Correlation Tracking for Points-To Analysis of JavaScript

M. Sridharan, J. Dolby, S. Chandra, M. Schaefer, F. Tip

ECOOP 2012

Present by Dong Chen

The contributions of this paper

- The authors show that a standard implementation of field-sensitive Andersen’s points-to analysis extended to handle **dynamic property accesses** has O(N 4) **worst case** running time, in contrast to the O(N 3) bound for other languages.
- The authors present a technique to address **scalability issues** caused by dynamic property accesses
- The authors report on an implementation of their **correlation tracking technique** on top of WALA and its application to JavaScript frameworks

Dynamic Property Accesses

- Example from the paper:

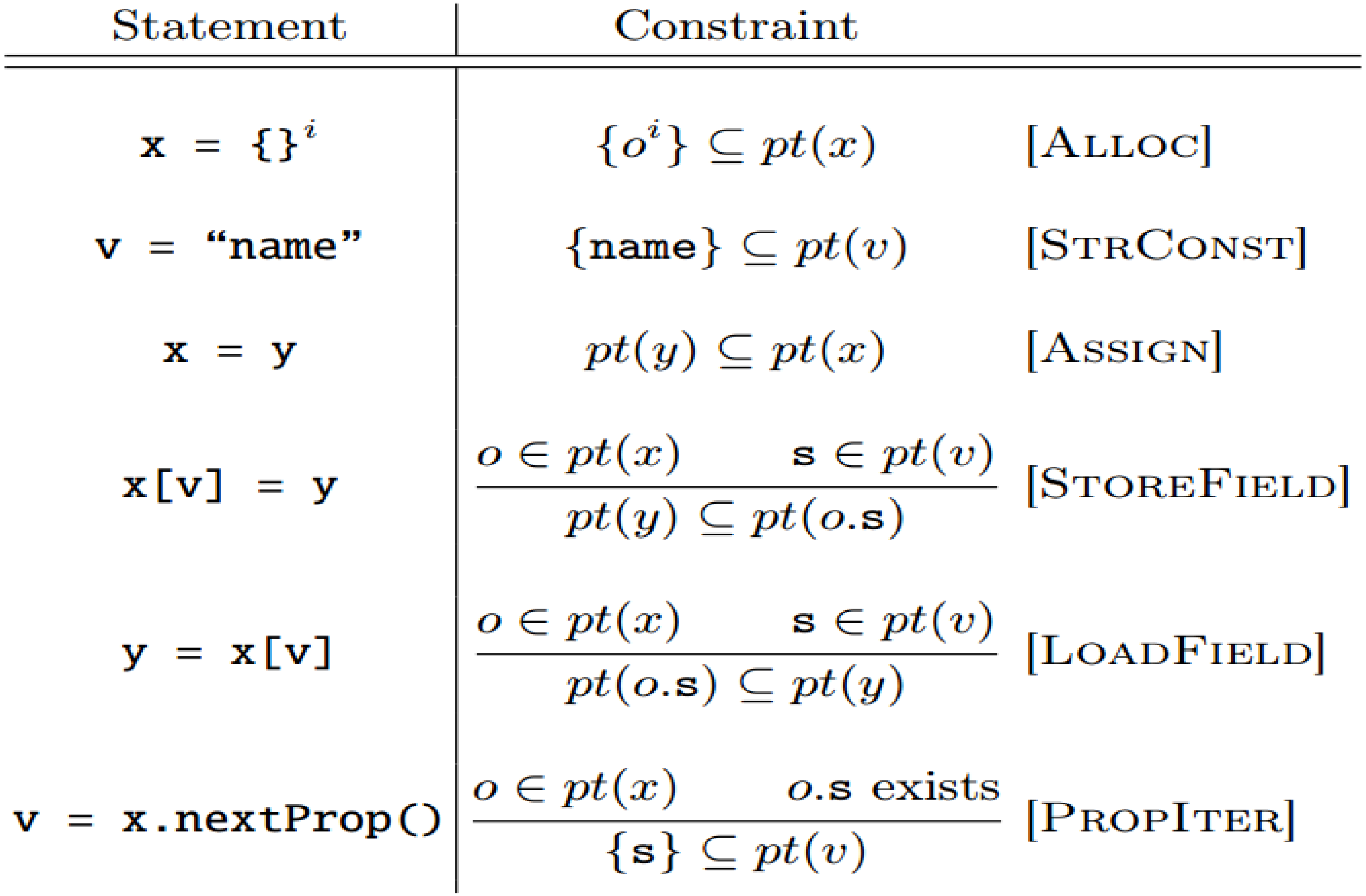
```
o.foo = function f1() { return 23; };
o.bar = function f2() { return 42; };
o.foo();

f = p(*) ? "foo" : "baz";
//writes to o.foo or o.baz
o[f] = "Hello , world!";
```

