Data-Driven Context-Sensitivity for Points-to Analysis

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

Korea University

OOPSLA 2017

Heuristics in Static Analysis

- Modern static analyzers use many heuristics
  - Partial context-sensitivity
  - Partial octagon analysis
  - Partial flow-sensitivity
  - Partial path-sensitivity
  - Partial unsoundness
- Designing such heuristics is an art

Automatic Heuristic Generation

GitHub

Atlassian Bitbucket

Context-sensitivity heuristics
Flow-sensitivity heuristics
Path-sensitivity heuristics
...

- **Automatic**: Learning algorithms infer heuristics from data.
- **Powerful**: Machine-tuned heuristics outperform hand-tuned ones.
- **Stable**: Learned heuristics perform well across training data.

Context-sensitivity

```java
class D {} class E {}
class C {
  void dummy(){}
  Object id1(Object v){ return id2(v); }
  Object id2(Object v){ return v; }
}
class B {
  void m (){
    C c = new C();
    D d = (D)c.id1(new D());   //Query 1
    E e = (E)c.id1(new E());   //Query 2
    c.dummy();
  }
}
public class A {
  public static void main(String[] args){
    B b = new B();
    b.m();
    b.m();
  }
}
```

- Context-insensitivity fails to prove queries.
  - It merges `id1` invocations.
- 2-call-site sensitivity succeeds but not scale.
  - m and dummy are not worthy.

Selective 2-callsite-Sensitive Analysis

```
class D {} class E {}
class C {
  void dummy(){}
  Object id1(Object v){ return id2(v); }
  Object id2(Object v){ return v; }
}
class B {
  void m (){
    C c = new C();
    D d = (D)c.id1(new D());   //Query 1
    E e = (E)c.id1(new E());   //Query 2
    c.dummy();
  }
}
public class A {
  public static void main(String[] args){
    B b = new B();
    b.m();
    b.m();
  }
}
```

- Apply 2-callsite-sens: C.id2
- Apply 1-callsite-sens: C.id1
- Apply context-insens: main, B.m, C.dummy

- Proves all queries.

Selective 2-callsite-Sensitive Analysis

```
class D {} class E {}
class C {
  void dummy(){}
  Object id1(Object v){ return id2(v); }
  Object id2(Object v){ return v; }
}
class B {
  void m (){
    C c = new C();
    D d = (D)c.id1(new D());    //Query 1
    E e = (E)c.id1(new E());    //Query 2
    c.dummy();
  }
}
public class
  public stat
    B b = new
    b.m();
    b.m();
  }
}
```

- Apply 2-callsite-sens: C.id2
- Apply 1-callsite-sens: C.id1
- Apply context-insens: main, B.m, C.dummy

Who decide?

Data-driven approach

This talk

Data-Driven Context Sensitivity

Parametric Static Analyzer

Training Data (program)

Atomic Features $(a_1, a_2, \dots, a_{25})$

Our DD Framework

Predicate over methods
e.g.) has return stmt?,
has voidreturn stmt?

This talk

Data-Driven Context Sensitivity

Parametric Static Analyzer

Training Data (program)

