예비분석과 기계학습을 이용하여 선별적으로 정확하게 정적 분석

허 기홍
서울대학교
(오학주, 이원찬, 양홍석, 이광근)
2017. 2. 20 @ Codemind
• Sound, precise yet scalable static analysis
목표

• Sound, precise yet scalable static analysis

(2007)
• Sound, precise yet scalable static analysis

General Sparse Analysis Framework
[PLDI’12, TOPLAS’14]
목표

• Sound, precise yet scalable static analysis

Selective X-sensitive Analysis
- by Impact Pre-analysis [PLDI’14, TOPLAS’16, SPE’17]
- by Machine Learning [SAS’16]
실험 결과

• 선별적으로 문맥을 구분하는 분석 (Selective Context-sensitive Analysis)
실험 결과

- 선별적으로 변수 관계를 추적하는 분석 (Selective Relational Analysis)

![Graph 1](#Alarm)

- Baseline Syntactic, IMPCT, ML

![Graph 2](Time)

- Baseline Syntactic, IMPCT, ML
 목표

- Sound, precise yet scalable static analysis

Selectively Unsound Analysis [ICSE’17]

Sparrow
The Early Bird

(2017)
실험 결과

• 선별적으로 안전한 분석 (loop, lib call 안전성 조절)

Buffer Overrun Analysis

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Selective</th>
<th>Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>#True Alarm</td>
<td>100</td>
<td>75</td>
<td>50</td>
</tr>
<tr>
<td>#False Alarm</td>
<td>100</td>
<td>75</td>
<td>25</td>
</tr>
</tbody>
</table>

Taint Analysis

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Selective</th>
<th>Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>#True Alarm</td>
<td>100</td>
<td>75</td>
<td>50</td>
</tr>
<tr>
<td>#False Alarm</td>
<td>100</td>
<td>75</td>
<td>25</td>
</tr>
</tbody>
</table>
핵심 기술

- 효율적인 정적 분석을 위해 적절한 요약을 찾기
  - 정적 분석이론기반(예비분석)/통계기반(기계학습)
  - 대상: 변수관계, 문맥(context), 흐름(flow), 안전성(soundness), 등

\[ F \in Pgm \times \Pi \rightarrow A \]
핵심 기술

• 효율적인 정적 분석을 위해 적절한 요약을 찾기
핵심 기술

• 효율적인 정적 분석을 위해 적절한 요약을 찾기
핵심 기술

- 효율적인 정적 분석을 위해 적절한 요약을 찾기

sound analysis

selective analysis

sound & precise analysis

unsound analysis
예비 분석을 이용하여 선별적으로 정확하게 정적 분석
(Selective X-sensitive Analysis by Impact Pre-analysis)

- Selective Context-Sensitivity Guided by Impact Pre-Analysis, PLDI’14
- Selective X-Sensitive Analysis Guided by Impact Pre-Analysis, TOPLAS’16
- Selective Conjunction of Context-sensitivity and Octagon Domain toward Scalable and Precise Global Static Analysis, SPE’17
선별적으로 정확한 분석

- 특정 X 를 필요한 곳에서만 정확하게 분석하는 방법

- X : 문맥, 관계 등 정확성을 높이지만 비싼 기술

부정확한 분석
선별적으로 정확한 분석
정확한 분석
관계 분석

• 변수 사이의 관계를 특정한 형태로 분석

• e.g.) octagon analysis : \( (\pm x) - (\pm y) \leq c \)

```java
int a = b;
int c = input(); // User input
for (i = 0; i < b; i++) {
    assert (i < a); // Query 1
    assert (i < c); // Query 2
}
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*Consider x-y \leq c only, for simplicity*
관계 분석

- 변수 사이의 관계를 특정한 형태로 분석

- e.g.) octagon analysis: \((\pm x) - (\pm y) \leq c\)

```plaintext
1 int a = b;
2 int c = input(); // User input
3 for (i = 0; i < b; i++) {
4   assert (i < a); // Query 1
5   assert (i < c); // Query 2
6 }
```

![Octagon Analysis Diagram]
관계 분석

• 변수 사이의 관계를 특정한 형태로 분석

• e.g.) octagon analysis: \((\pm x) - (\pm y) \leq c\)

```java
int a = b;
int c = input(); // User input
for (i = 0; i < b; i++) {
    assert (i < a); // Query 1
    assert (i < c); // Query 2
}
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

\{a, b, c, i\}

\[a - c \leq \infty\]
\[b - c \leq \infty\]

\[c - a \leq \infty\]
\[c - b \leq \infty\]
관계 분석

・ 변수 사이의 관계를 특정한 형태로 분석

・ e.g.) octagon analysis: $$\pm x - \pm y \leq c$$

```java
1  int a = b;
2  int c = input();  // User input
3  for (i = 0; i < b; i++) {
4      assert (i < a);  // Query 1
5      assert (i < c);  // Query 2
6  }
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>i - b ≤ -1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>-1</td>
</tr>
<tr>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

\{a, b, c, i\}
관계 분석

• 변수 사이의 관계를 특정한 형태로 분석

• e.g.) octagon analysis: \((\pm x) - (\pm y) \leq c\)

```java
1  int a = b;
2  int c = input();       // User input
3  for (i = 0; i < b; i++) {
4      assert (i < a);    // Query 1
5      assert (i < c);    // Query 2
6  }
```

![Octagon analysis matrix](image)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>(\infty)</td>
<td>-1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>(\infty)</td>
<td>-1</td>
</tr>
<tr>
<td>c</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>0</td>
<td>(\infty)</td>
</tr>
<tr>
<td>i</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>(\infty)</td>
<td>0</td>
</tr>
</tbody>
</table>

\(i - a \leq -1\)

\{a, b, c, i\}
관계 분석

- 변수 사이의 관계를 특정한 형태로 분석

- e.g.) octagon analysis: \((\pm x) - (\pm y) \leq c\)

```
1    int a = b;
2    int c = input();    // User input
3    for (i = 0; i < b; i++) {
4        assert (i < a);  // Query 1
5        assert (i < c);  // Query 2
6    }
```

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>-1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>-1</td>
</tr>
<tr>
<td>c</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>i</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

\[i - c \leq \infty\]

\{a, b, c, i\}
관계 분석

• 변수 사이의 관계를 특정한 형태로 분석

• e.g.) octagon analysis: \((\pm x) - (\pm y) \leq c\)

```plaintext
1    int a = b;
2    int c = input();       // User input
3    for (i = 0; i < b; i++) {
4        assert (i < a);     // Query 1
5        assert (i < c);     // Query 2
6    }
```

![Octagon Analysis Diagram]

\{a, b, c, i\}

c가 필요한가?
선별적 관계 분석

- 선별적으로 변수사이의 관계를 분석

- 같은 묶음 (cluster) 안에 있는 변수 사이 관계만

