Introspective Analysis: Context-sensitivity, Across the Board

Authors: **Yannis Smaragdakis, George Kastrinis, George Balatsouras**

Venue:PLDI 2014

Present by
Peeratham (Karn) Techapalokul
09/29/2015

Problem

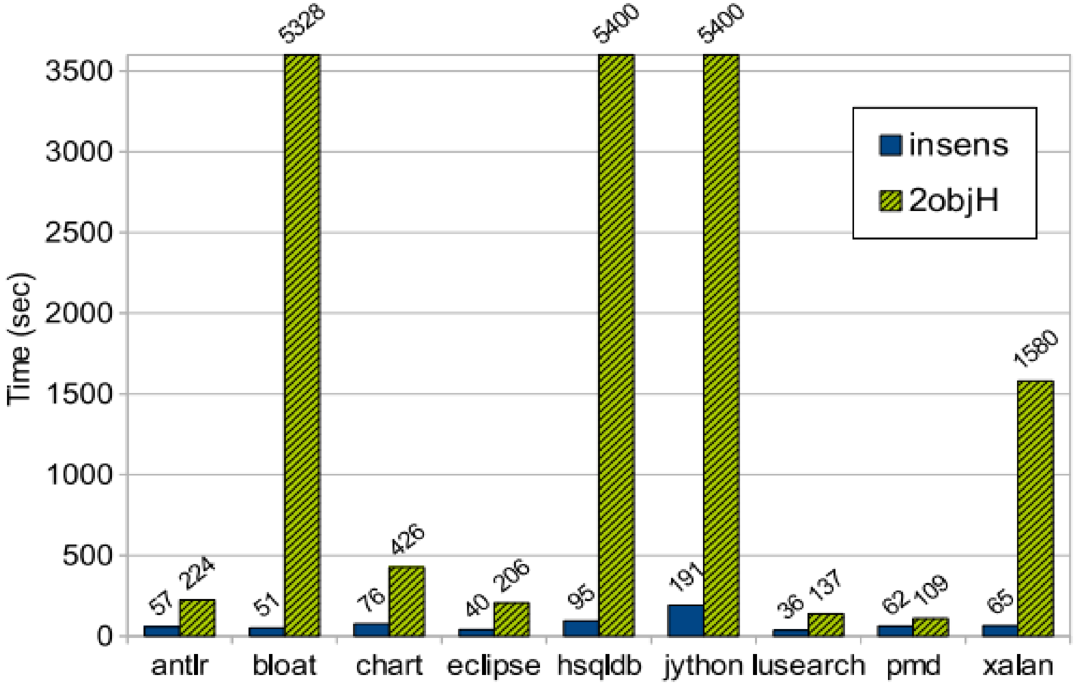


Figure 1. Comparison of running times of context-insensitive analysis vs. 2-object-sensitive with context-sensitive heap. The y-axis is truncated to 1hr for readability.

How to make precise context-sensitive analyses scale like context-insensitive analysis?

Observations:

- performance of a deep-context analysis is **bimodal**
- for some methods and objects, more contexts doesn’t help but incurs performance overhead

Solution: Introspective analysis

- Run cheap analysis first to see where to analyze context-sensitively
  - use metrics and heuristics to guide our decision
- Selectively refine the analysis

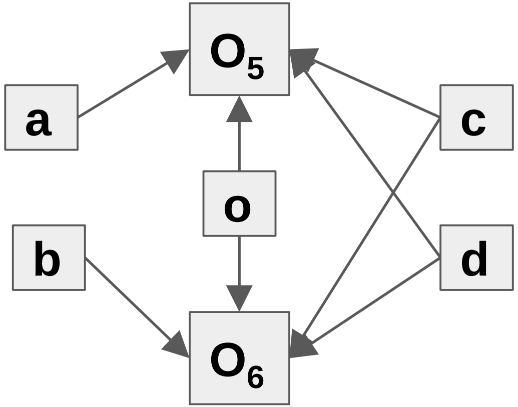
“ a knob for adjusting the precision/scalability tradeoff ”