Atomic Features $(a_1, a_2, \ldots, a_{25})$

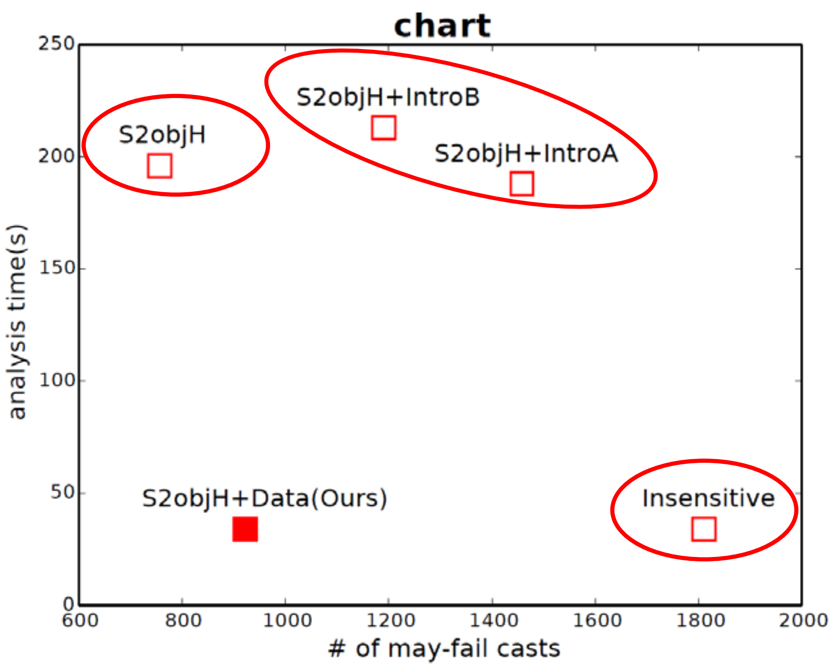
Our DD Framework

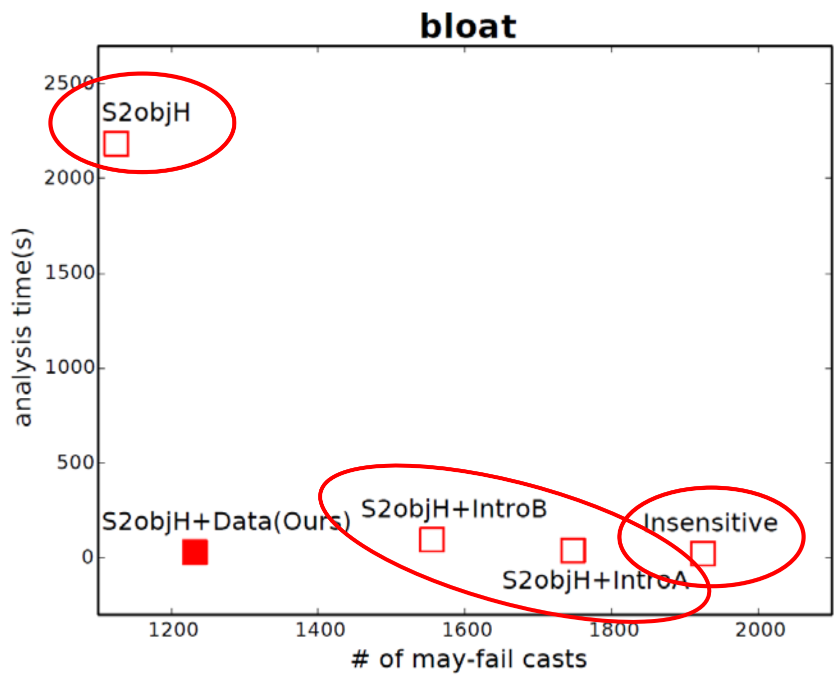
Heuristic for applying 2-hybrid-object-sensitivity

- **Methods that require 2-hybrid-object-sensitivity**
  $1 \land \neg 3 \land \neg 6 \land 8 \land \neg 9 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \neg 20 \land \ldots \land \neg 24 \land \neg 25$
- **Methods that require 1-hybrid-object-sensitivity**
  $(1 \land \neg 3 \land \neg 4 \land \neg 7 \land \neg 8 \land 6 \land \neg 9 \land \neg 15 \land \neg 16 \land \neg 17 \land \ldots \land \neg 24 \land \neg 25) \lor$
  $(\neg 3 \land \neg 4 \land \neg 7 \land \neg 8 \land \neg 9 \land 10 \land 11 \land 12 \land 13 \land \neg 16 \land \ldots \land \neg 24 \land \neg 25) \lor$
  $(\neg 3 \land \neg 9 \land 13 \land 14 \land 15 \land \neg 16 \land \neg 17 \land \neg 18 \land \neg 19 \land \ldots \land \neg 24 \land \neg 25) \lor$
  $(1 \land 2 \land \neg 3 \land 4 \land \neg 5 \land \neg 6 \land \neg 7 \land \neg 8 \land \neg 9 \land \neg 10 \land \neg 13 \land \ldots \land \neg 24 \land \neg 25)$

Performance Highlight

- Well-trained disjunctive heuristics tops hand-tuning.





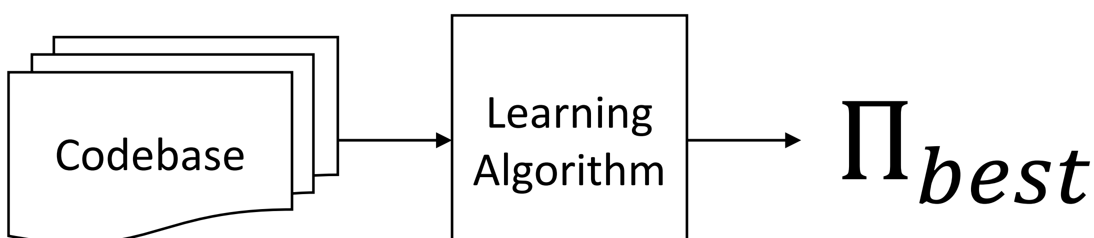
Details

Data-Driven Context Sensitivity

1. Parameterized disjunctive heuristics.

$$H_{\Pi}: M_P \rightarrow \{0,1,\ldots,k\}$$

2. Learning disjunctive heuristics for context-sensitivity.



Parametric Static Analyzer

$$F_P : A_P \to 2^Q \times Cost$$

- Context-sensitivity abstraction
- a $\in A_P : M_P \to \{0,1, \ldots, k\}$
  - k is the maximum context depth
  - M_P is a set of methods in program P

Parametric Static Analyzer

$$F_P(M_P \to \{0\})$$

- Context-insensitive analysis

Parametric Static Analyzer

$$F_P(M_P \rightarrow \{2\})$$

- Conventional 2-context-sensitive analysis
  - 2-object-sensitivity
  - 2-call-site-sensitivity
  - hybrid-2-object-sensitivity
  - 2-type sensitivity
  - ...

