# Pick Your Contexts Well: Understanding Object-Sensitivity

## The Making of a Precise and Scalable Pointer Analysis

**Yannis Smaragdakis** - University of Massachusetts, Amherst and University of Athens

**Martin Bravenboer** - LogicBlox Inc.

**Ondřej Lhoták** - University of Waterloo

# Context Sensitivity

Call Site sensitivity vs Object Sensitivity

# Object Sensitivity

```
class A {
    void foo(Object o) {...}
}
class Client{
    void bar(A a1,A a2){
        ...
        a1.foo(someobj1);
        ...
        a2.foo(someobj2);
    }
}
```

# General Framework for context-sensitive analyses

precision vs performance

# General Framework for context-sensitive analyses

precision vs performance

Context - HContext

# General Framework for context-sensitive analyses

precision vs performance

Context - HContext

Framework functions:

- Record (object is created)
- Merge (method invocation)

# Full Object Sensitivity

- Original object sensitivity
- Allocation site of receiver object + allocation site of the receiver’s object allocator object + ...
- Framework
  - Context = HContext = $Lab^n$
  - Record(l,c) = cons(l, first(n - 1, c))
  - Merge(l, hc, c) = hc

# Plain Object Sensitivity

- Allocation site of receiver object + allocation site of caller object + allocation site of caller' s caller object + ...
- Framework
  - Context = HContext = $Lab^n$
  - Record(l,c) = cons(l, first(n - 1, c))
  - Merge(l, hc, c) = cons(car(hc), first(n - 1, c))

# Type Sensitivity

- Why not reduce the combinatorial explosion

by reducing combinations

- Instead of allocation sites, keep types

- Just some elements of context are transformed to types by a function

T: Instr -> ClassName

# Choice of Type Contexts

```
class C
{
        …
        void m()
        {
                ...
                new A();
                ...
        }
        ...
}
```

```
class C{
        …
        new A();
        ...
}
```

# Choice of Type Contexts

- 2Type+1H
  - Dynamic Type A of the allocated object
  - Upper bound C on the dynamic type of the allocator object

```
class C{
    …
    new A();
    ...
}
```

# Choice of Type Contexts

- 2Type+1H
  - Dynamic Type A of the allocated object
  - Upper bound C on the dynamic type of the allocator object
- 1Type1Obj+1H
  - Dynamic Type of the receiver object
  - Upper bound on the dynamic type of the receiver object’s allocator object.
  - Dynamic Type of the receiver object’s allocator object
  - Upper bound on the dynamic type of the receiver object’s allocator’s allocator object.

# Implementation

- DOOP framework

- Datalog language

- Explicit representation of relations
  - opposed to BDDs.
  - explicit is faster, but may introduce redundancy.

# Evaluation

1. Full-object-sensitivity vs Plain-object-sensitivity
    a. Is full-object-sensitivity advantageous compared to plain-object-sensitivity in terms of precision and performance?

# Evaluation

1. Full-object-sensitivity vs Plain-object-sensitivity
    a. Is full-object-sensitivity advantageous compared to plain-object-sensitivity in terms of precision and performance?
2. Importance of Type Context Choice
    a. Does the definition of the functions matter?

