Learning a Strategy for Adapting a Program Analysis via Bayesian Optimization

Hakjoo Oh
Korea University

Hongseok Yang
Oxford University

Kwangkeun Yi
Seoul National University

OOPSLA 2015 @ Pittsburgh, US
Challenge in Static Analysis

- Scalability
- Precision

Diagram showing the trade-off between scalability and precision.
Challenge in Static Analysis

- Scalability
- Precision?

Graph showing trade-off between scalability and precision.
Flow-Sensitivity

\[ x = y = 0; z = 1 \]

\[ x = z \]

\[ z = z + 1 \]

\[ y = x \]

\[ \text{assert}(y > 0) \]
Flow-Sensitivity

\[
x = y = 0; z = 1
\]

\[
x = z
\]

\[
z = z + 1
\]

\[
y = x
\]

\[
\text{assert}(y > 0)
\]

\[
\begin{array}{|c|c|}
\hline
\text{x} & [0,0] \\
\hline
\text{y} & [0,0] \\
\hline
\text{z} & [1,1] \\
\hline
\end{array}
\]
Flow-Sensitivity

\[x = y = 0; z = 1\]

\[x = z\]

\[z = z + 1\]

\[y = x\]

assert\((y > 0)\)
Flow-Sensitivity

\[
\begin{align*}
x &= y = 0; z = 1 \\
x &= z \\
z &= z + 1 \\
y &= x \\
assert(y > 0)
\end{align*}
\]
Flow-Sensitivity

\begin{align*}
x &= y = 0; z = 1 \\
x &= z \\
z &= z + 1 \\
y &= x \\
\text{assert}(y > 0)
\end{align*}

\begin{tabular}{|c|c|}
\hline
x & [0,0] \\
y & [0,0] \\
z & [1,1] \\
\hline
x & [1,1] \\
y & [0,0] \\
z & [1,1] \\
\hline
x & [1,1] \\
y & [0,0] \\
z & [2,2] \\
\hline
x & [1,1] \\
y & [1,1] \\
z & [2,2] \\
\hline
\end{tabular}
Flow-Sensitivity

x = y = 0; z = 1

x = z

z = z + 1

y = x

assert(y > 0)

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[1,1]</td>
</tr>
<tr>
<td>2</td>
<td>[1,1]</td>
<td>[0,0]</td>
<td>[1,1]</td>
</tr>
<tr>
<td>3</td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>[2,2]</td>
</tr>
<tr>
<td>4</td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>[2,2]</td>
</tr>
</tbody>
</table>

precise but costly
Flow-Insensitivity

\[ x = y = 0; z = 1 \]

\[ x = z \]

\[ z = z + 1 \]

\[ y = x \]

\[ \text{assert}(y > 0) \]

\[
\begin{array}{c|c}
  x & [0, +\infty] \\
  y & [0, +\infty] \\
  z & [1, +\infty]
\end{array}
\]
Flow-Insensitivity

\[ x = y = 0; z = 1 \]

\[ x = z \]

\[ z = z + 1 \]

\[ y = x \]

assert(\( y > 0 \))

cheap but imprecise

<table>
<thead>
<tr>
<th></th>
<th>( [0, +\infty] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( [0, +\infty] )</td>
</tr>
<tr>
<td>( y )</td>
<td>( [1, +\infty] )</td>
</tr>
<tr>
<td>( z )</td>
<td>( [1, +\infty] )</td>
</tr>
</tbody>
</table>
Selective Flow-Sensitivity

FS : \{x\}  
FI : \{y,z\}
Selective Flow-Sensitivity

\[ x = y = 0; z = 1 \]

\[ x = z \]

\[ z = z + 1 \]

\[ y = x \]

assert(\( y > 0 \))

FS : \{x\}

<table>
<thead>
<tr>
<th>x</th>
<th>[0,0]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>([1, +\infty])</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>([0, +\infty])</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>z</th>
<th>([1, +\infty])</th>
</tr>
</thead>
</table>