Parametric Static Analyzer

$$F_P(H_\Pi)$$

- Ours uses machine-tuned heuristic H.

Parametric Disjunctive Heuristics

$$H_{\Pi}: M_P \rightarrow \{0,1,\ldots,k\}$$

- Takes two steps:

1. Feature extraction: Represent M_P as feature sets.
2. Decision making: Assign context-depth to each methods.

Atomic Features

- Predicates over method definitions

$$\mathbb{A} = \{a_1, a_2, \ldots, a_{25}\}$$

$$a_i : M_P \to \mathbb{B}$$

Atomic Features

"Does the method has a specific word in its signature string?"

Signature features									
#1	"java"	#3	"sun"	#5	"void"	#7	"int"	#9	"String"
#2	"lang"	#4	"()"	#6	"security"	#8	"util"	#10	"init"

Statement features					
#11	AssignStmt	#16	BreakpointStmt	#21	LookupStmt
#12	IdentityStmt	#17	EnterMonitorStmt	#22	NopStmt
#13	InvokeStmt	#18	ExitMonitorStmt	#23	RetStmt
#14	ReturnStmt	#19	GotoStmt	#24	ReturnVoidStmt
#15	ThrowStmt	#20	IfStmt	#25	TableSwitchStmt

"Does the method has a specific type of statement in its body?"

(1) Feature Extraction

$$a(m) = \{a_i \in \mathbb{A} \mid a_i(m) = true\}$$

$$\begin{array}{lcc} a(\mathrm{M1}) & = & \{a_1\} \\ a(\mathrm{M2}) & = & \{a_2\} \\ a(\mathrm{M3}) & = & \{a_3, a_4\} \end{array}$$

- Represent methods into feature sets.

