2 :	`}`
ℓ_3 :	`function f(){`
ℓ_4 :	`    var obj = {};`
ℓ_5 :	`    var a = g(obj);`
ℓ_6 :	`    obj.p = 3;`
ℓ_7 :	`    return obj;`
ℓ_8 :	`}`
ℓ_9 :	`var x = f();`
ℓ_{10} :	`var y = f();`
ℓ_{11} :	

allocation-site + 0-CFA

$$\delta_0 : \mathbb{A} \to \{\ell_4\}$$

allocation-site + 1-CFA

$$\delta_1 : \mathbb{A} \to \{\ell_{4/9}, \ell_{4/10}\}$$

$$\mathbb{A}^\sharp_{\delta_1} \preceq_p \mathbb{A}^\sharp_{\delta_0}$$

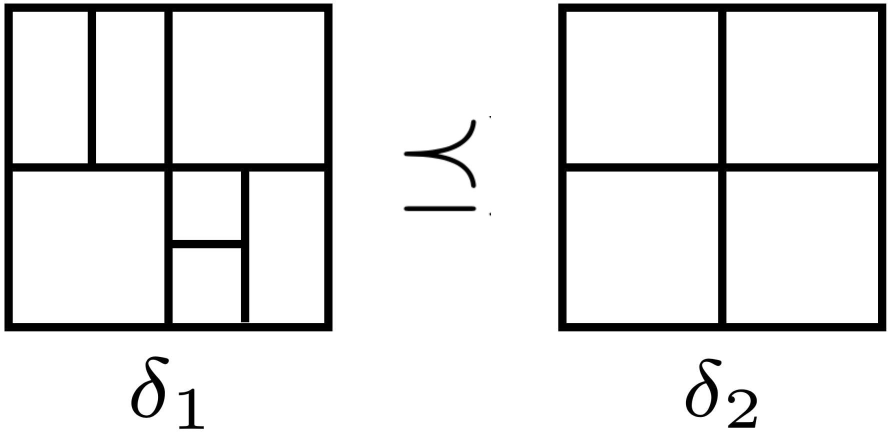
$$\preceq_p \qquad\qquad \preceq_p$$

$$\mathbb{A}^\sharp_{\mathrm{r}[\delta_1]} \not\preceq_p \mathbb{A}^\sharp_{\mathrm{r}[\delta_0]}$$

Why?

- refinement relationship

$$\mathbb{A}^{\sharp}_{\delta_1} \preceq \mathbb{A}^{\sharp}_{\delta_2} \; \textit{iff } \delta_1 \textit{ is a refinement partition of } \delta_2$$

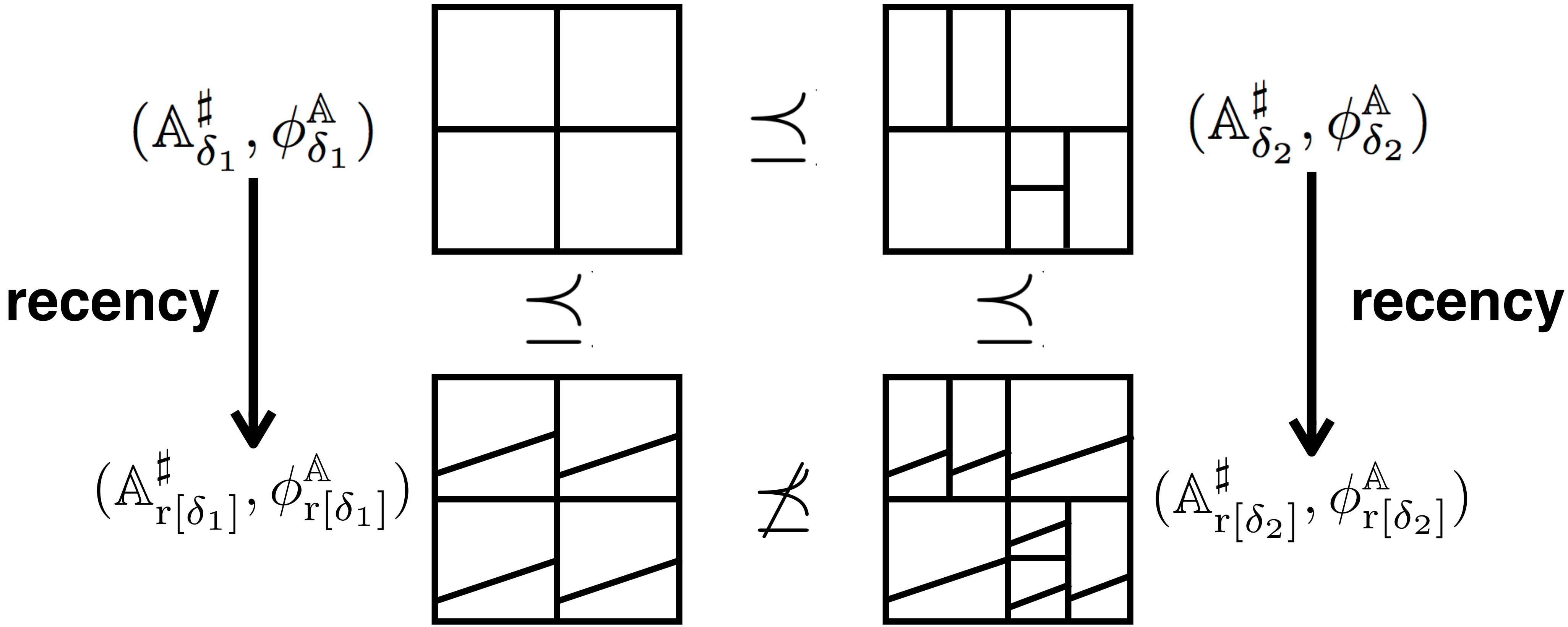


Theorem 1 (Implication of precision from refinement)**.**

$$\mathbb{A}^{\sharp}_{\delta_1} \preceq \mathbb{A}^{\sharp}_{\delta_2} \Rightarrow \mathbb{A}^{\sharp}_{\delta_1} \preceq_p \mathbb{A}^{\sharp}_{\delta_2}$$

Why?