```java
int a = b;
int c = input(); // User input
for (i = 0; i < b; i++) {
    assert (i < a); // Query 1
    assert (i < c); // Query 2
}
```

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

{a, b, i}

-∞ ≤ c ≤ +∞

{c}
예비 분석

• X의 효과를 가늠하는 정적 분석

• X는 가장 정확하게 + 나머지는 과감히 요약

Impact Pre-analysis

Main Analysis

X (Relational)

Other Precision Aspects
예비 분석

• 예) octagon analysis:
모든 변수의 관계 추적 + 차이는 과감히 요약

1: int a = b;
2: int c = input();
3: for (i = 0; i < b; i ++) {
4:   assert (i < a);
5:   assert (i < c);
6: }

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>★</td>
<td>★</td>
<td>T</td>
<td>★</td>
</tr>
<tr>
<td>★</td>
<td>★</td>
<td>T</td>
<td>★</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>★</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>★</td>
</tr>
</tbody>
</table>

무한할 수도 ($\mathbb{Z} \cup \{+\infty\}$)
반드시 유한 ($\mathbb{Z}$)
변수 묶음

• 묶음: ★ 을 만드는데 기여한 변수 묶음

• 변수 의존관계를 따라 간단히 계산 가능

1: int a = b;
2: int c = input();
3: for (i = 0; i < b; i ++) {
4:   assert (i < a);
5:   assert (i < c);
6: }

\[
\begin{align*}
\text{a} &= \text{b} \\
\text{c} &= \text{input()} \\
\text{i} &= 0 \\
\text{i} &< \text{b} \\
\text{Q: i} &< \text{a?} \\
\text{i} &\text{++}
\end{align*}
\]

\[
\begin{align*}
\{a, b, i\}
\end{align*}
\]
Acknowledgements

We believe that our approach can be used for developing other syntactic variable packing and reduces the analysis cost by 81%.

Our selective octagon relational analysis with octagons using the same idea of impact pre-analysis. We proposed a method of designing a selective "X-sensitive" analysis-parameter inference tailored to the queries (our analysis is run once for the entire set of queries, we compute the exhaustive solution with an abstraction inference). The problem is how to find a set of minimal, or at least sufficient parameters to tune in static analysis, to improve either precision or scalability. Zhang et al. [21] followed this approach, presented two program analyses that separate static and dynamic analysis, and our experiments showed that our selective octagon analysis is similar to the existing octagon analysis, where the selection is guided by an impact pre-analysis.

Table 2.

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>#Variable</th>
<th>#Query</th>
<th>Syntactic Packing Approach</th>
<th>Our Selective Relational Analysis</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>proven</td>
<td>time</td>
<td>mem</td>
</tr>
<tr>
<td>calculator-1.0</td>
<td>298</td>
<td>197</td>
<td>10</td>
<td>2</td>
<td>0.3</td>
<td>63</td>
</tr>
<tr>
<td>spell-1.0</td>
<td>2,213</td>
<td>531</td>
<td>16</td>
<td>1</td>
<td>4.8</td>
<td>109</td>
</tr>
<tr>
<td>barcode-0.96</td>
<td>4,460</td>
<td>2,002</td>
<td>37</td>
<td>16</td>
<td>11.8</td>
<td>221</td>
</tr>
<tr>
<td>httptunnel-3.3</td>
<td>6,174</td>
<td>1,908</td>
<td>28</td>
<td>16</td>
<td>26.0</td>
<td>220</td>
</tr>
<tr>
<td>bc-1.06</td>
<td>13,093</td>
<td>2,194</td>
<td>10</td>
<td>2</td>
<td>247.1</td>
<td>945</td>
</tr>
<tr>
<td>tar-1.17</td>
<td>20,258</td>
<td>5,332</td>
<td>17</td>
<td>7</td>
<td>1,043.2</td>
<td>1,311</td>
</tr>
<tr>
<td>less-382</td>
<td>23,822</td>
<td>4,482</td>
<td>13</td>
<td>0</td>
<td>3,031.5</td>
<td>1,439</td>
</tr>
<tr>
<td>a2ps-4.14</td>
<td>64,590</td>
<td>16,531</td>
<td>11</td>
<td>0</td>
<td>29,479.3</td>
<td>2,304</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>135,008</td>
<td>33,177</td>
<td>142</td>
<td>44</td>
<td>33,840.3</td>
<td>6,611</td>
</tr>
</tbody>
</table>

References


수험 결과

- 선별적 관계 분석
- 기존 기술 대비, 쿼리 3배 많이 증명, 분석 시간 81% 절감
- cf.) 모든 관계 분석 (fully relational)은 2KLOC 까지
예) 문맥 구분 (Context-sensitivity)

```c
int h(n) { return n; }
void f(s) {
    1:   p = h(s);
        assert(p > 1);    // Q1
    2:   q = h(input());
        assert(q > 1);    // Q2
}
3: void g() { f(8); }
void main(){
4:   f(4);
5:   g();
6:   g();
}
```
예) 문맥을 위한 예비 분석

모든 인터벌
음이 아닌 인터벌 (e.g. [1,5], [0, ∞])

모든 문맥 구분 분석
예비 분석
실험 결과

- 선별적 문맥 구분 분석
- 거짓경보 24% 감소, 분석 시간 28% 증가
- cf.) 3-CFA : 같은 정확도, 분석시간 1300% 증가

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Proc</th>
<th>Context-Insensitive</th>
<th>Our Selective Context-Sensitive Analysis</th>
<th>Alarm reduction</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>#alarm</td>
<td>time</td>
<td>#alarm</td>
<td>pre</td>
</tr>
<tr>
<td>spell-1.0</td>
<td>2,213</td>
<td>31</td>
<td>58</td>
<td>0.6</td>
<td>30</td>
<td>0.1</td>
</tr>
<tr>
<td>bc-1.06</td>
<td>13,093</td>
<td>134</td>
<td>606</td>
<td>14.0</td>
<td>483</td>
<td>1.9</td>
</tr>
<tr>
<td>tar-1.17</td>
<td>20,258</td>
<td>222</td>
<td>940</td>
<td>42.1</td>
<td>799</td>
<td>5.4</td>
</tr>
<tr>
<td>less-382</td>
<td>23,822</td>
<td>382</td>
<td>654</td>
<td>123.0</td>
<td>562</td>
<td>3.3</td>
</tr>
<tr>
<td>sed-4.0.8</td>
<td>26,807</td>
<td>294</td>
<td>1,325</td>
<td>107.5</td>
<td>1,238</td>
<td>7.4</td>
</tr>
<tr>
<td>make-3.76</td>
<td>27,304</td>
<td>191</td>
<td>1,500</td>
<td>84.4</td>
<td>1,028</td>
<td>7.1</td>
</tr>
<tr>
<td>grep-2.5</td>
<td>31,495</td>
<td>153</td>
<td>735</td>
<td>12.1</td>
<td>653</td>
<td>2.4</td>
</tr>
<tr>
<td>wget-1.9</td>
<td>35,018</td>
<td>434</td>
<td>1,307</td>
<td>69.0</td>
<td>942</td>
<td>12.5</td>
</tr>
<tr>
<td>a2ps-4.14</td>
<td>64,590</td>
<td>980</td>
<td>3,682</td>
<td>118.1</td>
<td>2,121</td>
<td>29.5</td>
</tr>
<tr>
<td>bison-2.5</td>
<td>101,807</td>
<td>1,427</td>
<td>1,894</td>
<td>136.3</td>
<td>1,742</td>
<td>34.6</td>
</tr>
<tr>
<td>Total</td>
<td>346,407</td>
<td>4,248</td>
<td>12,701</td>
<td>707.1</td>
<td>9,598</td>
<td>104.2</td>
</tr>
</tbody>
</table>
여러 X가 동시에 필요하다면?
Selective Relational and Context-Sensitive Analysis

1. Introduction

Recently, selective context-sensitive analysis and selective relational analysis such as interval analysis. This is because the value of an abstract bound is infinite (\(\pm\infty\)) or may be infinite (\(\infty\)) or may be infinite (\(\infty\)). Our backward pre-analysis infers the weakest pre-conditions that are sufficient to select each call site at line 18, 19, 20 and 21 while merging all other contexts, and then yield the impact pre-analysis compositionally using procedure summary. We select queries that are judged promising by the pre-analysis: the abstract domain used in the pre-analysis only distinguishes whether the difference of two variables is finite (\(\Delta\)), our abstract domain describes an abstract constraint (\(\Delta\)), or may be abstract constraint (\(\Delta\)). We present a method for selectively applying relational and context-sensitive analysis. Our technique is based on a pre-analysis approach using procedure summary and show practical experimental results. In theory, we design a fully relational and context-sensitive analysis. Though the pre-analysis aggressively abstracts the numerical analysis. However, there are still some parts that need both context-sensitive analysis and selective relational-analysis can abstract constraints (\(\Delta\)), \(i\), \(j\), \(k\), \(l\), \(m\), \(n\), \(o\), \(p\), \(q\), \(r\), \(s\), \(t\), \(u\), \(v\), \(w\), \(x\), \(y\), \(z\), \(A\), \(B\), \(C\), \(D\), \(E\), \(F\), \(G\), \(H\), \(I\), \(J\), \(K\), \(L\), \(M\), \(N\), \(O\), \(P\), \(Q\), \(R\), \(S\), \(T\), \(U\), \(V\), \(W\), \(X\), \(Y\), \(Z\).

Example Program