(2) Decision Making

Extracted Feature

M1	:	$\{a_1\}$
M2	:	$\{a_2\}$
M3	:	$\{a_3, a_4\}$

Learned Parameter

$$\Pi = \langle f_2 = (a_1 \wedge \neg a_2)\,, \\ f_1 = a_1 \vee a_2 \rangle$$

Initial Decision

f_2: {M1}

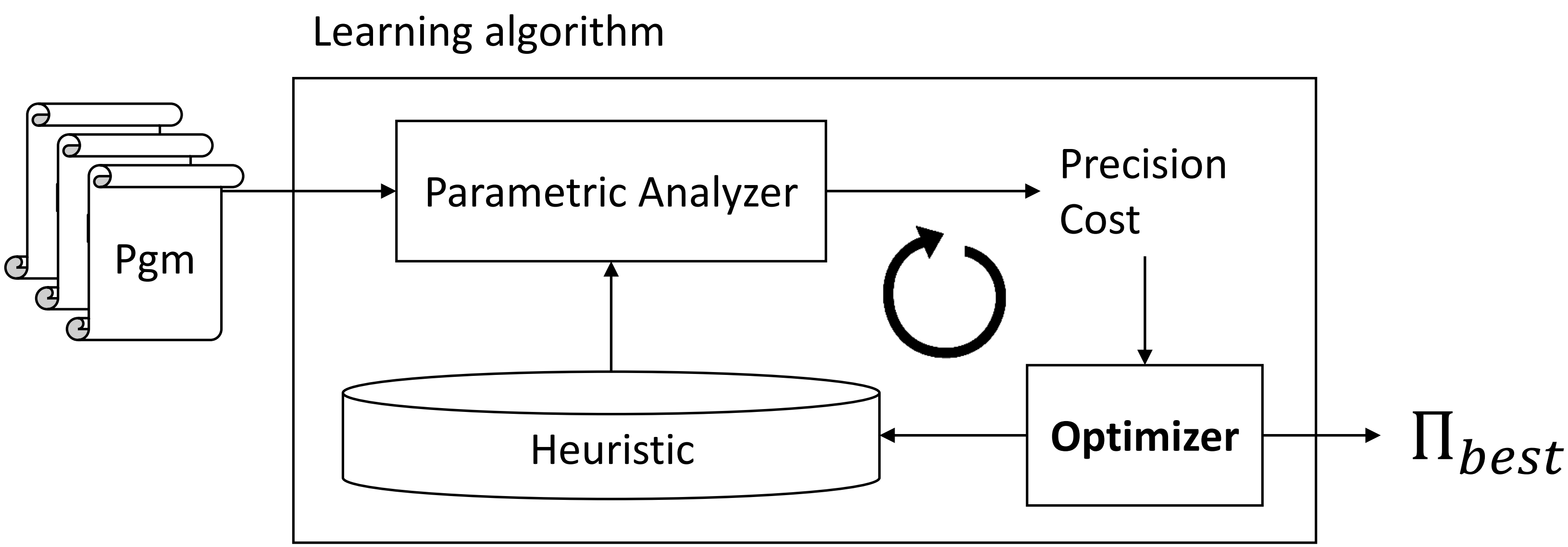
f_1: {M2, M1}

Context-sensitivity Abstraction

H_Π: {{M1} $\mapsto$ 2,
{M2} $\mapsto$ 1,
{M3} $\mapsto$ 0}

Learning Disjunctive Heuristics

- Our learning process searches good heuristic in iterative fashion.



Optimization Problem

- $\Pi = \langle f_1, f_2, \ldots, f_k \rangle$
- Find Π that minimizes

$$\sum_{P \in Pgm} \text{cost}(F_P(H_\Pi(P)))$$

- while satisfying

$$\frac{\sum_{P \in Pgm} |\text{proved}\big(F_P(H_\Pi(P))\big)|}{\sum_{P \in Pgm} |\text{proved}\big(F_P(k)\big)|} \geq \gamma$$

Problem Decomposition Reduces Search Space

Goal

Find Π that minimizes cost
while satisfying target precision

Learning Π as a whole

Set of all possible Boolean formulas

$$\Pi = \boxed{\langle f_1, f_2, \ldots, f_{k-1}, f_k \rangle} \Rightarrow |S|^k$$

Our approach: Learning one f at a time

$$\Pi = \langle true, \ldots, true, \boxed{f_k} \rangle$$

Problem Decomposition Reduces Search Space

Goal

Find Π that minimizes cost while satisfying target precision

Learning Π as a whole

Set of all possible Boolean formulas

$$\Pi = \boxed{\langle f_1, f_2, \ldots, f_{k-1}, f_k \rangle} \Rightarrow |S|^k$$

Our approach: Learning one f at a time

$$\Pi = \langle true, \ldots, true, \boxed{f_{k-1}}, f_k \rangle$$

Problem Decomposition Reduces Search Space

Goal

Find Π that minimizes cost while satisfying target precision

Learning Π as a whole

Set of all possible Boolean formulas

$$\Pi = \langle \boxed{f_1, f_2, \ldots, f_{k-1}, f_k} \rangle \Rightarrow |S|^k$$

Our approach: Learning one f at a time

$$\Pi = \langle true, \boxed{f_2}, \ldots, f_{k-1}, f_k \rangle$$

Problem Decomposition Reduces Search Space

Goal

Find Π that minimizes cost while satisfying target precision

Learning Π as a whole

$$\Pi = \langle \boxed{f_1, f_2, \dots, f_{k-1}, f_k} \rangle \Rightarrow |S|^k$$

Set of all possible Boolean formulas

Our approach: Learning one f at a time

$$\Pi = \langle \boxed{f_1}, f_2, \dots, f_{k-1}, f_k \rangle$$

Problem Decomposition Reduces Search Space

Goal

Find Π that minimizes cost while satisfying target precision

Learning Π as a whole

Set of all possible Boolean formulas

$$\Pi = \langle \boxed{f_1, f_2, \ldots, f_{k-1}, f_k} \rangle \Rightarrow |S|^k$$

Our approach: Learning one f at a time

$$\Pi = \langle f_1, f_2, \ldots, f_{k-1}, f_k \rangle \Rightarrow k \cdot |S|$$

Has Same Power? Yes.

- We have a theorem for it.
- Please consult with our paper.

Illustration of Algorithm for Searching Boolean Formula f

Initial formula

$$a_1 \vee \neg a_1 \vee a_2 \vee \neg a_2$$

The most general formula

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Performance Table

GOAL: Find f that proves all queries

$$W = \{a_1, \neg a_1, a_2, \neg a_2\}$$

Refinement Targets

Iteration 1

$$a_1 \lor \neg a_1 \lor a_2 \lor \neg a_2$$

	Proven Qs	Cost
$\boldsymbol{a_1}$	Q1, Q3	**20**
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Pick the most expensive clause

$$W = \{a_1, \neg a_1, a_2, \neg a_2\}$$

Iteration 1

$$a_1 \lor \neg a_1 \lor a_2 \lor \neg a_2$$

	Proven Qs	Cost
$\boldsymbol{a_1}$	Q1, Q3	**20**
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Remove the clause from the workset

$$W = \{\cancel{a_1}, \neg a_1, a_2, \neg a_2\}$$

Iteration 1

$(a_1 \land a_2) \lor \neg a_1 \lor a_2 \lor \neg a_2$

	Proven Qs	Cost
a_1	**Q1, Q3**	**20**
$\neg a_1$	Q2	7
a_2	**Q1, Q2, Q3**	15
$\neg a_2$	Q1, Q2, Q4	10

