OOPSLA 2017

Data-Driven Context-Sensitivity for Points-to Analysis

KOREA UNIVERSITY

Sehun Jeong*, Minseok Jeon*, Sungdeok Cha, and Hakjoo Oh

PRL Programming Research Laboratory

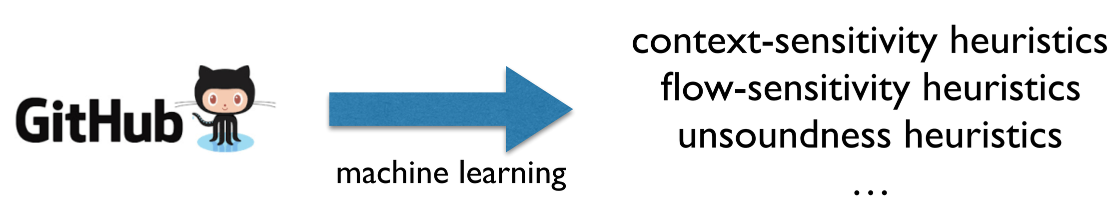
Heuristic Decisions in Static Analysis

Sparrow The Early Bird · WALA · Astrée · DOOP · TAJS · SAFE

- Practical static analyzers involve many heuristics
  - Which procedures should be analyzed context-sensitively?
  - Which relationships between variables should be tracked?
  - Which program parts to analyze unsoundly or soundly?, etc
- Designing a good heuristic is an art
  - Usually done by trials and error: **nontrivial** and **suboptimal**

Automatically Generating Heuristics from Data

- Automate the process: use data to make heuristic decisions in static analysis
- Automatic: little reliance on analysis designers



- Powerful: machine-tuning outperforms hand-tuning
- Stable: can be generated for arbitrary programs

Selective Context-Sensitivity

```
class D{} class E{}
class C{
  void dummy(){}
  Object id1(Object v){return id2(v);}//4
  Object id2(Object v){return v;}}
class B{
  void m(){
    C c = new C();
    D d = (D)c.id1(new D());//query1 //9
    E e = (E)c.id1(new E());//query2 //10
    c.dummy();}}//11
public class A{
  public static void main(String[] args){
    B b = new B();
    b.m();//15
    b.m();}}//16
```

- Context-insensitivity fails to prove the queries
- 2-call-site-sensitivity succeeds but not scale