```c
int inc(int i) { r = i + 1; return r; }
void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x); // Query 1
}
void g(int z){
    n = input();
    assert(n < z); // Query 2
}
void main(){
    a = inc(b);
    c = inc(d);
    f(a, b);
    f(c, d);
    g(a);
    g(c);
}
```

\((b = i) \lor (d = i) = T\)

\[ b = i \quad \text{inc}(int \ i) \quad d = i \]

\[ a = \text{inc}(b); \quad c = \text{inc}(d); \]

\[ a - b <= +\infty \quad \text{and} \quad c - d <= +\infty \]
Abstract

We present a method for selective relational and context-sensitive analysis. Our goal is to effectively select promising queries that are likely to prove the query. It is due to that the analysis excessively abstracts the numerical information, and keeping track of the relation only between callees at each call site.

The underlying principle is the same as previous approaches, but we provide a more practical method to perform pre-analysis. Recently, selective context-sensitive analysis and selective relational analysis are successfully achieved by the impact pre-analysis compositionally using procedure summary. We select queries that are judged promising by the pre-analysis: for example, the first query is selected if $n^2 - n - 1 > 0$. For example, the assignment at line 6 produces two abstract constraints as $r - b = O(2^n)$ or may be infinite ($r - b$ is given, the procedure yields the corresponding post-conditions). Our backward pre-analysis computes the abstract constraint as $r - b \leq O(2^n)$. For example, the assignment at line 6 produces two abstract constraints as $r - b = O(2^n)$ or may be infinite ($r - b$ is given, the procedure yields the corresponding post-conditions). Our backward pre-analysis computes the abstract constraint as $r - b \leq O(2^n)$.

Example Program

```c
int inc(int i) { r = i + 1; return r; }

void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x); // Query 1
}

void g(int z){
    n = input();
    assert(n < z); // Query 2
}

void main(){
    a = inc(b);
    c = inc(d);
    f(a, b);
    f(c, d);
    g(a);
    g(c);
}
```

We select queries that are judged promising by the pre-analysis: for example, the first query is selected if $n^2 - n - 1 > 0$. For example, the assignment at line 6 produces two abstract constraints as $r - b = O(2^n)$ or may be infinite ($r - b$ is given, the procedure yields the corresponding post-conditions). Our backward pre-analysis computes the abstract constraint as $r - b \leq O(2^n)$.
해결책

• 전체 변수의 관계와 모든 문맥을 구분하는 예비 분석
  
• 값 도메인은 본분석보다 훨씬 요약

• 개별 (modular) 분석을 이용해 효율적으로
  
  • 각 함수의 입출력을 따로 요약하고 조립

• 정확도 향상이 있을 부분에서 필요한 관계, 문맥 추출
5.4 Running Selectively Relational Octagon Analysis

Let \( S \) be the bookkeeping information at program point (3). (3) shows that the octagonal relation between \( a, b, s, p, i \) is necessary for the relation between \( a, b, i, s, p \). Finally, \( a, b, s, p, i \) will be finite at program point (5).

\[
\begin{align*}
\text{dstrsplit}(\text{char}** \text{list}, \text{int}** \text{length}) & \{ \\
& \quad \text{*list} = \text{malloc}(\text{length}); \\
& \}
\end{align*}
\]

\[
\begin{align*}
\text{a_find_input_rages}(...) & \{ \\
& \quad \text{dstrsplit}(&\text{list1}, &\text{length1}); \\
& \quad i = 0; \\
& \quad \text{while}(i < \text{length1}) \{ \\
& \quad \quad \text{list1}[i] = \ldots \\
& \quad \} \\
& \}
\end{align*}
\]

\[
\begin{align*}
\text{find_field}(...) & \{ \\
& \quad \text{dstrsplit}(&\text{list2}, &\text{length2}); \\
& \quad i = 0; \\
& \quad \text{while}(i < \text{length2}) \{ \\
& \quad \quad \text{list2}[i] = \ldots \\
& \quad \} \\
& \}
\end{align*}
\]

\[
\begin{align*}
{\text{list1, length1, list, length}} & \{ \\
& \}
\end{align*}
\]

\[
\begin{align*}
{\text{list2, length2, list, length}} & \{ \\
& \}
\end{align*}
\]

*from combine-0.3.3*
예

```c
void str_add_char (str* str, char c) {
    str->len++;
    if (str->len >= str->len){
        str->mem = str->len + 64;
        str->str = xrealloc(str->str, str->mem);
    }
    str->str[str->len - 1] = c;
}
```

```c
void f (){
    // str1->str1.size = str1->mem > str1->len
    str_add_char(str1, c1);
}
```

```c
void g (){
    // str2->str2.size = str2->mem > str2->len
    str_add_char(str2, c2);
}
```

*from spell-1.0*
예비 분석

• 모든 문맥과 모든 변수 관계를 고려하는 분석
• 값은 기존처럼 요약: 무한할 수도 / 반드시 유한
• 문제점: 기존방식으로는 예비분석조차 불가능
개별 분석 (modular analysis)

- 큰 프로그램을 각 부분 (주로 함수) 을 따로 분석 + 조합
개별 분석 (modular analysis)

• 큰 프로그램을 각 부분 (주로 함수) 을 따로 분석 + 조합
개별 분석 (modular analysis)

• 큰 프로그램을 각 부분 (주로 함수) 을 따로 분석 + 조합
개별 분석 (modular analysis)

- 큰 프로그램을 각 부분(주로 함수)을 따로 분석 + 조합
개별 분석 (modular analysis)

• 각 함수마다 입출력, 쿼리 정보를 요약해놓고 조립

```c
int inc(int i) { r = i + 1; return r; }

void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x); // Query 1
}

void g(int z){
    n = input();
    assert(n < z); // Query 2
}

void main(){
    a = inc(b);
    c = inc(d);
    f(a, b);
    f(c, d);
    g(a);
    g(c);
}
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Pre</th>
<th>Post</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>inc</td>
<td>T</td>
<td>i - r \geq \star \wedge r - i \leq \star</td>
<td>\emptyset</td>
</tr>
<tr>
<td>f</td>
<td>T</td>
<td>T</td>
<td>\emptyset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>y - x \leq \star \wedge n - x \leq \star \wedge \cdots</td>
<td>{1}</td>
</tr>
<tr>
<td>g</td>
<td>T</td>
<td>T</td>
<td>\emptyset</td>
</tr>
<tr>
<td>main</td>
<td>T</td>
<td>a - b \leq \star \wedge \cdots</td>
<td>{1}</td>
</tr>
</tbody>
</table>
후진 분석

• 각 쿼리를 선택하기 위한 충분 조건 유추 (뒤로)

```c
void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x);  // Query 1
}
```

```
<table>
<thead>
<tr>
<th>Query</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q: n - x ⊆ ★</td>
<td>y - x ⊆ ★</td>
</tr>
<tr>
<td>n - y ⊆ ★</td>
<td>y - x ⊆ ★</td>
</tr>
<tr>
<td>x = y</td>
<td>y = n + 64</td>
</tr>
<tr>
<td>n - x ⊆ ★</td>
<td>assume(n - y &lt;= -1)</td>
</tr>
<tr>
<td>n - x ⊆ ★</td>
<td>assume(y - n &lt;= 0)</td>
</tr>
<tr>
<td>n - x ⊆ ★</td>
<td>n = input()</td>
</tr>
</tbody>
</table>
```

Intuitively, the domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infinite (\(\infty\)) or may be \(\pm\) such that \(\pm v\). The abstract domain used in the pre-analysis only distinguishes the abstract bound is infi```
전진 분석

- 유추한 입력과 알수 없는 입력에 대한 출력을 유추 (앞으로)

```c
int inc(int i) { r = i + 1; return r; }

void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x); // Query 1
}
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>inc</td>
<td>⊤</td>
<td>i - r ⊑ ★ ∧ r - i ⊑ ★</td>
</tr>
<tr>
<td>f</td>
<td>⊤</td>
<td>⊤</td>
</tr>
<tr>
<td></td>
<td>y - x ⊑ ★</td>
<td>n - x ⊑ ★ ∧ ...</td>
</tr>
</tbody>
</table>
요약과 조립

• 각 함수의 요약은 함수호출문의 의미로 사용

<table>
<thead>
<tr>
<th>Function</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>inc</td>
<td>T</td>
<td>( i - r \sqsubseteq \star \land r - i \sqsubseteq \star )</td>
</tr>
</tbody>
</table>

\[
a = \text{inc}(b);
\]

\[
a - b \sqsubseteq \star \lor b - a \sqsubseteq \star
\]

\[
c = \text{inc}(d);
\]

\[
c - d \sqsubseteq \star \lor d - c \sqsubseteq \star
\]

\[
\ldots
\]
요약과 조립

- 각 입력이 만족되면 선택할 쿼리를 기록
- 최종 목표: 알수 없는 입력이 main에 들어온 경우 선택할 쿼리