Outline

- Background: variants of context sensitive analysis
- Metrics and Heuristics : where not to refine
- High-level structure of analysis model
- Datalog rules for points-to analysis
- Evaluation

Background: Context-insensitive analysis

```
Object id(Object o) {
    return o;
}
void f() {
    Object a = new Object();
    Object b = new Object();
    Object c = id(a);
    Object d = id(b);
}
```



Background: Context-sensitive analysis

- **context-insensitivity**
- **Call-site sensitivity**
- **Object-sensitivity**
- **Type sensitivity**

More on type sensitive analysis in the original paper:
Smaragdakis, Yannis, Martin Bravenboer, and Ondrej Lhoták. "Pick your contexts well: understanding object-sensitivity." *ACM SIGPLAN Notices* 46.1 (2011): 17-30.

2objH: 2-object-sensitive analysis with 1 context-sensitive heap

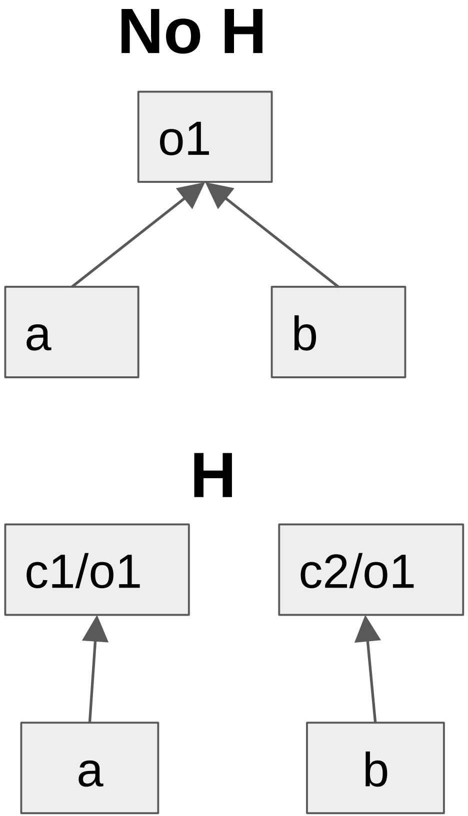
2obj : up to 2 context-depth of the allocation sites of the receiver objects

H: context-sensitive heap

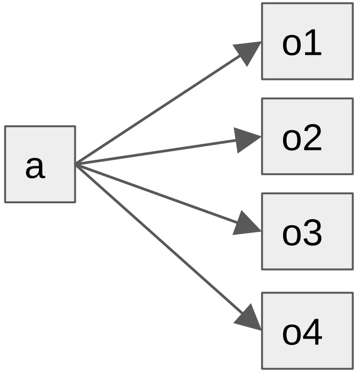
- just by the allocation site (0H)
- a combination of the allocation site and the calling context in which the method containing it is called. (1H)
- add to other kind of context-sensitivity (objH,callH,typeH)

Context-sensitive heap object: H

```
    Object alloc() {
o1:   return new Object();
    }
    void f() {
c1:   Object a = alloc();
c2:   Object b = alloc();
    }
```

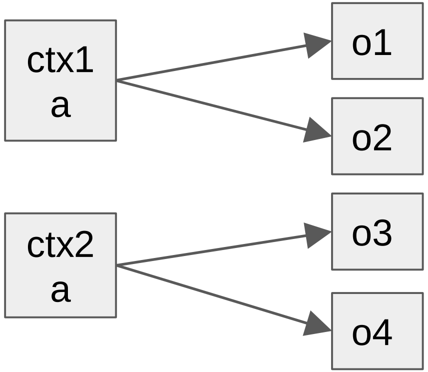


Why does scalability barrier arise?