Correlated Pairs

- There is often an obvious correlation between the updated location and the stored value which is ignored by the points-to analysis.
- For example,

```
function extend(dest,src) {
    for (var prop in src)
        // correlated accesses
        dest[prop] = src[prop];
}
```

Andersen’s Analysis



Statement	Constraint	
`x = {}`^{i}	$\{o^i\} \subseteq pt(x)$	[ALLOC]
`v = “name”`	$\{\texttt{name}\} \subseteq pt(v)$	[STRCONST]
`x = y`	$pt(y) \subseteq pt(x)$	[ASSIGN]
`x[v] = y`	$\dfrac{o \in pt(x) \qquad \texttt{s} \in pt(v)}{pt(y) \subseteq pt(o.\texttt{s})}$	[STOREFIELD]
`y = x[v]`	$\dfrac{o \in pt(x) \qquad \texttt{s} \in pt(v)}{pt(o.\texttt{s}) \subseteq pt(y)}$	[LOADFIELD]
`v = x.nextProp()`	$\dfrac{o \in pt(x) \qquad o.\texttt{s} \text{ exists}}{\{\texttt{s}\} \subseteq pt(v)}$	[PROPITER]

Worst-Case Complexity

For Java:

x.f=y

$$\frac{o \in pt(x)}{pt(y) \subseteq pt(o.\mathtt{f})}$$

O(N^3)

For JavaScript:

x[v]=y

$$\frac{o \in pt(x) \qquad \mathbf{s} \in pt(v)}{pt(y) \subseteq pt(o.\mathbf{s})}$$

O(N^4)

Problem

```
1 src = {}
2 dest = {}
3 src["ext"] = {}
4 src["ins"] = {}
5 prop = (*) ? "ext" : "ins";
6 t = src[prop];
7 dest[prop] = t;
```

Prop re-defined between accesses

$$\dfrac{\dfrac{\text{L4}\,\dfrac{o_1 \in pt(src)}{o_4 \in pt(o_1.\mathtt{ins})} \qquad \dfrac{o_1 \in pt(src) \qquad \mathtt{ins} \in pt(prop)}{pt(o_1.\mathtt{ins}) \subseteq pt(t)}\,\text{L6}}{o_4 \in pt(t)} \qquad \dfrac{\mathtt{ext} \in pt(prop) \qquad o_2 \in pt(dest)}{pt(t) \subseteq pt(o_2.\mathtt{ext})}\,\text{L7}}{o_4 \in pt(o_2.\mathtt{ext})}$$

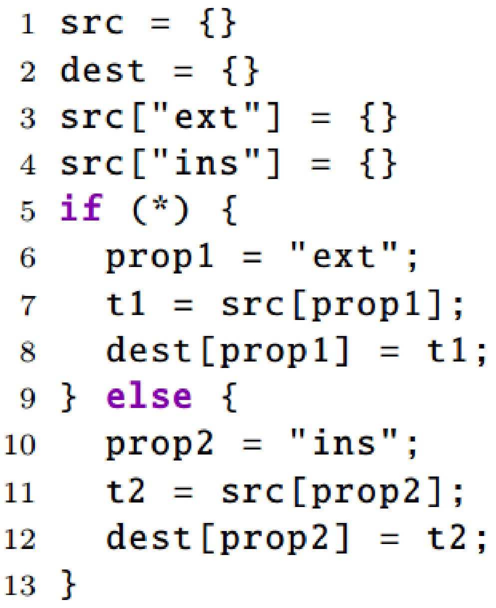
Correlation Tracking Technique

- A technique that helps address issues caused by dynamic property accesses by making the points-to analysis *more precise*.