Refine the clause conservatively

$$W = \{\neg a_1, a_2, \neg a_2\}$$

Iteration 1

$$(a_1 \wedge a_2) \vee \neg a_1 \vee a_2 \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

$$\frac{\sum_{P \in Pgm} |\text{proved}\big(F_P(H_\Pi(P))\big)|}{\sum_{P \in Pgm} |\text{proved}\big(F_P(k)\big)|} = 1$$

Check whether the formula satisfies the precision goal

$$W = \{\neg a_1, a_2, \neg a_2\}$$

Iteration 1

$$(a_1 \wedge a_2) \vee \neg a_1 \vee a_2 \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

$$F_P\left(H_\Pi(P)\right): \{Q1, Q2, Q3, Q4\}$$

$$W = \{\neg a_1, a_2, \neg a_2\}$$

Current formula proves all queries

Iteration 1

subset

$\subset$

$$(a_1 \wedge a_2) \vee \neg a_1 \vee a_2 \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

However, the refinement didn't refine the formula

$$W = \{\neg a_1, a_2, \neg a_2\}$$

Iteration 1

subset

$\subset$

$$(a_1 \quad a_2) \vee \neg a_1 \vee a_2 \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Drop the refined clause from the formula

$$W = \{\neg a_1, a_2, \neg a_2\}$$

Iteration 2

$\neg a_1 \lor a_2 \lor \neg a_2$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

$$W = \{a_1, \neg a_1, a_2\}$$

Iteration 2

$$\neg a_1 \lor a_2 \lor \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
$\boldsymbol{a_2}$	Q1, Q2, Q3	**15**
$\neg a_2$	Q1, Q2, Q4	10

Again, pick the most expensive clause

$$W = \{a_1, \neg a_1, a_2\}$$

Iteration 2

$$\neg a_1 \lor a_2 \lor \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
$\boldsymbol{a_2}$	Q1, Q2, Q3	**15**
$\neg a_2$	Q1, Q2, Q4	10

$$\boxed{\text{Remove the clause from the workset}}$$

$$W = \{a_1, \neg a_1, \cancel{a_2}\}$$

Iteration 2

$$\neg a_1 \vee (a_1 \wedge a_2) \vee \neg a_2$$

	Proven Qs	Cost
a_1	**Q1, Q3**	20
$\neg a_1$	Q2	7
a_2	**Q1, Q2, Q3**	**15**
$\neg a_2$	Q1, Q2, Q4	10

Refine the clause conservatively

$$W = \{a_1, \neg a_1\}$$

Iteration 2

$$\neg a_1 \lor (a_1 \land a_2) \lor \neg a_2$$

$$\frac{\sum_{P \in Pgm} |\mathrm{proved}\big(F_P(H_\Pi(P))\big)|}{\sum_{P \in Pgm} |\mathrm{proved}\big(F_P(k)\big)|} = 1$$

	Proven Qs	**Cost**
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Check whether the formula satisfies the precision goal

$$W = \{a_1, \neg a_1\}$$

Iteration 2

$$\neg a_1 \vee (a_1 \wedge a_2) \vee \neg a_2$$

$$F_P(H_\Pi(P)): \{Q1, Q2, Q3, Q4\}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10

Still, the formula proves all queries

$$W = \{a_1, \neg a_1\}$$

Iteration 2

$$\neg a_1 \vee (a_1 \wedge a_2) \vee \neg a_2$$

	Proven Qs	Cost
$\boldsymbol{a_1}$	**Q1, Q3**	20
$\neg a_1$	Q2	7
$\boldsymbol{a_2}$	**Q1, Q2, Q3**	**15**
$\neg a_2$	Q1, Q2, Q4	10

Refined clause passes subset checking

$$W = \{a_1, \neg a_1\}$$

Iteration 2

$$\neg a_1 \vee (\overset{C_1}{a_1} \wedge a_2) \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	**Q1, Q2, Q4**	**13**

Record the refined clause C_1

$$W = \{a_1, \neg a_1\}$$

Iteration 2

$$\neg a_1 \vee (\overset{C_1}{a_1 \wedge a_2}) \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	**Q1, Q2, Q4**	**13**