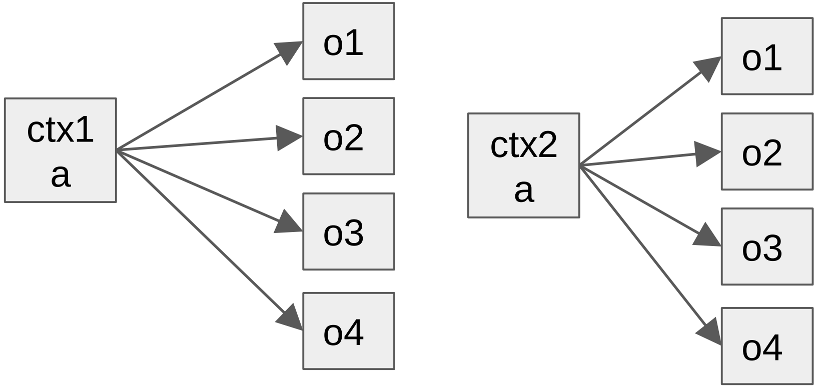


Scenario: a method argument points to n-objects
c contexts

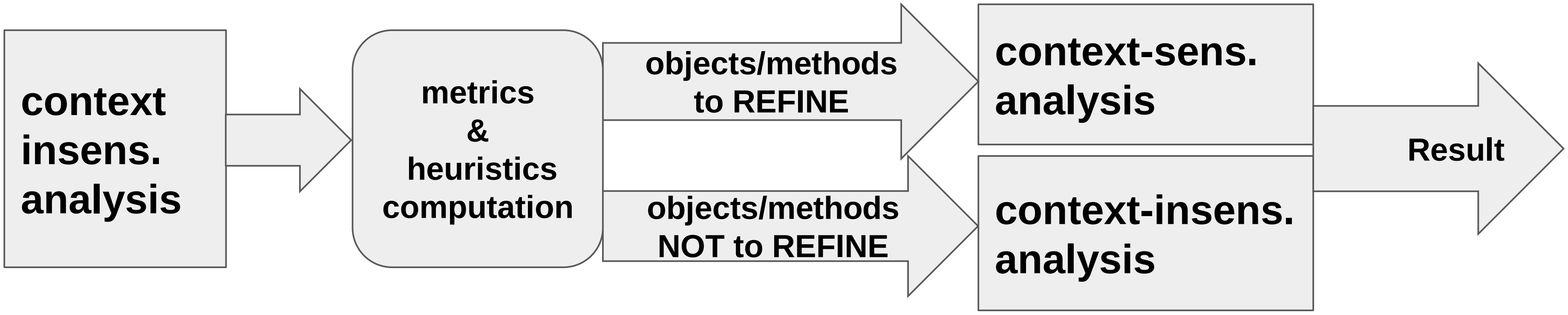
Best-case:
n/c points-to facts per context



Worst-case:
n*c points-to facts!



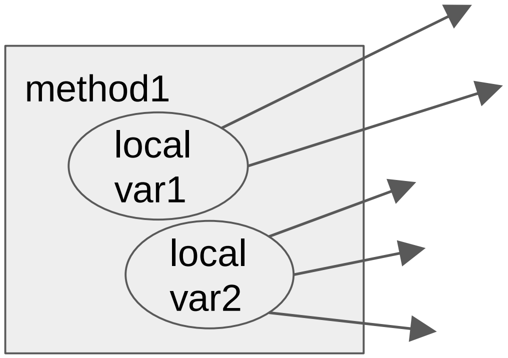
High-level structure of analysis model



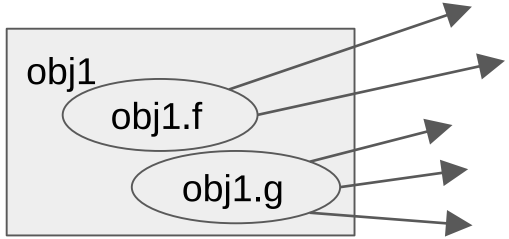
What metrics & heuristics to use?