- The key idea is to enhance Andersen's analysis to track correlations between dynamic property reads and writes that use the same property name.

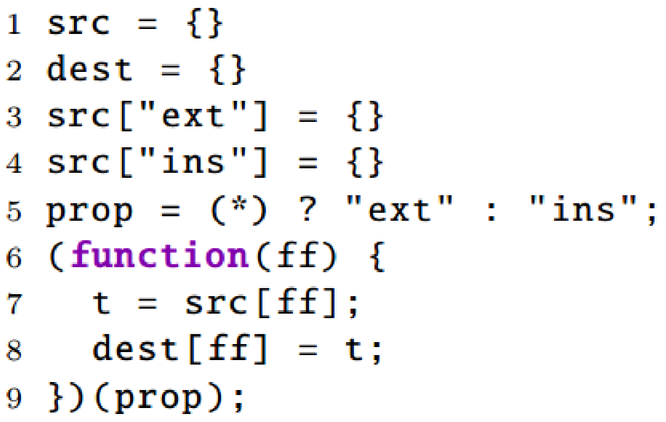
Correlation Tracking Technique

- Example:



```
src = {}
dest = {}
src["ext"] = {}
src["ins"] = {}
if (*) {
  prop1 = "ext";
  t1 = src[prop1];
  dest[prop1] = t1;
} else {
  prop2 = "ins";
  t2 = src[prop2];
  dest[prop2] = t2;
}
```

(a)



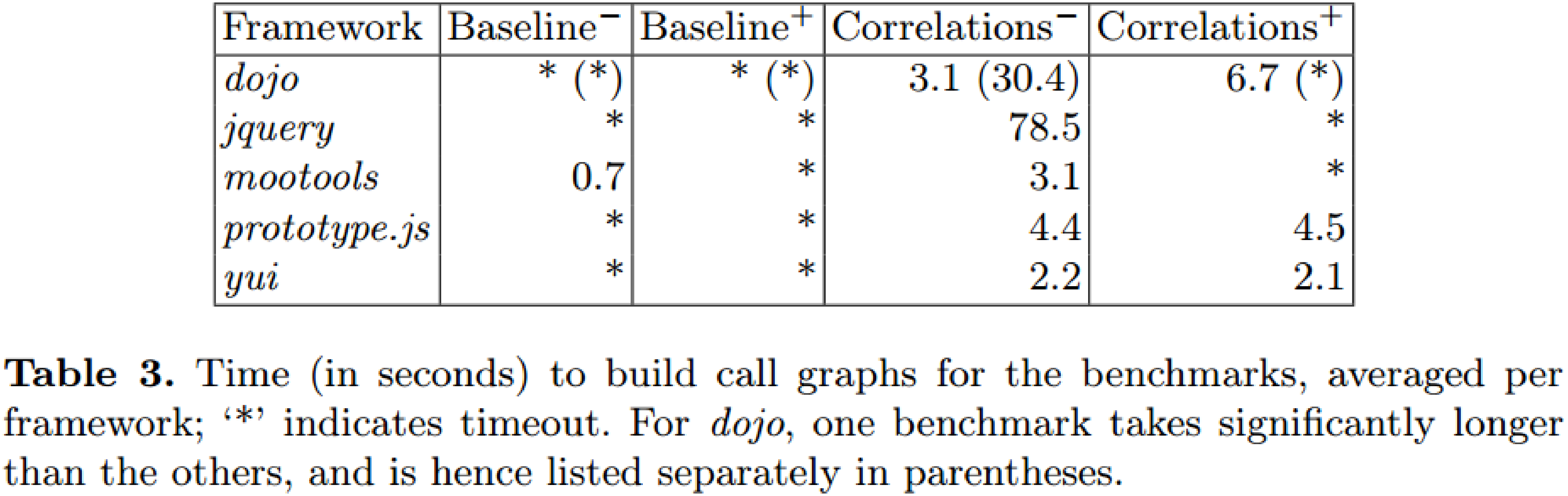
```
src = {}
dest = {}
src["ext"] = {}
src["ins"] = {}
prop = (*) ? "ext" : "ins";
(function(ff) {
  t = src[ff];
  dest[ff] = t;
})(prop);
```