```c
int inc(int i) { r = i + 1; return r; }

void f(int x, int y){
    n = input();
    if(n >= y){
        y = n + 64;
        x = y;
    }
    assert(n < x); // Query 1
}

void g(int z){
    n = input();
    assert(n < z); // Query 2
}

void main(){
    a = inc(b);
    c = inc(d);
    f(a, b);
    f(c, d);
    g(a);
    g(c);
}
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Pre</th>
<th>Post</th>
</tr>
</thead>
<tbody>
<tr>
<td>inc</td>
<td>$\top$</td>
<td>$i-r \sqsubseteq \bigstar \land r-i \sqsubseteq \bigstar$</td>
</tr>
<tr>
<td>f</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
<tr>
<td>g</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
<tr>
<td>main</td>
<td>$\top$</td>
<td>$a-b \sqsubseteq \bigstar \land \cdots$</td>
</tr>
</tbody>
</table>
필요한 관계와 문맥 유추

• 정확한 값 (★)을 만드는데 기여하는 변수와 문맥

```c
int inc(int i) {
    r = i + 1;
    return r;
}
```

```
18: a = inc(b);
```

```
\[
\begin{align*}
    r - i & \subseteq \star : \{(r, i), \emptyset\} \\
    i - r & \subseteq \star : \{(r, i), \emptyset\} \\
    a - b & \subseteq \star : \{(a, b, r, i), \{18\}\} \\
    b - a & \subseteq \star : \{(a, b, r, i), \{18\}\}
\end{align*}
\]
```
정리

- 예비 분석: 정확도 상승 기술 X 가 필요한 부분을 가늠

- 원칙: X 는 최대한 정확하게, 나머지는 과감히 요약

- 본 분석: 예비 분석의 결과에 따라 선별적으로 정확하게
기계 학습을 이용하여 선별적으로 정확하게 정적 분석
(Selective X-sensitive Analysis by Machine Learning)

- Learning a Variable-Clustering Strategy for Octagon from Labeled Data Generated by a Static Analysis, SAS’16
목표

• 많은 데이터를 공부하며 스스로 진화하는 분석기

• 데이터: 비슷한 코드, 이전 버전, 사용자 피드백, 버그 리포트, 테스트 결과 등

• 다른 분야에서는 이미 성숙: facebook, youtube, amazon, ...

Big Data + SourceForge  스파로우

Bugzilla
문제

• 예비 분석도 여전히 버거운 상황 (예. 관계 분석)

• 모든 관계를 추적하는 예비 분석: online estimator

• e.g.) 17 open source benchmarks (~100KLOC)
해결책

• 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습

• 모든 관계를 추적하는 예비 분석: offline teacher

• 33배 빠르고 정확도는 비슷

![Graph showing comparison between Var.Clustering and Main Analysis]
큰 그림

• 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습

Codebase  →  Static Analysis  →  Training Data (Var. relationship)  →  Machine Learning  →  Classifier

Target Program  →  Variable Clustering  →  Results (Var. Relationship)  →  Clusters
큰 그림

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습
Selectively checking constraints between variables only, our selective octagon algorithm tracks only these constraints, which are formally defined as pairs of constraints that hold together. For instance, in our example program, the pre-analysis of our example program helps improve the precision regarding given queries.

### Octagon Analysis

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>∞</td>
</tr>
<tr>
<td>c</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
<tr>
<td>i</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

$\oplus : \{(a,b), (a,i), (b,a) \ldots\}$

$\ominus : \{(a,c), (b,c), (c,a) \ldots\}$
The selective octagon analysis tracks octagon constraints only when doing so is likely to improve the precision that matters for each of selected queries, we do a dependency analysis to find out values are evaluated to result at line 4 is the table on the righthand side in (11).

For simplicity, we consider only constraints of the form $	ext{int } a = b$; $\text{int } c = \text{input}();$ // User input $\text{for } (i = 0; i < b; i++) \{$ $\text{assert } (i < a);$ // Query 1 $\text{assert } (i < c);$ // Query 2 $\}$

\begin{align*}
\gamma(\star) &= \mathbb{Z} \\
\gamma(\top) &= \mathbb{Z} \cup \{+\infty\}
\end{align*}

$\oplus : \{(a,b), (a,i), (b,a) \ldots\}$

$\otimes : \{(a,c), (b,c), (c,a) \ldots\}$
 큰 그림

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습
특질 (feature)

- 프로그램 P 안에 있는 변수 쌍 \((x, y)\) 에 관한 30가지 특질

(Please note: The text is in Korean, and the English text is for reference.)

(Positive situations for Octagon)
- \(x = y + k\) or \(y = x + k\)
- \(x \leq y + k\) or \(y \leq x + k\)
- \(x = \text{malloc}(y)\) or \(y = \text{malloc}(x)\)
- \(x[y]\) or \(y[x]\)
- ...

(Genral syntactic features)
- \(x\) or \(y\) is a field
- \(x\) and \(y\) represent sizes of arrays
- \(x\) or \(y\) is the size of a const string
- \(x\) or \(y\) is a global variable
- ...

(Negative situations for Octagon)
- \(x = cy\) or \(y = cx\) (\(c \neq 1\))
- \(x = yz\) or \(y = xz\)
- \(x = y/z\) or \(y = x/z\)
- ...

(General semantic features)
- \(x\) or \(y\) has a finite interval
- \(x\) or \(y\) is a local var in a recursive function
- \(x, y\) are not accessed in the same function
- ...

58
특질 (feature)

• 특질의 중요도 (Gini Index 로 측정)

• 부정 & 보편 > 긍정 & 특수

(Positive situations for Octagon)
- \( x = y + k \) or \( y = x + k \)
- \( x \leq y + k \) or \( y \leq x + k \)
- \( x = \text{malloc}(y) \) or \( y = \text{malloc}(x) \)
- \( x[y] \) or \( y[x] \)
- ...

(General syntactic features)
- \( x \) or \( y \) is a field
- \( x \) and \( y \) represent sizes of arrays
- \( x \) or \( y \) is the size of a const string
- \( x \) or \( y \) is a global variable
- ...

(Negative situations for Octagon)
- \( x = cy \) or \( y = cx \) (\( c \neq 1 \))
- \( x = yz \) or \( y = xz \)
- \( x = y/z \) or \( y = x/z \)
- ...

(General semantic features)
- \( x \) or \( y \) has a finite interval
- \( x \) or \( y \) is a local var in a recursive function
- \( x, y \) are not accessed in the same function
- ...

*Top 5 most important features
분류기 (Classifier)

• 변수 쌍 분류기 $C : Var \times Var \to \{\oplus, \ominus\}$ 학습

• 잘 알려진 ML 알고리즘 (decision tree)

• 선형 모델보다 훨씬 풍부한 표현력

• c.f.) logistic regression 으로 학습: 10~12x 느림
큰 그림

• 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습
변수 묶음 전략

- ⊕-표 변수쌍을 같은 묶음에

- transitive 관계도 자연스럽게 포함

```java
1  int a = b;
2  int c = input();  // User input
3  for (i = 0; i < b; i++) {
4      assert (i < a);  // Query 1
5      assert (i < c);  // Query 2
6  }
```

<table>
<thead>
<tr>
<th>(a,b)</th>
<th>⊕</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a,i)</td>
<td>⊗</td>
</tr>
<tr>
<td>(b,i)</td>
<td>⊕</td>
</tr>
<tr>
<td>(a,c)</td>
<td>⊗</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

변수 a, b, c, i에 대한 관계도와 C(x,y)의 표.
실험 결과

- Effectiveness (leave-one-out cross validation)