...and add C_1 to the workset for further refinement

$$W = \{a_1, \neg a_1, (\boldsymbol{a_1 \wedge a_2})\}$$

Iteration 3

$$\neg a_1 \vee (\underbrace{a_1 \wedge a_2}_{C_1}) \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13

$$W = \{a_1, \neg a_1, (a_1 \wedge a_2)\}$$

Iteration 3

$$\neg a_1 \vee \overbrace{(a_1 \wedge a_2)}^{\boldsymbol{C_1}} \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
$\boldsymbol{C_1}$	Q1, Q2, Q4	**13**

Pick the most expensive clause

$$W = \{a_1, \neg a_1, (a_1 \wedge a_2)\}$$

Iteration 3

$$\neg a_1 \vee \overbrace{(a_1 \wedge a_2)}^{C_1} \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	**13**

Remove the clause from the workset

$$W = \{a_1, \neg a_1, \cancel{(a_1 \wedge a_2)}\}$$

Iteration 3

c_1

$$\neg a_1 \vee (a_1 \wedge a_2 \wedge \neg a_2) \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	**Q1, Q2, Q4**	10
c_1	**Q1, Q2, Q4**	**13**

Refine the clause conservatively

$$W = \{a_1, \neg a_1\}$$

Iteration 3

C_1

$$\neg a_1 \vee (a_1 \wedge a_2 \wedge \neg a_2) \vee \neg a_2$$

$$\frac{\sum_{P \in Pgm} |\text{proved}\big(F_P(H_\Pi(P))\big)|}{\sum_{P \in Pgm} |\text{proved}\big(F_P(k)\big)|} = 1$$

	Proven Qs	**Cost**
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	**Q1, Q2, Q4**	10
C_1	**Q1, Q2, Q4**	**13**

Check whether the formula satisfies the precision goal

$$W = \{a_1, \neg a_1\}$$

Iteration 3

$$\neg a_1 \lor (\overset{C_1}{a_1} \land a_2 \land \neg a_2) \lor \neg a_2$$

$$F_P\big(H_\Pi(P)\big): \{Q1, Q2, \cancel{Q3}, Q4\}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	**Q1, Q2, Q4**	10
C_1	**Q1, Q2, Q4**	**13**

This refinement failed to prove all queries

$$W = \{a_1, \neg a_1\}$$

Iteration 3

$$\neg a_1 \vee (\overset{c_1}{a_1} \wedge a_2 \wedge \cancel{\neg a_2}) \vee \neg a_2$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	**Q1, Q2, Q4**	10
c_1	**Q1, Q2, Q4**	**13**

Revert to the last state

$$W = \{a_1, \neg a_1\}$$

After iteration 4 and 5...

$$\overset{C_1}{(a_1 \wedge a_2)} \vee \overset{C_2}{(\neg a_2 \wedge \neg a_1)}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
C_2	Q1, Q2, Q3	

$$W = \{(\neg a_2 \wedge \neg a_1)\}$$

Iteration 6

$$(\underbrace{a_1 \wedge a_2}_{C_1}) \vee (\underbrace{\neg a_2 \wedge \neg a_1}_{C_2})$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
$\boldsymbol{C_2}$	Q1, Q2, Q3	**7**

Pick the most expensive clause in the workset

$$W = \{(\neg a_2 \wedge \neg a_1)\}$$

Iteration 6

$$\overbrace{(a_1 \wedge a_2)}^{C_1} \vee \overbrace{(\neg a_2 \wedge \neg a_1)}^{C_2}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
$\boldsymbol{C_2}$	Q1, Q2, Q3	**7**

Remove the clause from the workset

$$W = \{\cancel{(\neg \boldsymbol{a_2} \wedge \neg \boldsymbol{a_1})}\}$$

Iteration 6

$$\overset{C_1}{(a_1 \wedge a_2)} \vee \overset{C_2}{(\neg a_2 \wedge \neg a_1 \wedge a_2)}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	**Q1, Q2, Q3**	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
C_2	**Q1, Q2, Q3**	**7**

Refine the clause conservatively

$$W = \emptyset$$

Iteration 6

$$\overset{C_1}{(a_1 \wedge a_2)} \vee \overset{C_2}{(\neg a_2 \wedge \neg a_1 \wedge a_2)}$$

$$\frac{\sum_{P \in Pgm} |\text{proved}(F_P(H_\Pi(P)))|}{\sum_{P \in Pgm} |\text{proved}(F_P(k))|} = 1$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	**Q1, Q2, Q3**	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
C_2	**Q1, Q2, Q3**	7

Check whether the formula satisfies the precision goal

$$W = \emptyset$$

Iteration 6

$$\Big(\overset{C_1}{(a_1 \wedge a_2)} \vee \overset{C_2}{(\neg a_2 \wedge \neg a_1 \wedge a_2)}\Big)$$

$$F_P\big(H_\Pi(P)\big): \{Q1, Q2, \cancel{Q3}, Q4\}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	**Q1, Q2, Q3**	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
C_2	**Q1, Q2, Q3**	**7**