(b)

Evaluation

- Five popular JavaScript frameworks and six benchmarks
- For each benchmark, the authors compared their techniques with built-in WALA standard points-to analysis

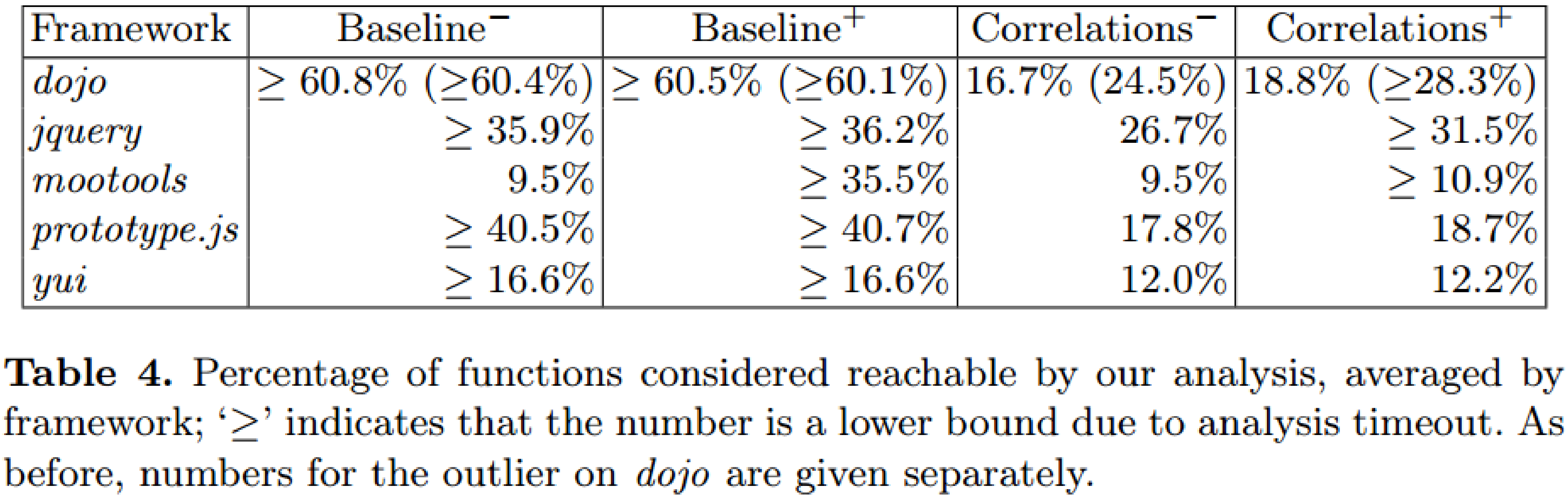
Evaluation



Framework	Baseline$^-$	Baseline$^+$	Correlations$^-$	Correlations$^+$
dojo	* (*)	* (*)	3.1 (30.4)	6.7 (*)
jquery	*	*	78.5	*
mootools	0.7	*	3.1	*
prototype.js	*	*	4.4	4.5
yui	*	*	2.2	2.1

Table 3. Time (in seconds) to build call graphs for the benchmarks, averaged per framework; '*' indicates timeout. For *dojo*, one benchmark takes significantly longer than the others, and is hence listed separately in parentheses.