Decide what to refine

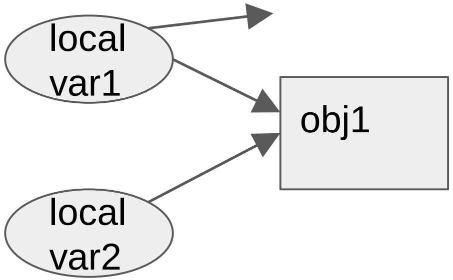
Metric#2 method's total points-to volume
Compute for every method the cumulative size of points-to sets over all local variables.



Metric#3 total field points-to
For each object (i.e., allocation site)
compute the maximum field points-to set over all of its fields



Metric#5 object's pointed-by vars
For each object (i.e., allocation site)
compute the number of local variables pointing to it.



Example: Heuristics B

What not to refine:

method : calls sites that get invoked with a ***total points-to volume*** (metric #2) > 10000

object : object allocations that have ***total field points-to*** (metric#3) x ***pointed-by-vars*** (metrics#5) >10000.

Datalog: Input relations

program Instruction

ALLOC (*var* : *V*, *heap* : *H*, *inMeth* : *M*) # *var* = *new* ...
MOVE (*to* : *V*, *from* : *V*) # *to* = *from*
LOAD (*to* : *V*, *base* : *V*, *fld* : *F*) # *to* = *base.fld*
STORE (*base* : *V*, *fld* : *F*, *from* : *V*) # *base.fld* = *from*
VCALL (*base* : *V*, *sig* : *S*, *invo* : *I*, *inMeth* : *M*) # *base.sig*(..)

type system & other environment info

FORMALARG (*meth* : *M*, *i* : $\mathbb{N}$, *arg* : *V*)
ACTUALARG (*invo* : *I*, *i* : $\mathbb{N}$, *arg* : *V*)
FORMALRETURN (*meth* : *M*, *ret* : *V*)
ACTUALRETURN (*invo* : *I*, *var* : *V*)
THISVAR (*meth* : *M*, *this* : *V*)
HEAPTYPE (*heap* : *H*, *type* : *T*)
LOOKUP (*type* : *T*, *sig* : *S*, *meth* : *M*)

where to refine:

SITETOREFINE (*invo* : *I*, *meth* : *M*)
OBJECTTOREFINE (*heap* : *H*)

Output relations:

VARPOINTSTO (*var* : *V*, *ctx* : *C*, *heap* : *H*, *hctx* : *HC*)
CALLGRAPH (*invo* : *I*, *callerCtx* : *C*, *meth* : *M*, *calleeCtx* : *C*)
FLDPOINTSTO (*baseH*: *H*, *baseHCtx*: *HC*, *fld*: *F*, *heap*: *H*, *hctx*: *HC*)
INTERPROCASSIGN (*to* : *V*, *toCtx* : *C*, *from* : *V*, *fromCtx* : *C*)
REACHABLE (*meth* : *M*, *ctx* : *C*)

VARPOINTSTO (var : V , ctx : C , heap : H , hctx : HC)

- **VARPOINTSTO** relation links a variable (var) to a heap object (heap)
- method calls are qualified by calling contexts ctx
- heap object, *heap* is qualified with a heap context, *hctx*

Constructors of contexts:

$\textsc{Record}\ (heap : H, ctx : C) = \boldsymbol{newHCtx : HC}$

$\textsc{Merge}\ (heap : H, hctx : HC, invo : I, ctx : C) = \boldsymbol{newCtx : C}$

$\textsc{RecordRefined}\ (heap : H, ctx : C) = \boldsymbol{newHCtx : HC}$

$\textsc{MergeRefined}\ (heap : H, hctx : HC, invo : I, ctx : C) = \boldsymbol{newCtx : C}$

1CallH
RECORD(heap : H, ctx: C) = ctx
MERGE(heap: H, hctx: HC, invo :I, ctx :C) = invo