This refinement failed to prove all queries

$$W = \emptyset$$

Iteration 6

$$\overset{C_1}{(a_1 \wedge a_2)} \vee \overset{C_2}{(\neg a_2 \wedge \neg a_1 \wedge \cancel{\quad})}$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
$\boldsymbol{a_2}$	**Q1, Q2, Q3**	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
$\boldsymbol{C_2}$	**Q1, Q2, Q3**	**7**

Revert to the last state

$$W = \emptyset$$

END

$$(\overbrace{a_1 \wedge a_2}^{C_1}) \vee (\overbrace{\neg a_2 \wedge \neg a_1}^{C_2})$$

	Proven Qs	Cost
a_1	Q1, Q3	20
$\neg a_1$	Q2	7
a_2	Q1, Q2, Q3	15
$\neg a_2$	Q1, Q2, Q4	10
C_1	Q1, Q2, Q4	13
C_2	Q1, Q2, Q3	7

$\boxed{\text{Algorithm ends when there is no refineable claus.}}$

$$\boldsymbol{W = \emptyset}$$

Experiments

Settings

- Data-Driven Doop
    - Our version of Doop points-to analysis framework

- Four context sensitivities, One human-tuned heuristic
    - Selective hybrid object, object, type, call-site
    - Introspective analysis

- DaCapo Benchmark suite
    - Four small programs for training
    - One large programs for validation
    - Five large programs for testing