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>#Abs.Loc.</th>
<th># Alarms</th>
<th>Time(s)</th>
<th>#Alarms</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Itv</td>
<td>Impt</td>
<td>ML</td>
<td>Itv</td>
</tr>
<tr>
<td>brutefir</td>
<td>103</td>
<td>54</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>consol</td>
<td>298</td>
<td>165</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>id3</td>
<td>512</td>
<td>527</td>
<td>15</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>spell</td>
<td>2,213</td>
<td>450</td>
<td>20</td>
<td>8</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>mp3rename</td>
<td>2,466</td>
<td>332</td>
<td>33</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>irmp3</td>
<td>3,797</td>
<td>523</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>barcode</td>
<td>4,460</td>
<td>1,738</td>
<td>235</td>
<td>215</td>
<td>215</td>
<td>2</td>
</tr>
<tr>
<td>httptunnel</td>
<td>6,174</td>
<td>1,622</td>
<td>52</td>
<td>29</td>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>e2ps</td>
<td>6,222</td>
<td>1,437</td>
<td>119</td>
<td>58</td>
<td>58</td>
<td>3</td>
</tr>
<tr>
<td>bc</td>
<td>13,093</td>
<td>1,891</td>
<td>371</td>
<td>364</td>
<td>364</td>
<td>14</td>
</tr>
<tr>
<td>less</td>
<td>23,822</td>
<td>3,682</td>
<td>625</td>
<td>620</td>
<td>625</td>
<td>83</td>
</tr>
<tr>
<td>bison</td>
<td>56,361</td>
<td>14,610</td>
<td>1,988</td>
<td>1,955</td>
<td>1,955</td>
<td>137</td>
</tr>
<tr>
<td>pies</td>
<td>66,196</td>
<td>9,472</td>
<td>795</td>
<td>785</td>
<td>785</td>
<td>49</td>
</tr>
<tr>
<td>icecast-</td>
<td>68,564</td>
<td>6,183</td>
<td>239</td>
<td>232</td>
<td>232</td>
<td>51</td>
</tr>
<tr>
<td>raptor</td>
<td>76,378</td>
<td>8,889</td>
<td>2,156</td>
<td>2,148</td>
<td>2,148</td>
<td>242</td>
</tr>
<tr>
<td>dico</td>
<td>84,333</td>
<td>4,349</td>
<td>402</td>
<td>396</td>
<td>396</td>
<td>38</td>
</tr>
<tr>
<td>lsh</td>
<td>110,898</td>
<td>18,880</td>
<td>330</td>
<td>325</td>
<td>325</td>
<td>33</td>
</tr>
</tbody>
</table>

Total | 7,406 | 7,154 | 7,166 | 656 | 40,677 | 1,207 |
실험 결과

• Effectiveness (leave-one-out cross validation)

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>#Abs.Loc.</th>
<th># Alarms</th>
<th>Time(s)</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Itv</td>
<td>Impt</td>
<td>ML</td>
</tr>
<tr>
<td>brutefir</td>
<td>103</td>
<td>54</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>consol</td>
<td>298</td>
<td>165</td>
<td>20</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>id3</td>
<td>512</td>
<td>527</td>
<td>15</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>spell</td>
<td>2,213</td>
<td>450</td>
<td>20</td>
<td>8</td>
<td>17</td>
</tr>
<tr>
<td>mp3rename</td>
<td>2,466</td>
<td>332</td>
<td>33</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>irmp3</td>
<td>3,797</td>
<td>523</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>barcode</td>
<td>4,460</td>
<td>1,738</td>
<td>235</td>
<td>215</td>
<td>215</td>
</tr>
<tr>
<td>httptunnel</td>
<td>6,174</td>
<td>1,622</td>
<td>52</td>
<td>29</td>
<td>27</td>
</tr>
<tr>
<td>e2ps</td>
<td>6,222</td>
<td>1,437</td>
<td>119</td>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td>bc</td>
<td>13,093</td>
<td>1,891</td>
<td>371</td>
<td>364</td>
<td>364</td>
</tr>
<tr>
<td>less</td>
<td>23,822</td>
<td>3,682</td>
<td>625</td>
<td>620</td>
<td>625</td>
</tr>
<tr>
<td>bison</td>
<td>56,361</td>
<td>14,610</td>
<td>1,988</td>
<td>1,955</td>
<td>1,955</td>
</tr>
<tr>
<td>pies</td>
<td>66,196</td>
<td>9,472</td>
<td>795</td>
<td>785</td>
<td>785</td>
</tr>
<tr>
<td>icecast</td>
<td>68,564</td>
<td>6,183</td>
<td>239</td>
<td>232</td>
<td>232</td>
</tr>
<tr>
<td>raptor</td>
<td>76,378</td>
<td>8,889</td>
<td>2,156</td>
<td>2,148</td>
<td>2,148</td>
</tr>
<tr>
<td>dico</td>
<td>84,333</td>
<td>4,349</td>
<td>402</td>
<td>396</td>
<td>396</td>
</tr>
<tr>
<td>Ish</td>
<td>110,898</td>
<td>18,880</td>
<td>330</td>
<td>325</td>
<td>325</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>7,406</td>
<td>7,154</td>
<td>7,166</td>
</tr>
</tbody>
</table>

Effectiveness (leave-one-out cross validation):
-252 -240
실험 결과

• Effectiveness (leave-one-out cross validation)

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>#Abs.Loc.</th>
<th># Alarms</th>
<th># Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>brutefir</td>
<td>103</td>
<td>54</td>
<td>Itv</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>0</td>
</tr>
<tr>
<td>consol</td>
<td>298</td>
<td>165</td>
<td>Itv</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>0</td>
</tr>
<tr>
<td>id3</td>
<td>512</td>
<td>527</td>
<td>Itv</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>0</td>
</tr>
<tr>
<td>spell</td>
<td>2,213</td>
<td>450</td>
<td>Itv</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>0</td>
</tr>
<tr>
<td>mp3rename</td>
<td>2,466</td>
<td>332</td>
<td>Itv</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>0</td>
</tr>
<tr>
<td>irmp3</td>
<td>3,797</td>
<td>523</td>
<td>Itv</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>2</td>
</tr>
<tr>
<td>barcode</td>
<td>4,460</td>
<td>1,738</td>
<td>Itv</td>
<td>235</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>3</td>
</tr>
<tr>
<td>httptunnel</td>
<td>6,174</td>
<td>1,622</td>
<td>Itv</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>3</td>
</tr>
<tr>
<td>e2ps</td>
<td>6,222</td>
<td>1,437</td>
<td>Itv</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>3</td>
</tr>
<tr>
<td>bc</td>
<td>13,093</td>
<td>1,891</td>
<td>Itv</td>
<td>371</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>3</td>
</tr>
<tr>
<td>less</td>
<td>23,822</td>
<td>3,682</td>
<td>Itv</td>
<td>625</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>139</td>
</tr>
<tr>
<td>bison</td>
<td>56,361</td>
<td>14,610</td>
<td>Itv</td>
<td>1,988</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>51</td>
</tr>
<tr>
<td>pies</td>
<td>66,196</td>
<td>9,472</td>
<td>Itv</td>
<td>795</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>51</td>
</tr>
<tr>
<td>icecast-raptor</td>
<td>68,564</td>
<td>6,183</td>
<td>Itv</td>
<td>239</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>51</td>
</tr>
<tr>
<td>dico</td>
<td>76,378</td>
<td>8,889</td>
<td>Itv</td>
<td>2,156</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>242</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>38</td>
</tr>
<tr>
<td>lsh</td>
<td>84,333</td>
<td>4,349</td>
<td>Itv</td>
<td>302</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>33</td>
</tr>
<tr>
<td>Total</td>
<td>110,898</td>
<td>18,880</td>
<td>Itv</td>
<td>7,406</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Impt</td>
<td>656</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ML</td>
<td>-252</td>
</tr>
</tbody>
</table>
실험 결과

- Effectiveness (leave-one-out cross validation)