Example of Datalog analysis rules

Inferred facts ← previously established facts

$$
\begin{aligned}
&\textbf{R}\textsc{\textbf{ecord}}\,(heap, ctx) = \boldsymbol{hctx},\\
&\text{V}\textsc{ar}\text{P}\textsc{oints}\text{T}\textsc{o}\,(var, ctx, heap, hctx) \longleftarrow\\
&\quad \text{R}\textsc{eachable}\,(meth, ctx),\ \text{A}\textsc{lloc}\,(var, heap, meth),\\
&\quad !\text{O}\textsc{bject}\text{T}\textsc{o}\text{R}\textsc{efine}\,(heap).
\end{aligned}
$$

Evaluation:

- **Our focus: introspective object-sensitivity**

    -introspective analysis : 2objH-introA, 2objH-introB

    -baselines: 2objH and context insensitive

Evaluation metrics

Performance

- running time

Precision

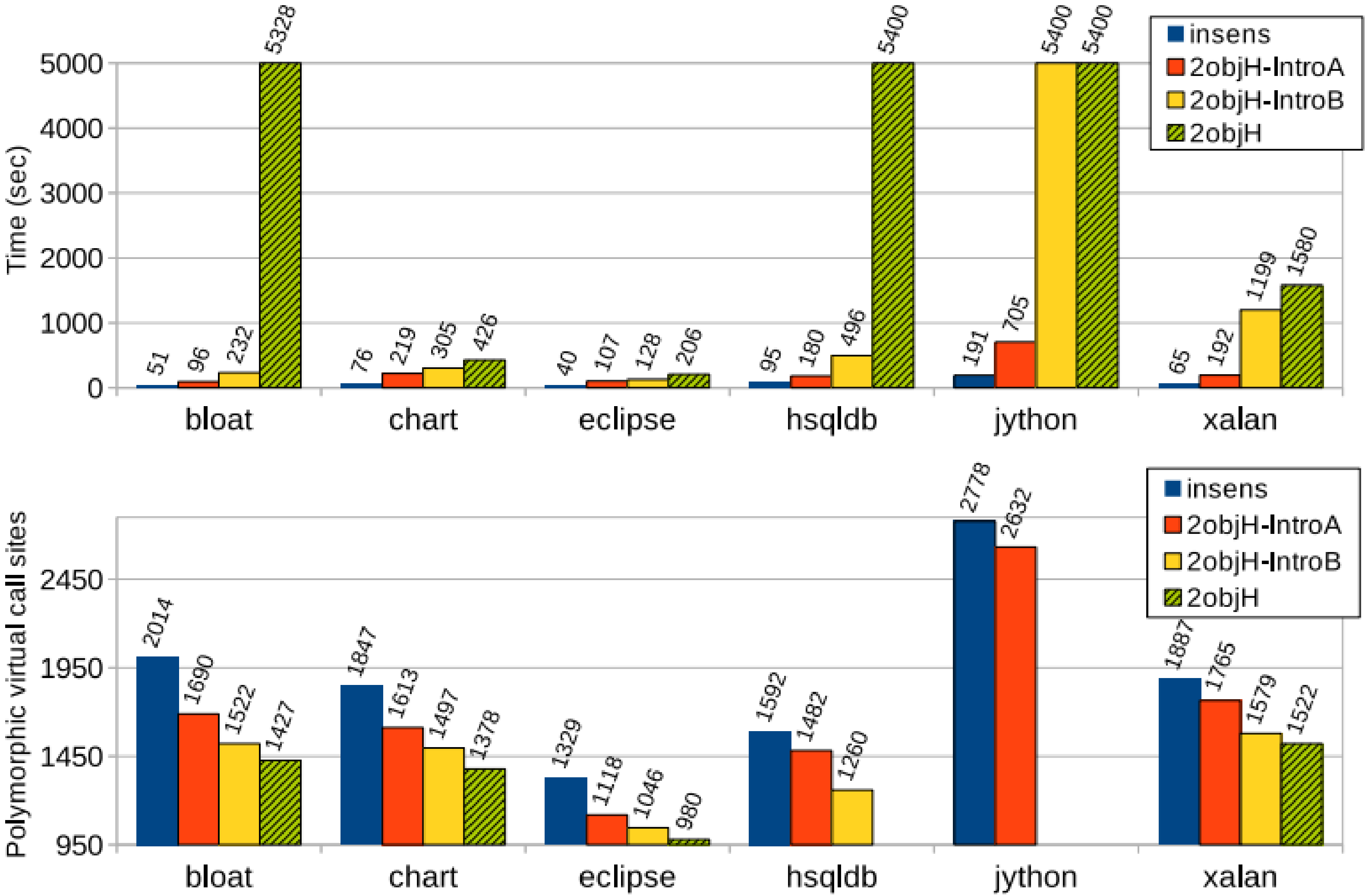
- Polymorphic virtual call sites (calls that cannot be devirtualized)
- reachable methods
- casts that cannot be eliminated