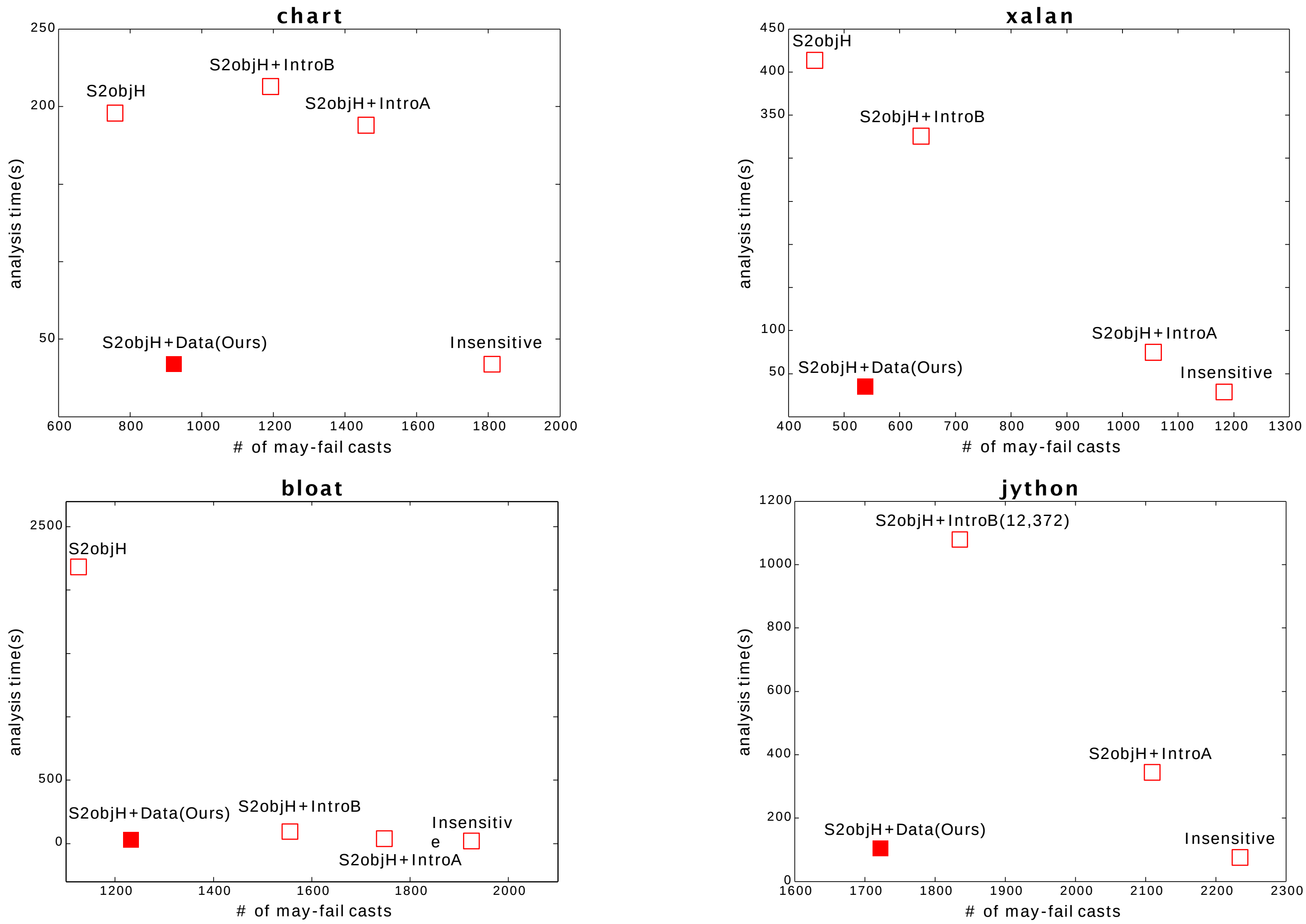
Research Questions

RQ1: Effectiveness

RQ2: Model adequacy

RQ3: Learned Heuristics

RQ1: Performance



RQ2: vs. Linear Model

- On same time budget for training, disjunctive model outperforms non-disjunctive model

Benchmarks	Non-disjunctive		Disjunctive(Ours)	
	may-fail casts	time(s)	may-fail casts	time(s)
eclipse	946	25	596	21
chart	1,569	48	937	33
bloat	1,771	46	1,232	27
xalan	996	42	539	33
jython	2069	346	1,738	104
Total	7,352	346	5,042	218

RQ3: Learned Features

- Object vs. Type
- Object sensitivity strictly more precise than Type.
- Our learning algorithm revealed the relationship.

$$
\begin{array}{lcl}
f_1 \text{ for } \textit{2objH+Data} & : & \boxed{(\mathbf{1} \wedge \mathbf{2} \wedge \neg\mathbf{3} \wedge \neg\mathbf{6} \wedge \neg\mathbf{7} \wedge \neg\mathbf{8} \wedge \neg\mathbf{9} \wedge \neg\mathbf{16} \wedge \cdots \wedge \neg\mathbf{22} \wedge \neg\mathbf{23} \wedge \neg\mathbf{24} \wedge \neg\mathbf{25}) \vee} \\
 & & (\neg 1 \wedge \neg 2 \wedge 8 \wedge 5 \wedge \neg 9 \wedge 11 \wedge 12 \wedge \cdots \wedge \neg 21 \wedge \neg 22 \wedge \neg 23 \wedge \neg 24 \wedge \neg 25) \vee \\
 & & (\neg 3 \wedge \neg 4 \wedge \neg 7 \wedge \neg 8 \wedge \neg 9 \wedge 10 \wedge 11 \wedge \cdots \wedge \neg 21 \wedge \neg 22 \wedge \neg 23 \wedge \neg 24 \wedge \neg 25) \\
\hline
f_1 \text{ for } \textit{2typeH+Data} & : & \boxed{\mathbf{1} \wedge \mathbf{2} \wedge \neg\mathbf{3} \wedge \neg\mathbf{6} \wedge \neg\mathbf{7} \wedge \neg\mathbf{8} \wedge \neg\mathbf{9} \wedge \neg\mathbf{15} \wedge \neg\mathbf{16} \wedge \cdots \wedge \neg\mathbf{22} \wedge \neg\mathbf{23} \wedge \neg\mathbf{24} \wedge \neg\mathbf{25}}
\end{array}
$$

RQ3: Learned Features

- Our algorithm revealed commonality among object-relaed context sensitivities.

f_2 for *S2objH+Data*	:	$1 \wedge \neg 3 \wedge \neg 6 \wedge 8 \wedge \neg 9 \wedge \neg 16 \wedge \neg 17 \wedge \neg 18 \wedge \cdots \wedge \neg 25$
f_2 for *2objH+Data*	:	$1 \wedge \neg 3 \wedge \neg 6 \wedge 8 \wedge \neg 9 \wedge \neg 16 \wedge \neg 17 \wedge \neg 18 \wedge \cdots \wedge \neg 25$
f_2 for *2typeH+Data*	:	$1 \wedge \neg 3 \wedge \neg 6 \wedge 8 \wedge \neg 9 \wedge \neg 16 \wedge \neg 17 \wedge \neg 18 \wedge \cdots \wedge \neg 25$
f_2 for call-site-sensitivity	:	$1 \wedge \neg 6 \wedge \neg 7 \wedge 11 \wedge 12 \wedge 13 \wedge \neg 16 \wedge \neg 17 \wedge \neg 18 \wedge \cdots \wedge \neg 25$

Conclusion

- Our approach uses a heuristic rule for context-sensitivity parameterized by Boolean formulas.
  - can express disjunctive properties of methods

- Good parameters for the heuristic can be learned from codebase by our learning algorithm

- The learned heuristics make context-sensitive points-to analysis more practical.