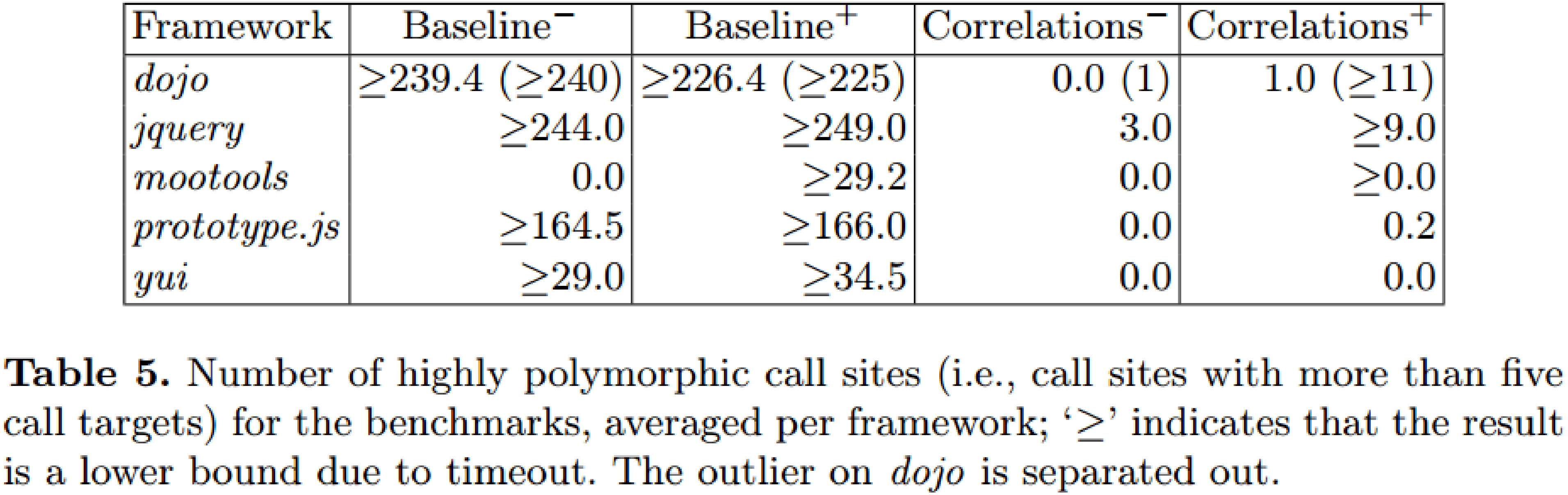
Evaluation



Framework	Baseline$^-$	Baseline$^+$	Correlations$^-$	Correlations$^+$
dojo	≥ 60.8% (≥60.4%)	≥ 60.5% (≥60.1%)	16.7% (24.5%)	18.8% (≥28.3%)
jquery	≥ 35.9%	≥ 36.2%	26.7%	≥ 31.5%
mootools	9.5%	≥ 35.5%	9.5%	≥ 10.9%
prototype.js	≥ 40.5%	≥ 40.7%	17.8%	18.7%
yui	≥ 16.6%	≥ 16.6%	12.0%	12.2%

Table 4. Percentage of functions considered reachable by our analysis, averaged by framework; '≥' indicates that the number is a lower bound due to analysis timeout. As before, numbers for the outlier on *dojo* are given separately.

Evaluation



Framework	Baseline$^-$	Baseline$^+$	Correlations$^-$	Correlations$^+$
dojo	≥239.4 (≥240)	≥226.4 (≥225)	0.0 (1)	1.0 (≥11)
jquery	≥244.0	≥249.0	3.0	≥9.0
mootools	0.0	≥29.2	0.0	≥0.0
prototype.js	≥164.5	≥166.0	0.0	0.2
yui	≥29.0	≥34.5	0.0	0.0

Table 5. Number of highly polymorphic call sites (i.e., call sites with more than five call targets) for the benchmarks, averaged per framework; '≥' indicates that the result is a lower bound due to timeout. The outlier on *dojo* is separated out.

Conclusion & Future Work

- These results clearly show that correlation tracking significantly improves scalability and precision of field-sensitive points-to analysis for a range of JavaScript frameworks.
- Finding and solving the further causes of complexity in the frameworks

Q&A