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>#Abs.Loc.</th>
<th># Alarms</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Itv</td>
<td>Impt</td>
</tr>
<tr>
<td>brutefir</td>
<td>103</td>
<td>54</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>consol</td>
<td>298</td>
<td>165</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>id3</td>
<td>512</td>
<td>527</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>spell</td>
<td>2,213</td>
<td>450</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>mp3rename</td>
<td>2,466</td>
<td>332</td>
<td>33</td>
<td>3</td>
</tr>
<tr>
<td>irmp3</td>
<td>3,797</td>
<td>523</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>barcode</td>
<td>4,460</td>
<td>1,738</td>
<td>235</td>
<td>215</td>
</tr>
<tr>
<td>httptunnel</td>
<td>6,174</td>
<td>1,622</td>
<td>52</td>
<td>29</td>
</tr>
<tr>
<td>e2ps</td>
<td>6,222</td>
<td>1,437</td>
<td>119</td>
<td>58</td>
</tr>
<tr>
<td>bc</td>
<td>13,093</td>
<td>1,891</td>
<td>371</td>
<td>364</td>
</tr>
<tr>
<td>less</td>
<td>23,822</td>
<td>3,682</td>
<td>625</td>
<td>620</td>
</tr>
<tr>
<td>bison</td>
<td>56,361</td>
<td>14,610</td>
<td>1,988</td>
<td>1,955</td>
</tr>
<tr>
<td>pies</td>
<td>66,196</td>
<td>9,472</td>
<td>795</td>
<td>785</td>
</tr>
<tr>
<td>icecast-raptor</td>
<td>68,564</td>
<td>6,183</td>
<td>239</td>
<td>232</td>
</tr>
<tr>
<td>dico</td>
<td>84,333</td>
<td>4,349</td>
<td>402</td>
<td>396</td>
</tr>
<tr>
<td>lsh</td>
<td>110,898</td>
<td>18,880</td>
<td>330</td>
<td>325</td>
</tr>
<tr>
<td>Total</td>
<td>7,406</td>
<td>7,154</td>
<td>7,166</td>
<td>656</td>
</tr>
</tbody>
</table>

66

\[
\begin{array}{c}
\times 62 \\
\times 2
\end{array}
\]
## 실험 결과

- Generalization: 작은 프로그램으로만 (<60KLOC) 학습

<table>
<thead>
<tr>
<th>Program</th>
<th>LOC</th>
<th>Abs. Loc.</th>
<th># Alarms</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Itv</td>
<td>All</td>
</tr>
<tr>
<td>pies</td>
<td>66,196</td>
<td>9,472</td>
<td>795</td>
<td>785</td>
</tr>
<tr>
<td>icecast-</td>
<td>68,564</td>
<td>6,183</td>
<td>239</td>
<td>232</td>
</tr>
<tr>
<td>raptor</td>
<td>76,378</td>
<td>8,889</td>
<td>2,156</td>
<td>2,148</td>
</tr>
<tr>
<td>dico</td>
<td>84,333</td>
<td>4,349</td>
<td>402</td>
<td>396</td>
</tr>
<tr>
<td>Ish</td>
<td>110,898</td>
<td>18,880</td>
<td>330</td>
<td>325</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>7,406</td>
<td>3,886</td>
</tr>
</tbody>
</table>

+4%
정리

• 선별적 관계 분석을 더욱 유연하게

• 기계학습 (학생) + 정적분석 (선생님)

• 예비분석만 이용하는 기술보다 33배 빠른 결과
기계 학습을 이용하여 선별적으로 안전하게 정적 분석
(Selectively Unsound Analysis by Machine Learning)

- Machine-Learning-Guided Selectively Unsound Static Analysis, ICSE’17
선별적으로 안전한 분석

• 안전성을 포기하는 전략을 조심스럽게 선별적으로

• 예) 순환문 해체, 명령어 무시

```plaintext
while(e){ C }  ▶ if(e){ C }      A;lib();B;  ▶ A;B;
```

![Diagram](image.png)
• (완전히) 안전한 버퍼 오버런 분석기

• interval domain + standard widening

```c
str = "hello world";
for(i=0; !str[i]; i++) // buffer access 1
    skip;

size = positive_input();
for(i=0; i<size; i++)
    skip;

... = str[i]; // buffer access 2
```
예제

- (완전히) 안전한 버퍼 오버런 분석기
- interval domain + standard widening

```c
str = "hello world";
for(i=0; !str[i]; i++) // buffer access 1
  skip;

size = positive_input();
for(i=0; i<size; i++)
  skip;

... = str[i]; // buffer access 2
```

Note that each loop is unrolled once and replaced with an if-statement. The analysis does not report a false alarm for buffer access 1, since the value of `size` is approximated as `[0, +\infty]` due to the unknown input value.

Therefore the analyzer reports an alarm for buffer access 2, which is a false alarm that we want to avoid. Meanwhile, the variable `i` in the first loop remains as `[0, +\infty]`, since this value is not refined by the loop condition `!str[i]`.

Inside the first loop, the analysis conservatively approximates the value of `str.size` as `[12, 12]`. Thus, the analysis reports an alarm for buffer access 2 while avoiding the false alarm for buffer access 1.

Note that we only unroll the first loop, not the second loop.
A. Uniformly Unsound Analysis

Consider a simple program in Figure 1. In the program, the string is initialized as "hello world". The first loop iterates over a constant string until the null value is reached. In the loop, buffer access 1 is always safe, since the index \(i\) is guaranteed to be smaller than the length of the string. However, buffer access 2 is not inside the loop. On the other hand, buffer access 2 is not always safe, since analyzing it soundly results in reporting a false alarm.

B. Uniformly Sound Analysis

For the uniformly sound analysis, the example program is treated as follows. The buffer access 2 while avoiding the false alarm for buffer access 1. By being unsound for the first loop and sound for the second loop, the uniformly sound analysis detects 33 bugs with 104 false alarms (FPR: 76%, FNR: 76%).

C. Selectively Unsound Analysis

Our selectively unsound analyzer applies unsoundness only where we control the soundness for both loops and library function calls. In the benchmarks with 138 bugs, the uniformly unsound analysis detects 33 bugs with 104 false alarms (FPR: 76%, FNR: 76%). The uniformly unsound analysis detects 96 bugs with 266 false alarms (FPR: 73%, FNR: 30%).

To summarize, our contributions are as follows:

- We present a machine-learning technique that can automatically tune a static analysis to be selectively unsound.
- We present a new approach of selectively employing interval analysis for buffer-overflow detection, which could be a potential cause of buffer overflow bugs in a program.

We illustrate our approach using a static analysis with the following steps:

1. The analysis conservatively approximates the value of \(i\) as \([0, size]\), where \(size\) is approximated as \([0, 11]\). However, it cannot capture non-convex properties (e.g. \(!str[i]\) due to the unknown input value.
2. Inside the first loop, the analysis conservatively approximates the index \(i\) in the second loop is upper bounded by \([0, 11]\), hence the analysis considers the value of \(i\) as \([0, 11]\), since this value is not refined by the analysis does not report a false alarm for buffer access 1, since the value of \(i\) remains as \([0, 11]\), where 11 is the null index of \(str\).

Note that each loop is unrolled once and replaced with an if-statement. The analysis does not report a false alarm for buffer access 1, since the value of \(i\) remains as \([0, 11]\), where 11 is the null index of \(str\).

... = str[i]; // buffer access 2
... = str[i]; // buffer access 2
A. Uniformly Unsound Analysis

Consider an analysis that is uniformly unsound for every loop. That is, all the loops in the given program are unrolled. By being unsound for the first loop and sound for the second loop, the analysis is able to report the true alarm for buffer access 2 while avoiding the false alarm for buffer access 1.

Note that we only unroll the first loop, not the second loop.

Inside the first loop, the analysis conservatively approximates the index i as the interval [0, 6]. However, it also fails to report a true alarm for buffer access 2; the value of i is approximated to [0, 0], hence the analysis considers the access 2 to be safe.

On the other hand, a sound interval analysis can detect the bug at buffer access 2 with a false alarm at buffer access 1. Therefore the analyzer reports an alarm for buffer access 2, since the value of i has the value of size after the first loop iterates over a constant string until the null value is found. In the loop, buffer access 1 is always safe, because the index i is guaranteed to be smaller than the length of the string. However, it cannot capture non-convex properties (e.g. i < size) in the loop condition.

For buffer access 2, the analysis conservatively approximates the index i as the interval [0, 6], which is a true alarm in this case. A sound interval analysis with loops in this section, which could be a potential cause of buffer overflow bugs. For simplicity, we only concern the buffer access to be analyzed soundly, since it has the possibility of causing an actual buffer overflow. The selectively unsound analysis also fails to report a true alarm for buffer access 2; the value of i is approximated as [0, 0]. Thus, the analysis reports an alarm for buffer access 1 as a potential buffer overflow error, which is a true alarm in this case.

B. Uniformly Sound Analysis

To summarize, our contributions are as follows:

• A uniform and selective unsound analysis for detecting buffer-overflow vulnerabilities.

• The uniformly sound analysis detects 118 bugs with 104 false alarms (FPR: 76%).

• The uniformly unsound analysis detects 33 bugs with 677 false alarms (FPR: 85%).