FI : \{y, z\}

\[ y \]

\[ z \]

<table>
<thead>
<tr>
<th>y</th>
<th>([0, +\infty])</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>z</th>
<th>([1, +\infty])</th>
</tr>
</thead>
</table>
Selective Flow-Sensitivity

x=y=0; z=1

x=z

z=z+1

y=x

assert(y>0)

FS : \{x\}

<table>
<thead>
<tr>
<th>x</th>
<th>[0,0]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>[1,+\infty]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>y</th>
<th>[0,+\infty]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>z</th>
<th>[1,+\infty]</th>
</tr>
</thead>
</table>

FL : \{y,z\}

fail to prove
Selective Flow-Sensitivity

FS : \{y\}  \quad  \text{FI : \{x, z\}}

\begin{align*}
x & = y = 0; z = 1 \\
x & = z \\
z & = z + 1 \\
y & = x \\
\text{assert}(y > 0)
\end{align*}
Selective Flow-Sensitivity

```
x = y = 0 ; z = 1
```

```
x = z
```

```
z = z + 1
```

```
y = x
```

```
assert(y > 0)
```

FS : \{y\}

<table>
<thead>
<tr>
<th>y</th>
<th>[0,0]</th>
</tr>
</thead>
</table>

FI : \{x,z\}

<table>
<thead>
<tr>
<th>x</th>
<th>[0, +\infty]</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>[1, +\infty]</td>
</tr>
</tbody>
</table>

fail to prove
Selective Flow-Sensitivity

- **x=y=0; z=1**
- **x=z**
- **z=z+1**
- **y=x**
- **assert(y>0)**

<table>
<thead>
<tr>
<th>FS</th>
<th>FI</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>x,y</td>
</tr>
<tr>
<td>[1,1]</td>
<td></td>
</tr>
</tbody>
</table>

**fail to prove**

<table>
<thead>
<tr>
<th>x</th>
<th>[0,+∞]</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>[0,+∞]</td>
</tr>
</tbody>
</table>
Selective Flow-Sensitivity

\[ x = y = 0; z = 1 \]

\[ x = z \]

\[ z = z + 1 \]

\[ y = x \]

\[ \text{assert}(y > 0) \]

FS : \{y, z\}  
FI : \{x\}
Selective Flow-Sensitivity

\[
\begin{align*}
x &= y = 0; z = 1 \\
x &= z \\
z &= z + 1 \\
y &= x \\
\text{assert}(y > 0)
\end{align*}
\]

FS : \{y, z\}

FI : \{x\}

\[
\begin{array}{|c|c|}
\hline
\text{y} & [0, 0] \\
\hline
\text{z} & [1, 1] \\
\hline
\text{y} & [0, 0] \\
\hline
\text{z} & [1, 1] \\
\hline
\text{y} & [0, 0] \\
\hline
\text{z} & [2, 2] \\
\hline
\text{y} & [0, +\infty] \\
\hline
\text{z} & [2, 2] \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{x} & [0, +\infty] \\
\hline
\end{array}
\]
Selective Flow-Sensitivity

\[ x = y = 0; z = 1 \]
\[ x = z \]
\[ z = z + 1 \]
\[ y = x \]
\[ \text{assert}(y > 0) \]

FS : \{y, z\}

\begin{array}{|c|c|}
\hline
y & [0,0] \\
\hline
z & [1,1] \\
\hline
\end{array}

FI : \{x\}

\begin{array}{|c|c|}
\hline
x & [0,\infty] \\
\hline
\end{array}

fail to prove
Selective Flow-Sensitivity

\[ x=y=0; \ z=1 \]
\[ x=z \]
\[ z=z+1 \]
\[ y=x \]
\[ \text{assert}(y>0) \]

**FS : \{x,y\}**

<table>
<thead>
<tr>
<th>( x )</th>
<th>[0,0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y )</td>
<td>[0,0]</td>
</tr>
</tbody>
</table>

\[ x \in [1,\infty] \]
\[ y \in [0,0] \]

**Fl : \{z\}**

| \( z \)   | [1,\infty] |

Succeed
Finding a Good Program
Abstraction is Challenging

- Intractably large space, if not infinite
- $2^\text{Var}$ different abstractions for FS
- Most of them are too imprecise or costly
- $P(\{x,y,z\}) = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x,y\}, \{y,z\}, \{x,z\}, \{x,y,z\}\}$
Our Research

• How to efficiently find a good abstraction?