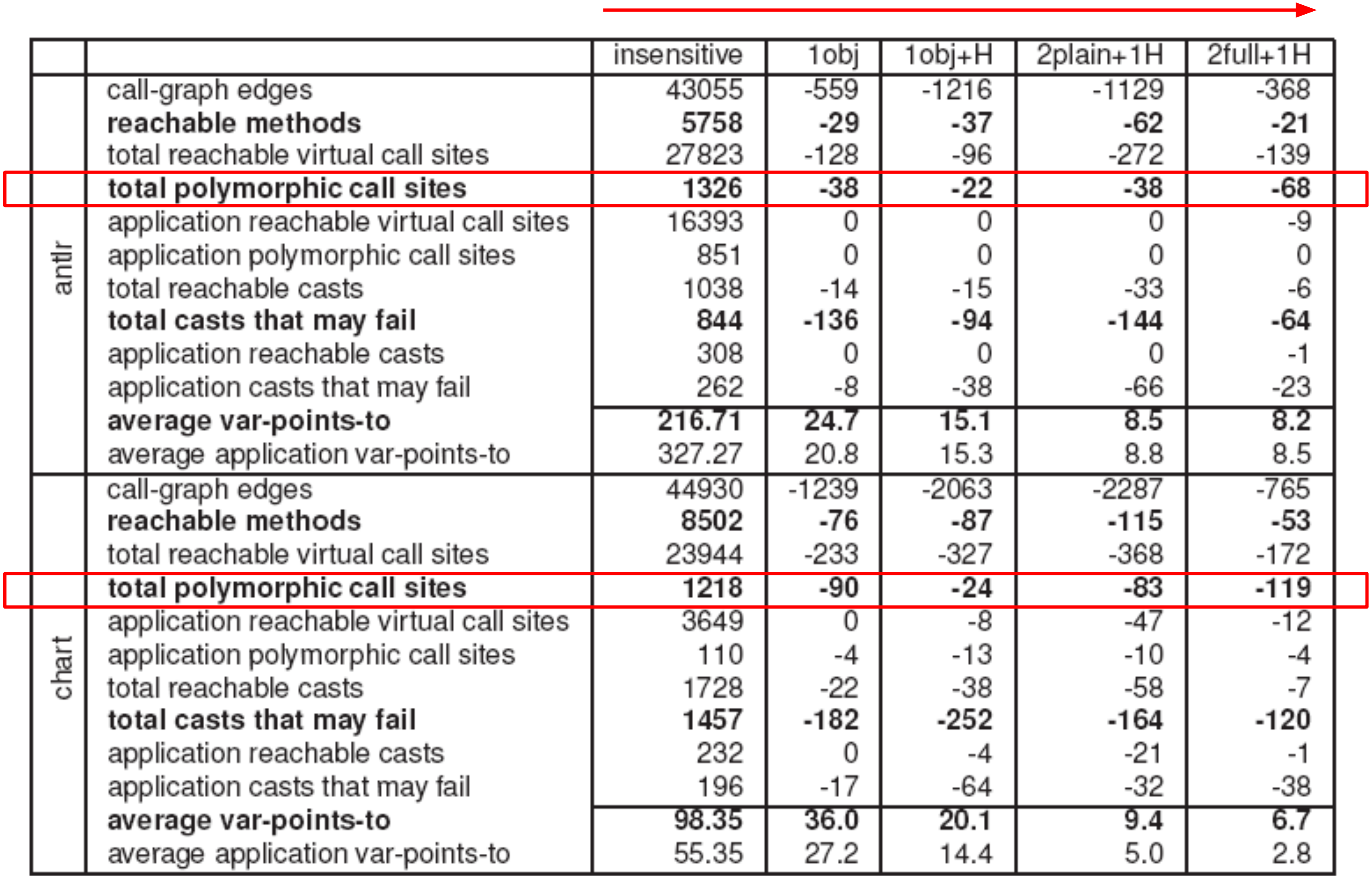
# Evaluation

1. Full-object-sensitivity vs Plain-object-sensitivity
   a. Is full-object-sensitivity advantageous compared to plain-object-sensitivity in terms of precision and performance?
2. Importance of Type Context Choice
   a. Does the definition of the T function matter?
3. Type-sensitive precision and performance
   a. Does type-sensitive achieve higher scalability than regular object-sensitive analyses while maintaining most of the precision?