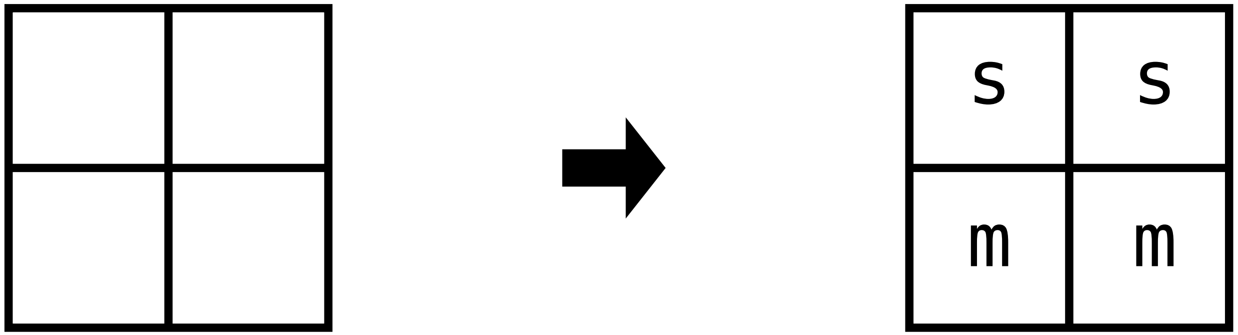
- Recency abstraction does **not preserve** the **refinement relationship** between given partition-based address abstractions



Singleton Abstraction

Singleton Abstraction

A given partition $\delta : \mathbb{A} \rightarrow \Pi$



$$\mathbb{A}^{\sharp}_{\delta} = \mathcal{P}(\Pi) \qquad \mathbb{H}^{\sharp}_{\mathbf{s}[\delta]} = \Pi \longrightarrow \mathbb{O}^{\sharp} \times \{\mathbf{s}, \mathbf{m}\}$$

- **singleton(s) - strong updates**
  - exactly one object
- **multiple(m) - weak updates**
  - more than one objects

Evaluation

Evaluation

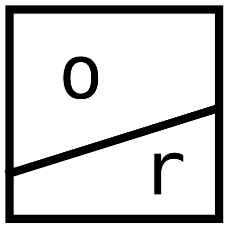
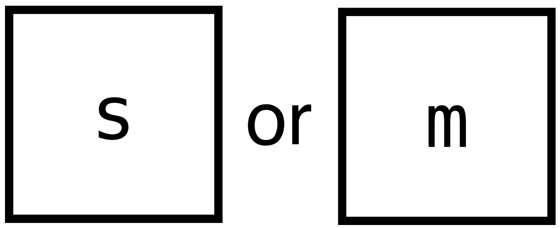
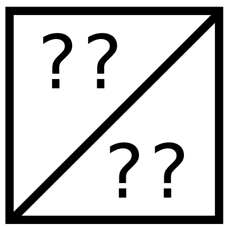
- **3 benchmarks** (24 programs)
  - JSAI, SunSpider, and V8
- **evaluation setting**
  - 2.8 GHz Intel Core i5 iMac with 16GB memory
- **Time**
  - Allocation-site Abstraction: 86.92 sec
  - Recency Abstraction: 122.73 sec
  - Singleton Abstraction: 79.77 sec

PLRG

- **Precision:**
  - # of properties more precise than allocation-site abstraction

Bench	Program	LOC	Recency	Singleton	Total
JSAI	adn-chess.js	234	90	55	127
	adn-coffee_pods_deals.js	367	45	37	141
	adn-less_spam_please.js	759	213	143	432
	adn-live_pagerank.js	882	132	117	323
	adn-odesk_job_watcher.js	168	56	52	71
	adn-pinpoints.js	548	58	57	232
	adn-tryagain.js	929	103	72	525
SunSpider	3d-morph.js	23	1	1	4
	access-binary-trees.js	38	14	10	16
	access-fannkuch.js	51	1	1	19
	access-nbody.js	142	32	15	78
	access-nsieve.js	28	2	0	4
	bitops-3bit-bits-in-byte.js	13	0	0	0
	bitops-bits-in-byte.js	14	0	0	0
	bitops-bitwise-and.js	3	0	0	0
	bitops-nsieve-bits.js	22	1	1	7
	controlflow-recursive.js	18	0	0	0
	math-cordic.js	53	4	4	6
	math-partial-sums.js	25	4	4	4
	math-spectral-norm.js	41	2	1	16
	string-fasta.js	70	15	10	18
V8	navier-stokes.js	331	36	17	92
	richards.js	288	119	117	197
	splay.js	205	108	108	132
Total			1036	831	2,444
Ratio (%)			42.39	33.63	–

Conclusion

Abstraction	more division	tags for strong update
Recency		recent(r) old(o)
Singleton		singleton(s) multiple(m)
Our Goal		singleton(s) multiple(m)