• ex) Impact pre-analysis [PLDI’14]
Learning-based Approach
Learning-based Approach

- Parameterized adaptation strategy

\[ S_w : \text{pgm} \rightarrow 2^{\text{Var}} \]
Learning-based Approach

- Parameterized adaptation strategy
  \[ S_w : \text{pgm} \rightarrow 2^{\text{Var}} \]

- Learn a good parameter \( W \) from existing codebase

\[
\begin{align*}
\text{Codebase} & \quad \Rightarrow \quad W \\
P_1, P_2, \ldots, P_m & \quad \Rightarrow \quad W
\end{align*}
\]
Learning-based Approach

- Parameterized adaptation strategy
  
  \[ S_w : \text{pgm} \rightarrow 2^{\text{Var}} \]

- Learn a good parameter \( W \) from existing codebase

- For new program \( P \), run static analysis with \( S_w(P) \)
Effectiveness

• Implemented in Sparrow, an interval analyzer for C
• Evaluated on open-source benchmarks
Effectiveness

• Implemented in Sparrow, an interval analyzer for C
• Evaluated on open-source benchmarks

Precision

FI  SFS  FS
0   70   100
Effectiveness

- Implemented in Sparrow, an interval analyzer for C
- Evaluated on open-source benchmarks

**Precision**

<table>
<thead>
<tr>
<th>FI</th>
<th>SFS</th>
<th>FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>70</td>
<td>100</td>
</tr>
</tbody>
</table>

**Cost**

<table>
<thead>
<tr>
<th>FI</th>
<th>SFS</th>
<th>FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x</td>
<td>2x</td>
<td>18x</td>
</tr>
</tbody>
</table>
Our Learning-based Approach
Static Analyzer

\[ F(p, a) \Rightarrow n \]

- number of proved assertions
- abstraction (e.g., a set of variables)
Our Learning-based Approach

1. The abstraction is determined by a parameterized strategy:

\[ S_w : \text{pgm} \rightarrow 2^{\text{Var}} \]

2. The parameter is learnt from an existing codebase:

\[
\begin{array}{c}
P_1, P_2, \ldots, P_m \\
\Rightarrow \\
W
\end{array}
\]

Codebase
1. Parameterized Strategy

\[ S_w : \text{pgm} \rightarrow 2^{\text{Var}} \]

(1) Represent program variables as feature vectors.
(2) Compute the score of each variable.
(3) Choose the top-k variables based on the score.
(1) Features

- Predicates over variables:

\[ f = \{f_1, f_2, \ldots, f_5\} \quad (f_i : \text{Var} \rightarrow \{0, 1\}) \]
Features

• Predicates over variables:

\[ f = \{f_1, f_2, \ldots, f_5\} \quad (f_i : \text{Var} \rightarrow \{0, 1\}) \]

• 45 simple syntactic features for variables: e.g,
  • local / global variable, passed to / returned from malloc, incremented by constants, etc
(1) Features

- Represent each variable as a feature vector:

\[ f(x) = \langle f_1(x), f_2(x), f_3(x), f_4(x), f_5(x) \rangle \]

\[ f(x) = \langle 1, 0, 1, 0, 0 \rangle \]
\[ f(y) = \langle 1, 0, 1, 0, 1 \rangle \]
\[ f(z) = \langle 0, 0, 1, 1, 0 \rangle \]
(2) Scoring