• Our selectively unsound analysis detects 96 bugs with 266 false alarms (FPR: 73%).

II. OVERVIEW

We present a new approach of selectively employing unsoundness in static analysis. All of the existing bug-finding static analyzers are uniformly unsound.

We demonstrate the effectiveness of the technique by automatically tuning a static analysis to be selectively unsound. We present a machine-learning technique that can automatically generate labelled data.

Our technique is based on anomaly detection with automatically tuning a static analysis to be selectively unsound.
Consider a simple program in Figure 1.

```
str = "hello world";
i = 0;
if(!str[i]) // buffer access 1
  skip;
size = positive_input();
for(i = 0; i < size; i++) // buffer access 1
  skip;
... = str[i]; // buffer access 2
```

### A. Uniformly Unsound Analysis

If we apply an unsound interval analysis to the program, we ignore the first loop, treating the example program as follows:

```
... = str[i]; // buffer access 2
```

This analysis is able to detect the true alarm for buffer access 2, which is a potential buffer overflow error. Meanwhile, buffer access 1 is approximated as safe, since the value of `i` is approximated to a lower bound of 0. However, it cannot capture non-convex properties (e.g. the loop condition `!str[i]`). Therefore the analyzer reports an alarm for buffer access 2, which is a false alarm.

### B. Uniformly Sound Analysis

A sound interval analysis can detect the actual buffer overflow. However, it also fails to report a true alarm for buffer access 2.

### C. Selectively Unsound Analysis

To summarize, our contributions are as follows:

- We present a new approach of selectively employing unsoundness in static analysis. All of the existing bug-finding static analyzers are uniformly unsound.
- We demonstrate the effectiveness of the technique by automatically tune a static analysis to be selectively unsound.
- We present a machine-learning technique that can automatically generate labelled data.
- Our technique is based on anomaly detection with automatic generation of labelled data.
- We present a selective unsound analysis technique that can improve the precision of static analysis while maintaining the original precision.
- We illustrate our approach using a static analysis with the example program in Figure 1, we ignore the first loop of the given program, which could be a potential cause of buffer overflow bugs.

By being unsound for the first loop and sound for the second loop, the analysis is able to report the true alarm for buffer access 2. The selectively unsound analysis detects 33 bugs with 104 false alarms (FPR: 76%, FNR: 13%). The second experiment is with loops in this section, which could be a potential cause of buffer access to be safe.

Note that each loop is unrolled once and replaced with an if-statement. The analysis does not report a false alarm for buffer access 1, since the value of `i` remains as approximated to a lower bound of 0. However, it cannot capture non-convex properties (e.g. the loop condition `!str[i]`). Therefore the analyzer reports an alarm for buffer access 2, which is a false alarm that we want to avoid. Meanwhile, the variable `i` is approximated as

```
i = 0;
```

In the second loop, on the other hand, a sound interval analysis can detect the actual buffer overflow. The selectively unsound analysis detects 96 bugs with 266 false alarms (FPR: 73%, FNR: 30%).
목표

\[ F \in Pgm \times \Pi \rightarrow A \]

- 선별적으로 안전성을 포기하는 전략 \( \pi \in \Pi \) 찾기
- 안전하게 분석할 순환문 (\( \Pi = 2^{Loop} \))
- 안전하게 분석할 라이브러리 호출 (\( \Pi = 2^{Lib} \))
- \( p \) 에만 선별적으로 안전성을 포기하는 전략 적용 (\( p \notin \pi \))
- 예) \( \text{while}(e)\{ \ C \ \} \quad \triangleright \quad \text{if}(e)\{ \ C \ \} \quad \)

\[ A; \text{lib}(); B; \quad \triangleright \quad A;B; \]
In this section, we explain the details of our technique. Our approach consists of two main phases: training and testing. The training phase involves the following steps:

1. **Codebase**: We start with an existing codebase to generate training data.
2. **Training Data Generation**: From the codebase, we generate training data that includes various program components and their corresponding soundness parameters.
3. **Machine Learning**: We use machine learning algorithms to learn from the training data. These algorithms are designed to infer soundness parameters that effectively detect real bugs while minimizing false alarms.

For the testing phase:

1. **Test Program**: We apply the learned soundness parameters to a new, unseen program.
2. **Inferring Harmless Unsoundness**: The machine learning algorithm infers the new harmless unsoundness for the test program.

Our method is designed to be effective yet harmless, ensuring that we detect real bugs with minimal false alarms. It is particularly useful when there is no confusion between sound and unsound analyses.
학습 데이터 생성

- 준비물: 버그 위치가 알려져 있는 프로그램
- 안전성을 포기할 시 정확도는 상승하지만 버그는 놓치지 않는 부품
- 예) 순환문, 라이브러리 호출

**Algorithm 1** Training data generation

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>$T := \emptyset$</td>
</tr>
<tr>
<td>2:</td>
<td>for all $(P_i, B_i) \in P$ do</td>
</tr>
<tr>
<td>3:</td>
<td>$A_i = F(P_i, 1)$</td>
</tr>
<tr>
<td>4:</td>
<td>$(A_t, A_f) := (A_i \cap B_i, A_i \setminus B_i)$</td>
</tr>
<tr>
<td>5:</td>
<td>for all $j \in \mathbb{J}_{P_i}$ do</td>
</tr>
<tr>
<td>6:</td>
<td>$A_i' = F(P_i, 1 \setminus {j})$</td>
</tr>
<tr>
<td>7:</td>
<td>$(A_t', A_f') = (A_i' \cap B_i, A_i' \setminus B_i)$</td>
</tr>
<tr>
<td>8:</td>
<td>if $</td>
</tr>
<tr>
<td>9:</td>
<td>$T := T \cup {f(j)}$</td>
</tr>
<tr>
<td>10:</td>
<td>end if</td>
</tr>
<tr>
<td>11:</td>
<td>end for</td>
</tr>
<tr>
<td>12:</td>
<td>end for</td>
</tr>
</tbody>
</table>

Suppose that we have $n$ binary and numeric features.

For each program component, i.e., unsoundly analyzing them does not miss bugs but improves the precision. Our system in the next section, we present the feature sets for our instance learning, where a classifier is learnt from a set of training examples. Since we use the One-Class SVM, the training data generation results than the fully sound analysis.

In the codebase, and decides the components as precision-effective yet inferred library calls are ignored like no-op. Hence often unsoundly handled in real-world analyzers. Instead, we targets to the individual elements. Instead, we infer precision-effective yet inferred loops are unrolled only once, hence regarded as conditional statements, meanwhile the inferred loops are unrolled only once, hence often unsoundly handled in real-world analyzers.

With the classifier, we define the model (1) as follows:  

$$M = \{ \text{harmless program components} \}$$

Based analysis for building models.

We automatically generate the feature vectors that describe precision-effective yet inferred loops are unrolled only once, hence often unsoundly handled in real-world analyzers.

3.3 Learning Precision-Effective Unsoundness

In this section, we show two instances of our system on top of various applications especially when the one common class is anomalous behavior detection. In our case, it is used to learn a single class of elements and derive an classifier that, given predicts precision-effective yet inferred loops are unrolled only once, hence often unsoundly handled in real-world analyzers.

For presentation brevity, In experiments, we use both binary and numeric features. For each program component, the static analysis for a new, unseen program component.

In all of the programs, the algorithm runs the static analysis (line 6). There might be combinatorial effects of each analysis, and library calls. The inferred loops are unrolled only once, hence often unsoundly handled in real-world analyzers.

The algorithm has linear time complexity to the number of target components in the codebase. For each component, we collect the program component.

That is, we first determine the soundness parameter ($M$) and instantiate the static analysis with ($M$).