solution

Apply 2-call-sens:{C.id2}
Apply 1-call-sens:{C.id1}
Apply insens:{B.m, C.dummy}

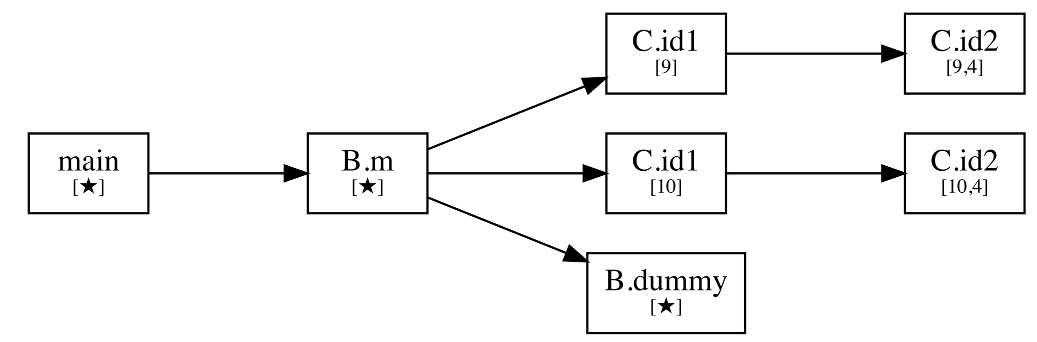


Figure 1: call graph of the solution

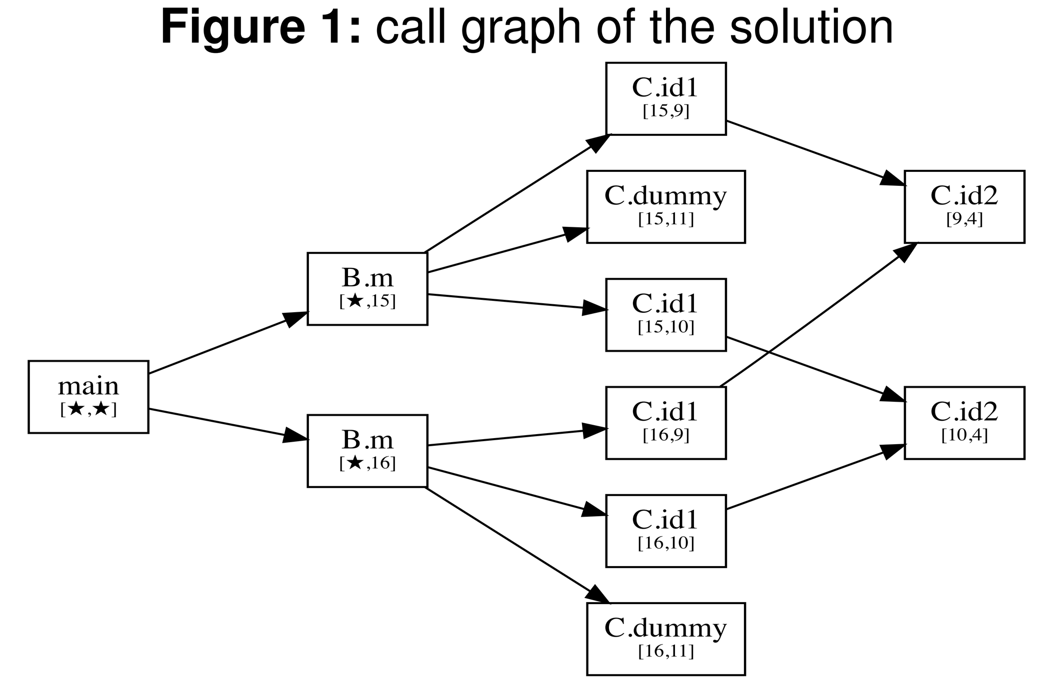
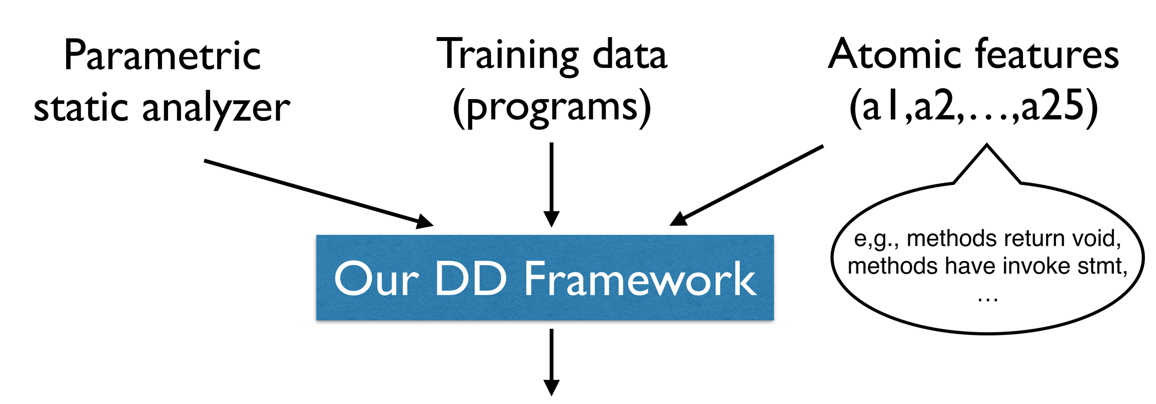


Figure 2: call graph of 2-call-site sensitive

Challenge: How to decide?

⇒Data-Driven approach

Data-Driven Ctx-Sensitivity



Heuristic for applying (hybrid) object-sensitivity:

f2: Methods that require 2-object-sensitivity

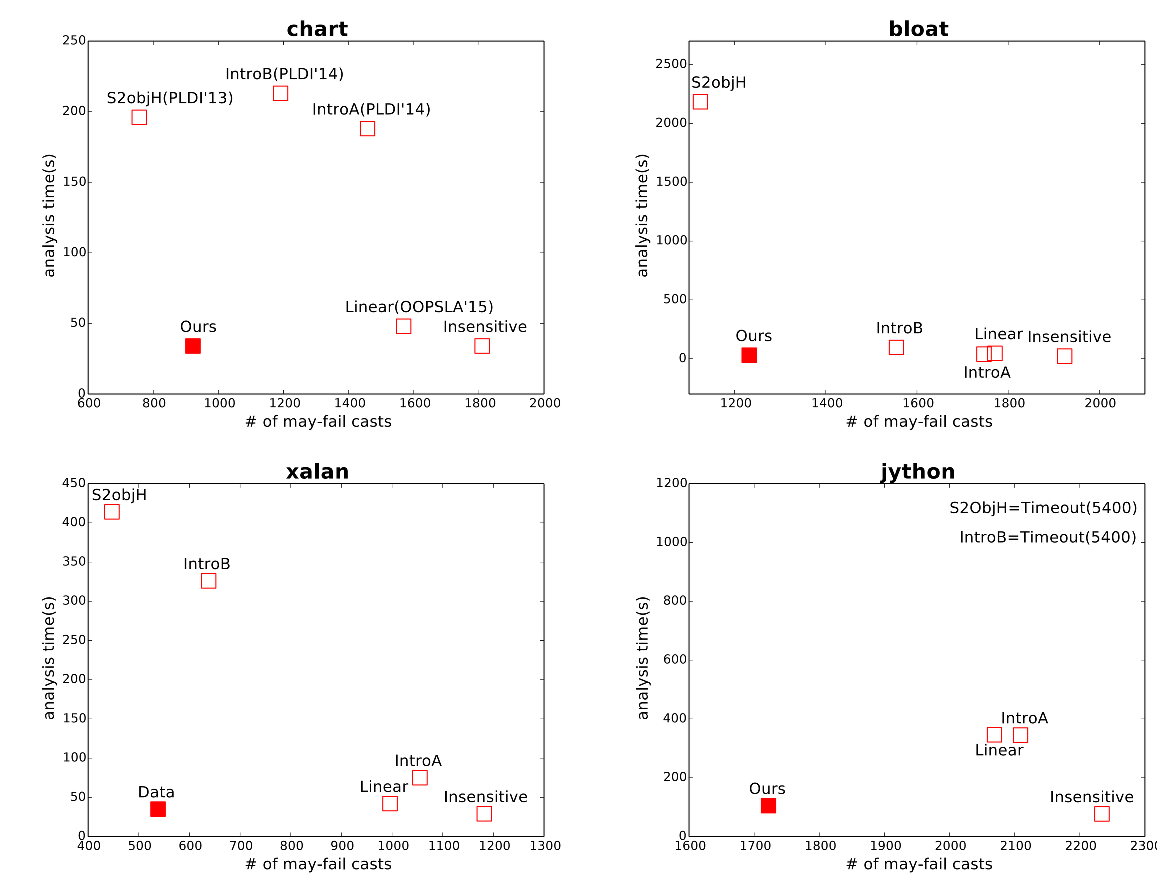
$$1 \land \neg 3 \land \neg 6 \land 8 \land \neg 9 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \neg 21 \land \neg 22 \land \neg 23 \land \neg 24 \land \neg 25$$