• The parameter $\mathbf{w}$ is a real-valued vector: e.g.,

$$\mathbf{w} = \langle 0.9, 0.5, -0.6, 0.7, 0.3 \rangle$$

• Compute scores of variables:

$$\text{score}(x) = \langle 1,0,1,0,0 \rangle \cdot \langle 0.9, 0.5, -0.6, 0.7, 0.3 \rangle = 0.3$$
$$\text{score}(y) = \langle 1,0,1,0,1 \rangle \cdot \langle 0.9, 0.5, -0.6, 0.7, 0.3 \rangle = 0.6$$
$$\text{score}(z) = \langle 0,0,1,1,0 \rangle \cdot \langle 0.9, 0.5, -0.6, 0.7, 0.3 \rangle = 0.1$$
Choose Top-k Variables

• Choose the top-k variables based on their scores:
  e.g., when k=2,

  \[
  \begin{align*}
  \text{score}(x) &= 0.3 \\
  \text{score}(y) &= 0.6 \\
  \text{score}(z) &= 0.1 \\
  \end{align*}
  \]

  \{x,y\}

• In experiments, we chosen 10% of variables with highest scores.
2. Learn a Good Parameter

\[ \text{Find } w \text{ that maximizes } \sum_{P_i} F(P_i, S_w(P_i)) \]
Learning via Random Sampling

repeat $N$ times

pick $w \in \mathbb{R}^n$ randomly

evaluate $\sum_{P_i} F(P_i, S_w(P_i))$

return best $w$ found
Learning via Random Sampling

![Bar chart](chart.png)

Table 4. Effectiveness of our method for flow-sensitivity. prove: the number of proved queries in each analysis (FI: flow-insensitivity, FS: flow-sensitivity, partial FS: partial flow-sensitivity). quality: the ratio of proved queries among the queries that require flow-sensitivity. cost: cost increase compared to the FI analysis.

<table>
<thead>
<tr>
<th>Trial</th>
<th>FI</th>
<th>FS</th>
<th>partial FS</th>
<th>prove</th>
<th>quality</th>
<th>sec</th>
<th>cost</th>
<th>prove</th>
<th>quality</th>
<th>sec</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>6,383</td>
<td>9,237</td>
<td>80.3%</td>
<td>2,788</td>
<td>48</td>
<td>75.7%</td>
<td>3,383</td>
<td>55</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>5,788</td>
<td>8,287</td>
<td>72.4%</td>
<td>3,383</td>
<td>57</td>
<td>65.9%</td>
<td>3,903</td>
<td>93</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>6,148</td>
<td>8,737</td>
<td>76.3%</td>
<td>3,023</td>
<td>48</td>
<td>60.9%</td>
<td>3,483</td>
<td>99</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>6,138</td>
<td>9,883</td>
<td>73.7%</td>
<td>3,033</td>
<td>38</td>
<td>63.7%</td>
<td>3,286</td>
<td>51</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>7,343</td>
<td>10,082</td>
<td>84.4%</td>
<td>1,828</td>
<td>28</td>
<td>81.4%</td>
<td>2,103</td>
<td>54</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>31,800</td>
<td>39,625</td>
<td>84.0%</td>
<td>14,055</td>
<td>218</td>
<td>69.6%</td>
<td>16,105</td>
<td>374</td>
</tr>
</tbody>
</table>

Table 5. Effectiveness for Flow-sensitivity + Context-sensitivity.

The implementation of our learning algorithm is available at [http://prl.korea.ac.kr/~hakjoo/research/oopsla15/].
Our Approach: Learning via Bayesian Optimization

• A powerful method for solving difficult optimization problems.

• Especially powerful when the objective function is expensive to evaluate.

• Key idea: use a probabilistic model to reduce the number of objective function evaluations.
Learning via Bayesian Optimization

repeat N times

select a promising w using the model

evaluate $\sum_{P_i} F(P_i, S_w(P_i))$

update the probabilistic model

return best w found

• Probabilistic model: Gaussian processes
• Selection strategy: Expected improvement
Learning via Bayesian Optimization

![Graph showing distribution of quality with count on the y-axis and quality on the x-axis.]