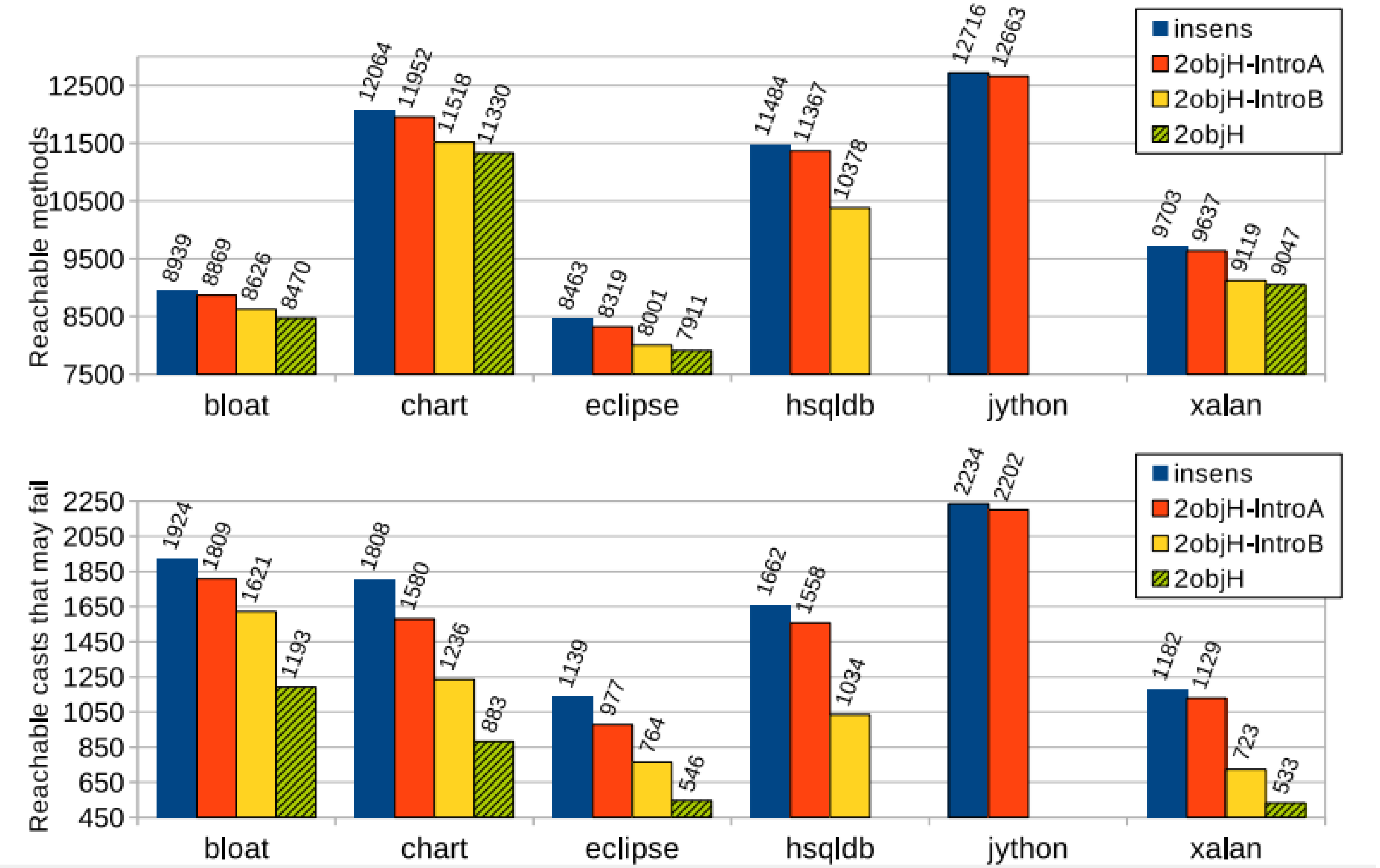
For all the metrics, the lower the better!