# Evaluation

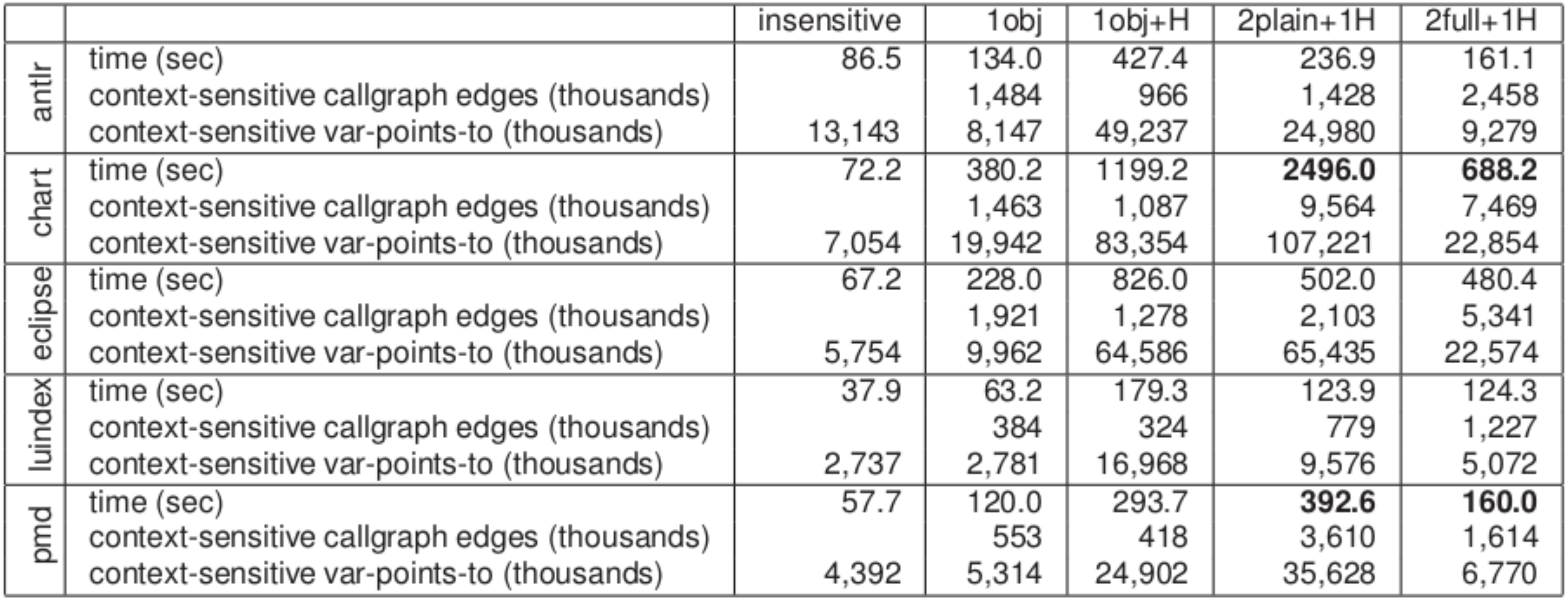
- 64 bit machine
- quad core
- 24GB RAM
- DaCapo benchmark programs.