f1: Methods that require 1-object-sensitivity

$$(1 \land \neg 3 \land \neg 4 \land \neg 7 \land \neg 8 \land 6 \land \neg 9 \land \neg 15 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \neg 21 \land \neg 22 \land \neg 23 \land \neg 24 \land \neg 25) \lor$$
$$(\neg 3 \land \neg 4 \land \neg 7 \land \neg 8 \land \neg 9 \land 10 \land 11 \land 12 \land 13 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \neg 21 \land \neg 22 \land \neg 23 \land \neg 24 \land \neg 25) \lor$$
$$(\neg 3 \land \neg 9 \land 13 \land 14 \land 15 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \neg 21 \land \neg 22 \land \neg 23 \land \neg 24 \land \neg 25) \lor$$
$$(1 \land 2 \land \neg 3 \land 4 \land \neg 5 \land \neg 6 \land \neg 7 \land \neg 8 \land \neg 9 \land \neg 10 \land \neg 13 \land \neg 15 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \neg 21 \land \neg 22 \land \neg 23 \land \neg 24 \land \neg 25)$$

Performance

- Training with 4 small programs from DaCapo, and applied to 6 large programs
- Machine-tuning outperforms hand-tuning



Key Contributions

We achieve the improvement with two key ideas.

- A new expressive model(Disjunctive Model)
- Learning algorithm for new model

Disjunctive Model

Disjunctive model expresses set with DNF form.

Method : Features	Goal = $\{M_1, M_4\}$
$M_1 : a_1 a_2$	Disjunctive Model(Possible):
$M_2 : a_1$	$(a_1 \land a_2) \lor (\neg a_1 \land \neg a_2)$
$M_3 : a_2$	Linear Model(Impossible):
$M_4 :$	$C_1 * a_1 + C_2 * a_2$

Figure 3: Disjunctive vs Linear

With $\{a_1, a_2\}$, Disjunctive model can express the target methods, but Linear model cannot.

Learning Algorithm

Let $\Pi = \{f_1, \dots, f_k\}$ be parameters. Each f_i expresses methods to be assigned with depth i. We assign deeper depth if a method is in both f_i and $f_j (i \neq j)$. We learn Π by solving the following problem.

Optimization problem

Find parameter $\Pi = \langle f_1, \dots, f_k \rangle$ that minimizes the cost of analysis while satisfying precision constraint over training set.

Challenge

Assuming that $|S|$ is the space of possible boolean formulas over which we learn, search space of original problem is $|S|^k$. We reduce the seach space into $k * |S|$ by decomposing the original problem into k subproblems($\Psi_1 \sim \Psi_k$). Each f_i is obtained from Ψ_i and we solve them from Ψ_k to Ψ_1.

Decomposed problem Ψ_i

Let $\Pi = \langle true, true, \dots, true, f_i, f_{i+1}, \dots, f_k \rangle$. Find formula f_i that makes Π minimize the cost while satisfying precision constraint over training set.

Learning Algorithm for Ψ_i

To solve Ψ_i, we made a greedy algorithm. Let $\{a_1, \dots, a_n\}$ be atomic features. Our algorithm proceeds in the following steps:

1. f_i starts from disjunctions of 2n clauses : $f_i = a_1 \lor \neg a_1 \lor \cdots \lor a_n \lor \neg a_n$
2. Choose the most expensive clause c_i to refine.
3. Strengthen the clause c_i by conjoining an decent atom a_k with $c_i : f_i' = c_1 \lor \cdots \lor (c_i \land a_k) \lor \cdots \lor c_j$.
4. Check if f_i' satisfies precision constraint. If it is, $f_i = f_i'$.
5. Repeat 2∼4 until f_i cannot be refined.

* co-first authors