Result: Majority of call sites and objects are refined

	Call Sites		Objects	
	Heur. A	Heur. B	Heur. A	Heur. B
bloat	28.0 %	0.7 %	12.3 %	7.1 %
chart	13.7 %	3.4 %	12.0 %	5.0 %
eclipse	14.5 %	0.0 %	11.0 %	5.0 %
hsqldb	30.0 %	1.1 %	16.8 %	14.0 %
jython	36.0 %	3.0 %	25.0 %	18.8 %
pmd	12.5 %	0.0 %	10.5 %	5.3 %
xalan	18.0 %	0.0 %	13.0 %	7.8 %
average	21.81 %	1.18 %	14.37 %	9.0 %

Figure 4. Number of call sites and objects selected to **not** be refined by each introspective variant. All results are rounded to the first decimal digit.


Time (sec)
0
1000
2000
3000
4000
5000
51
96
232
5328
76
219
305
426
40
107
128
206
95
180
496
5400
191
705
5400
5400
65
192
1199
1580
bloat
chart
eclipse
hsqldb
jython
xalan
insens
2objH-IntroA
2objH-IntroB
2objH
Polymorphic virtual call sites
950
1450
1950
2450
2014
1690
1522
1427
1847
1613
1497
1378
1329
1118
1046
980
1592
1482
1260
2778
2632
1887
1765
1579
1522
bloat
chart
eclipse
hsqldb
jython
xalan
insens
2objH-IntroA
2objH-IntroB
2objH


insens
2objH-IntroA
2objH-IntroB
2objH
Reachable methods
12500
11500
10500
9500
8500
7500
8939
8869
8626
8470
12064
11952
11518
11330
8463
8319
8001
7911
11484
11367
10378
12716
12663
9703
9637
9119
9047
bloat
chart
eclipse
hsqldb
jython
xalan
insens
2objH-IntroA
2objH-IntroB
2objH
Reachable casts that may fail
2250
2050
1850
1650
1450
1250
1050
850
650
450
1924
1809
1621
1193
1808
1580
1236
883
1139
977
764
546
1662
1558
1034
2234
2202
1182
1129
723
533
bloat
chart
eclipse
hsqldb
jython
xalan

Results:

- Scalability
    - 2objH-Intro scales much better than 2objH
- Precision
    - significant precision gains over a context-insens
    - 2obj-introB precision comparable to 2objH

Conclusion

Tuning (scalability, performance) for context-sensitive analysis is possible and practical:

- **precision loss is small**
- **the scalability gain is substantial**

Thank you!

Questions?

References

Smaragdakis, Yannis, George Kastrinis, and George Balatsouras. "Introspective analysis: context-sensitivity, across the board." *ACM SIGPLAN Notices*. Vol. 49. No. 6. ACM, 2014.

Smaragdakis, Yannis, Martin Bravenboer, and Ondrej Lhoták. "Pick your contexts well: understanding object-sensitivity." *ACM SIGPLAN Notices* 46.1 (2011): 17-30.

Lhoták, Ondřej, and Laurie Hendren. "Evaluating the benefits of context-sensitive points-to analysis using a BDD-based implementation." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 18.1 (2008): 3.

Kastrinis, George, and Yannis Smaragdakis. "Hybrid context-sensitivity for points-to analysis." *ACM SIGPLAN Notices* 48.6 (2013): 423-434.