# Evaluation



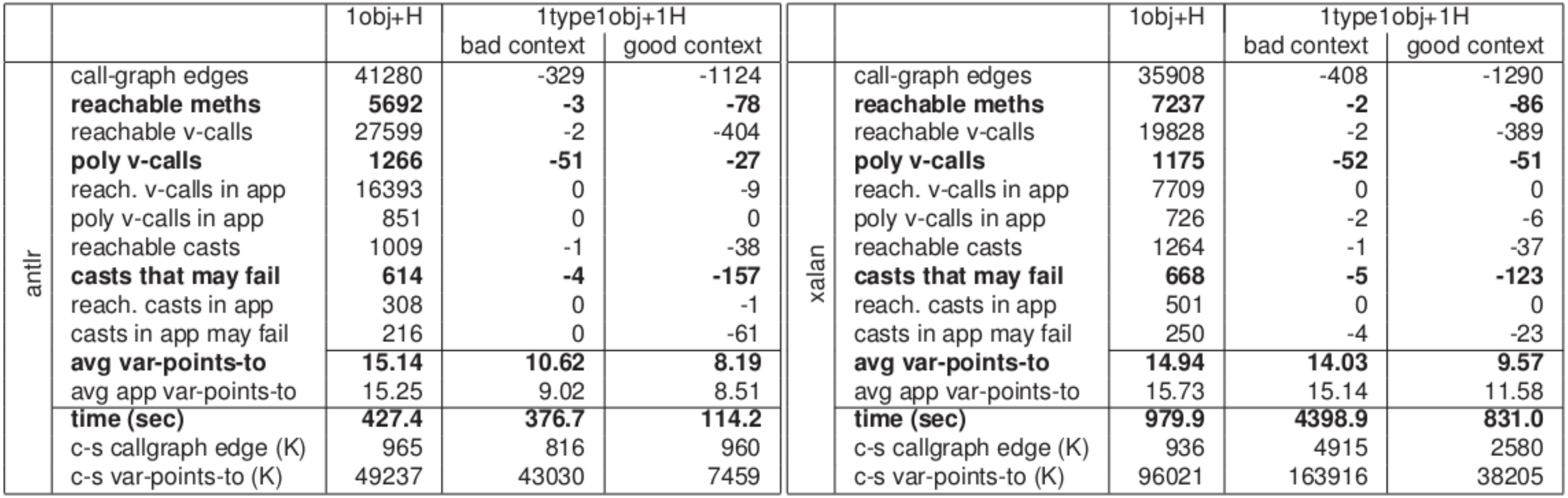
| | | insensitive | 1obj | 1obj+H | 2plain+1H | 2full+1H |
|---|---|---|---|---|---|---|
| antlr | call-graph edges | 43055 | -559 | -1216 | -1129 | -368 |
| | **reachable methods** | **5758** | **-29** | **-37** | **-62** | **-21** |
| | total reachable virtual call sites | 27823 | -128 | -96 | -272 | -139 |
| | **total polymorphic call sites** | **1326** | **-38** | **-22** | **-38** | **-68** |
| | application reachable virtual call sites | 16393 | 0 | 0 | 0 | -9 |
| | application polymorphic call sites | 851 | 0 | 0 | 0 | 0 |
| | total reachable casts | 1038 | -14 | -15 | -33 | -6 |
| | **total casts that may fail** | **844** | **-136** | **-94** | **-144** | **-64** |
| | application reachable casts | 308 | 0 | 0 | 0 | -1 |
| | application casts that may fail | 262 | -8 | -38 | -66 | -23 |
| | **average var-points-to** | **216.71** | **24.7** | **15.1** | **8.5** | **8.2** |
| | average application var-points-to | 327.27 | 20.8 | 15.3 | 8.8 | 8.5 |
| chart | call-graph edges | 44930 | -1239 | -2063 | -2287 | -765 |
| | **reachable methods** | **8502** | **-76** | **-87** | **-115** | **-53** |
| | total reachable virtual call sites | 23944 | -233 | -327 | -368 | -172 |
| | **total polymorphic call sites** | **1218** | **-90** | **-24** | **-83** | **-119** |
| | application reachable virtual call sites | 3649 | 0 | -8 | -47 | -12 |
| | application polymorphic call sites | 110 | -4 | -13 | -10 | -4 |
| | total reachable casts | 1728 | -22 | -38 | -58 | -7 |
| | **total casts that may fail** | **1457** | **-182** | **-252** | **-164** | **-120** |
| | application reachable casts | 232 | 0 | -4 | -21 | -1 |
| | application casts that may fail | 196 | -17 | -64 | -32 | -38 |
| | **average var-points-to** | **98.35** | **36.0** | **20.1** | **9.4** | **6.7** |
| | average application var-points-to | 55.35 | 27.2 | 14.4 | 5.0 | 2.8 |

# Evaluation



| | | insensitive | 1obj | 1obj+H | 2plain+1H | 2full+1H |
|---|---|---|---|---|---|---|
| antlr | time (sec) | 86.5 | 134.0 | 427.4 | 236.9 | 161.1 |
| | context-sensitive callgraph edges (thousands) | | 1,484 | 966 | 1,428 | 2,458 |
| | context-sensitive var-points-to (thousands) | 13,143 | 8,147 | 49,237 | 24,980 | 9,279 |
| chart | time (sec) | 72.2 | 380.2 | 1199.2 | **2496.0** | **688.2** |
| | context-sensitive callgraph edges (thousands) | | 1,463 | 1,087 | 9,564 | 7,469 |
| | context-sensitive var-points-to (thousands) | 7,054 | 19,942 | 83,354 | 107,221 | 22,854 |
| eclipse | time (sec) | 67.2 | 228.0 | 826.0 | 502.0 | 480.4 |
| | context-sensitive callgraph edges (thousands) | | 1,921 | 1,278 | 2,103 | 5,341 |
| | context-sensitive var-points-to (thousands) | 5,754 | 9,962 | 64,586 | 65,435 | 22,574 |
| luindex | time (sec) | 37.9 | 63.2 | 179.3 | 123.9 | 124.3 |
| | context-sensitive callgraph edges (thousands) | | 384 | 324 | 779 | 1,227 |
| | context-sensitive var-points-to (thousands) | 2,737 | 2,781 | 16,968 | 9,576 | 5,072 |
| pmd | time (sec) | 57.7 | 120.0 | 293.7 | **392.6** | **160.0** |
| | context-sensitive callgraph edges (thousands) | | 553 | 418 | 3,610 | 1,614 |
| | context-sensitive var-points-to (thousands) | 4,392 | 5,314 | 24,902 | 35,628 | 6,770 |