<table>
<thead>
<tr>
<th>Trial</th>
<th>FI Probe</th>
<th>FS Probe</th>
<th>Partial FS Probe</th>
<th>Quality</th>
<th>Cost Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6,383</td>
<td>7,316</td>
<td>7,089</td>
<td>75.7%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5,788</td>
<td>7,422</td>
<td>7,219</td>
<td>87.6%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6,148</td>
<td>7,842</td>
<td>7,595</td>
<td>85.4%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6,138</td>
<td>7,895</td>
<td>7,599</td>
<td>83.2%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7,343</td>
<td>9,150</td>
<td>8,868</td>
<td>84.4%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOTAL</th>
<th>FI Probe</th>
<th>FS Probe</th>
<th>Partial FS Probe</th>
<th>Quality</th>
<th>Cost Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>31,800</td>
<td>39,625</td>
<td>38,370</td>
<td></td>
<td>84.0%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Effectiveness for Flow-sensitivity + Context-sensitivity.

Figure 2. Comparison of Bayesian optimization with random sampling.
Random Sampling vs Bayesian Optimization

The random sampling method converged to the quality of the best parameter found in fewer evaluations compared to Bayesian optimization. However, Bayesian optimization achieved a higher quality of the best parameter found with the same budget. The difference in quality is more pronounced in later evaluations, indicating that Bayesian optimization is more effective in finding better parameters over time.
Experiments

• Sparrow: a C static analyzer for buffer-overrun checking

• Tune partial flow- and context-sensitivity of Sparrow
  • 10% of program variables for flow-sensitivity
  • 10% of procedures for context-sensitivity

• 30 open-source C programs (1K ~ 100KLoC)
  • 20 programs for training
  • 10 programs for testing
## Performance

### Flow-Sensitivity (12 hour time budget)

<table>
<thead>
<tr>
<th>Trial</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FI</td>
<td>FS</td>
</tr>
<tr>
<td></td>
<td>prove</td>
<td>prove</td>
</tr>
<tr>
<td>1</td>
<td>6,383</td>
<td>7,316</td>
</tr>
<tr>
<td>2</td>
<td>5,788</td>
<td>7,422</td>
</tr>
<tr>
<td>3</td>
<td>6,148</td>
<td>7,842</td>
</tr>
<tr>
<td>4</td>
<td>6,138</td>
<td>7,895</td>
</tr>
<tr>
<td>5</td>
<td>7,343</td>
<td>9,150</td>
</tr>
<tr>
<td>TOTAL</td>
<td>31,800</td>
<td>39,625</td>
</tr>
</tbody>
</table>
### Flow-Sensitivity (12 hour time budget)

<table>
<thead>
<tr>
<th>Trial</th>
<th>Training</th>
<th></th>
<th></th>
<th>Testing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FI prove</td>
<td>FS prove</td>
<td>partial FS prove</td>
<td>quality</td>
<td>FI prove</td>
</tr>
<tr>
<td>1</td>
<td>6,383</td>
<td>7,316</td>
<td>7,089</td>
<td>75.7 %</td>
<td>2,788</td>
</tr>
<tr>
<td>2</td>
<td>5,788</td>
<td>7,422</td>
<td>7,219</td>
<td>87.6 %</td>
<td>3,383</td>
</tr>
<tr>
<td>3</td>
<td>6,148</td>
<td>7,842</td>
<td>7,595</td>
<td>85.4 %</td>
<td>3,023</td>
</tr>
<tr>
<td>4</td>
<td>6,138</td>
<td>7,895</td>
<td>7,599</td>
<td>83.2 %</td>
<td>3,033</td>
</tr>
<tr>
<td>5</td>
<td>7,343</td>
<td>9,150</td>
<td>8,868</td>
<td>84.4 %</td>
<td>1,828</td>
</tr>
<tr>
<td>TOTAL</td>
<td>31,800</td>
<td>39,625</td>
<td>38,370</td>
<td>84.0 %</td>
<td>14,055</td>
</tr>
</tbody>
</table>