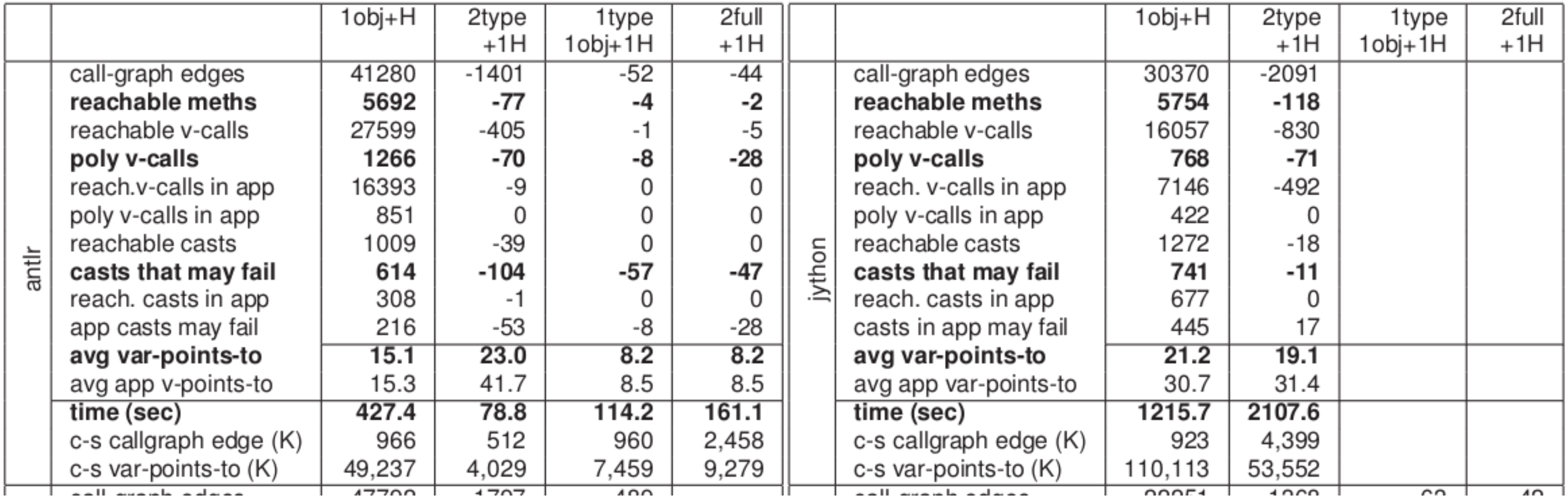
# Evaluation



| | | 1obj+H | 1type1obj+1H | |
|---|---|---|---|---|
| | | | bad context | good context |
| antlr | call-graph edges | 41280 | -329 | -1124 |
| | **reachable meths** | **5692** | **-3** | **-78** |
| | reachable v-calls | 27599 | -2 | -404 |
| | **poly v-calls** | **1266** | **-51** | **-27** |
| | reach. v-calls in app | 16393 | 0 | -9 |
| | poly v-calls in app | 851 | 0 | 0 |
| | reachable casts | 1009 | -1 | -38 |
| | **casts that may fail** | **614** | **-4** | **-157** |
| | reach. casts in app | 308 | 0 | -1 |
| | casts in app may fail | 216 | 0 | -61 |
| | **avg var-points-to** | **15.14** | **10.62** | **8.19** |
| | avg app var-points-to | 15.25 | 9.02 | 8.51 |
| | **time (sec)** | **427.4** | **376.7** | **114.2** |
| | c-s callgraph edge (K) | 965 | 816 | 960 |
| | c-s var-points-to (K) | 49237 | 43030 | 7459 |

| | | 1obj+H | 1type1obj+1H | |
|---|---|---|---|---|
| | | | bad context | good context |
| xalan | call-graph edges | 35908 | -408 | -1290 |
| | **reachable meths** | **7237** | **-2** | **-86** |
| | reachable v-calls | 19828 | -2 | -389 |
| | **poly v-calls** | **1175** | **-52** | **-51** |
| | reach. v-calls in app | 7709 | 0 | 0 |
| | poly v-calls in app | 726 | -2 | -6 |
| | reachable casts | 1264 | -1 | -37 |
| | **casts that may fail** | **668** | **-5** | **-123** |
| | reach. casts in app | 501 | 0 | 0 |
| | casts in app may fail | 250 | -4 | -23 |
| | **avg var-points-to** | **14.94** | **14.03** | **9.57** |
| | avg app var-points-to | 15.73 | 15.14 | 11.58 |
| | **time (sec)** | **979.9** | **4398.9** | **831.0** |
| | c-s callgraph edge (K) | 936 | 4915 | 2580 |
| | c-s var-points-to (K) | 96021 | 163916 | 38205 |

# Evaluation



| | | 1obj+H | 2type<br>+1H | 1type<br>1obj+1H | 2full<br>+1H |
|---|---|---|---|---|---|
| antlr | call-graph edges | 41280 | -1401 | -52 | -44 |
| | **reachable meths** | **5692** | **-77** | **-4** | **-2** |
| | reachable v-calls | 27599 | -405 | -1 | -5 |
| | **poly v-calls** | **1266** | **-70** | **-8** | **-28** |
| | reach.v-calls in app | 16393 | -9 | 0 | 0 |
| | poly v-calls in app | 851 | 0 | 0 | 0 |
| | reachable casts | 1009 | -39 | 0 | 0 |
| | **casts that may fail** | **614** | **-104** | **-57** | **-47** |
| | reach. casts in app | 308 | -1 | 0 | 0 |
| | app casts may fail | 216 | -53 | -8 | -28 |
| | **avg var-points-to** | **15.1** | **23.0** | **8.2** | **8.2** |
| | avg app v-points-to | 15.3 | 41.7 | 8.5 | 8.5 |
| | **time (sec)** | **427.4** | **78.8** | **114.2** | **161.1** |
| | c-s callgraph edge (K) | 966 | 512 | 960 | 2,458 |
| | c-s var-points-to (K) | 49,237 | 4,029 | 7,459 | 9,279 |

| | | 1obj+H | 2type<br>+1H | 1type<br>1obj+1H | 2full<br>+1H |
|---|---|---|---|---|---|
| jython | call-graph edges | 30370 | -2091 | | |
| | **reachable meths** | **5754** | **-118** | | |
| | reachable v-calls | 16057 | -830 | | |
| | **poly v-calls** | **768** | **-71** | | |
| | reach. v-calls in app | 7146 | -492 | | |
| | poly v-calls in app | 422 | 0 | | |
| | reachable casts | 1272 | -18 | | |
| | **casts that may fail** | **741** | **-11** | | |
| | reach. casts in app | 677 | 0 | | |
| | casts in app may fail | 445 | 17 | | |
| | **avg var-points-to** | **21.2** | **19.1** | | |
| | avg app var-points-to | 30.7 | 31.4 | | |
| | **time (sec)** | **1215.7** | **2107.6** | | |
| | c-s callgraph edge (K) | 923 | 4,399 | | |
| | c-s var-points-to (K) | 110,113 | 53,552 | | |

# Conclusions

Good choice of context...

- is more precise
  - smaller points-to sets
- yields much faster implementation
  - often 2x or more
- 2Type+1H has the best trade-of between precision and performance