### Flow-Sensitivity + Context-Sensitivity (12hrs)

<table>
<thead>
<tr>
<th>Trial</th>
<th>Training</th>
<th></th>
<th></th>
<th>Testing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FICI prove</td>
<td>FSCS prove</td>
<td>partial FSCS prove</td>
<td>quality</td>
<td>FICI prove</td>
</tr>
<tr>
<td>1</td>
<td>6,383</td>
<td>9,237</td>
<td>8,674</td>
<td>80.3 %</td>
<td>2,788</td>
</tr>
<tr>
<td>2</td>
<td>5,788</td>
<td>8,287</td>
<td>7,598</td>
<td>72.4 %</td>
<td>3,383</td>
</tr>
<tr>
<td>3</td>
<td>6,148</td>
<td>8,737</td>
<td>8,123</td>
<td>76.3 %</td>
<td>3,023</td>
</tr>
<tr>
<td>4</td>
<td>6,138</td>
<td>9,883</td>
<td>8,899</td>
<td>73.7 %</td>
<td>3,033</td>
</tr>
<tr>
<td>5</td>
<td>7,343</td>
<td>10,082</td>
<td>10,040</td>
<td>98.5 %</td>
<td>1,828</td>
</tr>
<tr>
<td>TOTAL</td>
<td>31,800</td>
<td>46,226</td>
<td>43,334</td>
<td>80.0 %</td>
<td>14,055</td>
</tr>
</tbody>
</table>
Insights on Flow-Sensitivity

• Relative importance between program features:

![Graph showing relative importance of program features with bars for each feature number]
Insights on Flow-Sensitivity

• Relative importance between program features:
Insights on Flow-Sensitivity

• Relative importance between program features:

- local variable
- used as an array index

![Graph showing the relative importance between program features](image)
Insights on Flow-Sensitivity

• Relative importance between program features:

- local variable
- used as an array index
- modified inside loops
Insights on Flow-Sensitivity

• Typical scenario where flow-sensitivity helps:

```c
int mirror[7];
int i = unknown;
for (i=1;i<7;i++)
  if (mirror[i-1] == '1') ...
```
Insights on Flow-Sensitivity

- Typical scenario where flow-sensitivity helps:

```c
int mirror[7];
int i = unknown;
for (i=1;i<7;i++)
    if (mirror[i-1] == '1') ...
```
Insights on Flow-Sensitivity

- Typical scenario where flow-sensitivity helps:

```
1  int mirror[7];
2  int i = unknown;
3  for (i=1;i<7;i++)
4    if (mirror[i-1] == '1') ...
```
Insights on Flow-Sensitivity

• Typical scenario where flow-sensitivity helps:

```
1  int mirror[7];
2  int i = unknown;
3  for (i=1;i<7;i++)
4      if (mirror[i-1] == '1') ...
```
Insights on Flow-Sensitivity

• Also provide insights difficult to find manually:

```c
int pos = unknown;
if (!pos)
    path[pos] = 0;
```

• Over the entire codebase, the feature is a strong indicator for flow-“insensitivity”
Insights on Flow-Sensitivity

• Also provide insights difficult to find manually:

```plaintext
1  int pos = unknown;
2  if (!pos)
3    path[pos] = 0;
```

• Over the entire codebase, the feature is a strong indicator for flow-“insensitivity”
Summary

Our Learning-based Approach

- First machine learning-based approach
  - formulated as an optimization problem
  - solved by Bayesian optimization
- Effective: 70% precision with 2x cost
- Generally applicable to any static analysis
Summary

Our Learning-based Approach

• First machine learning-based approach

  • formulated as an optimization problem

  • solved by Bayesian optimization

• Effective: 70% precision with 2x cost

• Generally applicable to any static analysis

Thank you