Testing.
학습 데이터 생성

- 준비물: 버그 위치가 알려져 있는 프로그램
- 안전성을 포기할 시 정확도는 상승하지만 버그는 놓치지 않는 부품
- 예) 순환문, 라이브러리 호출

Algorithm 1 Training data generation

1: \( T := \emptyset \)
2: for all \( (P_i, B_i) \in P \) do
3: \( A_i = F(P_i, B) \)
4: \( (A_t, A_f) := (A_i \cap B_i, A_i \setminus B_i) \)
5: for all \( j \in \mathbb{J}_{P_i} \) do
6: \( A'_i = F(P_i, 1 \setminus \{j\}) \)
7: \( (A'_t, A'_f) = (A'_i \cap B_i, A'_i \setminus B_i) \)
8: if \( |A'_i| = |A_t| \land |A'_f| < |A_f| \) then
9: \( T := T \cup \{f(j)\} \)
10: end if
11: end for
12: end for

(프로그램, 버그 위치)
학습 데이터 생성

- 준비물 : 버그 위치가 알려져 있는 프로그램
- 안전성을 포기할 시 정확도는 상승하지만 버그는 놓치지 않는 부품
- 예) 순환문, 라이브러리 호출

Algorithm 1 Training data generation

1: $T := \emptyset$
2: for all $(P_i, B_i) \in P$ do
3:   $A_i = F(P_i, 1)$
4:   $(A_t, A_f) = (A_i \cap B_i, A_i \setminus B_i)$
5:   for all $j \in P_i$ do
6:     $A'_i = F(P_i, 1 \setminus \{j\})$
7:     $(A'_t, A'_f) = (A'_i \cap B_i, A'_i \setminus B_i)$
8:     if $|A'_t| = |A_t| \land |A'_f| < |A_f|$ then
9:       $T := T \cup \{f(j)\}$
10:    end if
11:  end for
12: end for
학습 데이터 생성

- 준비물: 버그 위치가 알려져 있는 프로그램
- 안전성을 포기할 시 정확도는 상승하지만 버그는 놓치지 않는 부품
- 예) 순환문, 라이브러리 호출

Algorithm 1 Training data generation

1: $T := \emptyset$
2: for all $(P_i, B_i) \in P$ do
3:    $A_i = F(P_i, 1)$
4:    $(A_t, A_f) := (A_i \cap B_i, A_i \setminus B_i)$
5:    for all $j \in \mathbb{J}_{P_i}$ do
6:        $A'_i = F(P_i, 1 \setminus \{j\})$
7:        $(A'_t, A'_f) := (A_i \cap B_i, A'_i \setminus B_i)$
8:        if $|A'_i| = |A_t| \land |A'_f| < |A_f|$ then
9:            $T := T \cup \{f(j)\}$
10:        end if
11:    end for
12: end for
학습 데이터 생성

• 준비물: 버그 위치가 알려져 있는 프로그램
• 안전성을 포기할 시 정확도는 상승하지만 버그는 놓치지 않는 부품
• 예) 순환문, 라이브러리 호출

<table>
<thead>
<tr>
<th>Algorithm 1 Training data generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( T := \emptyset )</td>
</tr>
<tr>
<td>2: for all ( (P_i, B_i) \in P ) do</td>
</tr>
<tr>
<td>3: ( A_i = F(P_i, 1) )</td>
</tr>
<tr>
<td>4: ( (A_t, A_f) := (A_i \cap B_i, A_i \setminus B_i) )</td>
</tr>
<tr>
<td>5: for all ( j \in J_{P_i} ) do</td>
</tr>
<tr>
<td>6: ( A'_i = F(P_i, 1 \setminus {j}) )</td>
</tr>
<tr>
<td>7: ( (A'_t, A'_f) = (A'_i \cap B_i, A'_i \setminus B_i) )</td>
</tr>
<tr>
<td>8: if (</td>
</tr>
<tr>
<td>9: ( T := T \cup {f(j)} )</td>
</tr>
<tr>
<td>10: end if</td>
</tr>
<tr>
<td>11: end for</td>
</tr>
<tr>
<td>12: end for</td>
</tr>
</tbody>
</table>

거짓 경보는 감소하고 버그는 여전히 다 찾는다면 학습 데이터 (“무해한” 부품)
무해한 부품

• 거짓 경보를 유발하는 부품: 전형적, 풍부

• 프로그래밍 패턴, 분석기 도메인에 따른 특징

• 거짓 경보 >> 실제 경보

• 예)

```c
str = "hello world";
for(i=0; !str[i]; i++) // buffer access 1
    skip;

size = positive_input();
for(i=0; i<size; i++)
    skip;

... = str[i]; // buffer access 2
```
기계 학습

• 데이터의 전형적인 특징을 찾는데 특화된 알고리즘 사용

• 대표적으로 one-class SVM
특질

• 22 가지 순환문 특질, 15 가지 라이브러리 호출 특질
• 간단한 구문 + 의미 분석 결과 (숫자, 포인터, 문자열)

<table>
<thead>
<tr>
<th>Target</th>
<th>Feature</th>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop</td>
<td>Null</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains nulls or not</td>
</tr>
<tr>
<td></td>
<td>Const</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains constants or not</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains array accesses or not</td>
</tr>
<tr>
<td></td>
<td>Conjunction</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains &amp;&amp; or not</td>
</tr>
<tr>
<td></td>
<td>IdxSingle</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains an index for a single array in the loop</td>
</tr>
<tr>
<td></td>
<td>IdxMulti</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains an index for multiple arrays in the loop</td>
</tr>
<tr>
<td></td>
<td>IdxOutside</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether the loop condition contains an index for an array outside of the loop</td>
</tr>
<tr>
<td></td>
<td>InitIdx</td>
<td>Syntactic</td>
<td>Binary</td>
<td>Whether an index is initialized before the loop</td>
</tr>
<tr>
<td></td>
<td>Exit</td>
<td>Syntactic</td>
<td>Numeric</td>
<td>The (normalized) number of exits in the loop</td>
</tr>
<tr>
<td></td>
<td>Size</td>
<td>Syntactic</td>
<td>Numeric</td>
<td>The (normalized) size of the loop</td>
</tr>
<tr>
<td></td>
<td>ArrayAccess</td>
<td>Syntactic</td>
<td>Numeric</td>
<td>The (normalized) number of array accesses in the loop</td>
</tr>
<tr>
<td></td>
<td>ArithInc</td>
<td>Syntactic</td>
<td>Numeric</td>
<td>The (normalized) number of arithmetic increments in the loop</td>
</tr>
<tr>
<td></td>
<td>PointerInc</td>
<td>Syntactic</td>
<td>Numeric</td>
<td>The (normalized) number of pointer increments in the loop</td>
</tr>
<tr>
<td></td>
<td>Prune</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether the loop condition prunes the abstract state or not</td>
</tr>
<tr>
<td></td>
<td>Input</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether the loop condition is determined by external inputs</td>
</tr>
<tr>
<td></td>
<td>GVar</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether global variables are accessed in the loop condition</td>
</tr>
<tr>
<td></td>
<td>FinInterval</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether a variable has a finite interval value in the loop condition</td>
</tr>
<tr>
<td></td>
<td>FinArray</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether a variable has a finite size of array in the loop condition</td>
</tr>
<tr>
<td></td>
<td>FinString</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether a variable has a finite string in the loop condition</td>
</tr>
<tr>
<td></td>
<td>LSize</td>
<td>Semantic</td>
<td>Binary</td>
<td>Whether a variable has an array of which the size is a left-closed interval</td>
</tr>
<tr>
<td></td>
<td>LCOffset</td>
<td>Semantic</td>
<td>Numeric</td>
<td>The (normalized) number of abstract locations accessed in the loop</td>
</tr>
<tr>
<td></td>
<td>#AbsLoc</td>
<td>Semantic</td>
<td>Numeric</td>
<td>The (normalized) number of abstract locations accessed in the arguments</td>
</tr>
<tr>
<td></td>
<td>#ArgString</td>
<td>Semantic</td>
<td>Numeric</td>
<td>The (normalized) number of string arguments</td>
</tr>
</tbody>
</table>

| Library | Const     | Syntactic | Binary | Whether the parameters contain constants or not |
|         | Void      | Syntactic | Binary | Whether the return type is void or not |
|         | Int       | Syntactic | Binary | Whether the return type is int or not |
|         | CString   | Syntactic | Binary | Whether the function is declared in string.h or not |
|         | InsideLoop | Syntactic | Binary | Whether the function is called in a loop or not |
|         | #Args     | Syntactic | Numeric | The (normalized) number of arguments |
|         | DefParam  | Semantic | Binary | Whether a parameter are defined in a loop or not |
|         | UseRet    | Semantic | Binary | Whether the return value is used in a loop or not |
|         | UptParam  | Semantic | Binary | Whether a parameter is update via the library call |
|         | Escape    | Semantic | Binary | Whether the return value escapes the caller |
|         | GVar      | Semantic | Binary | Whether a parameters points to a global variable |
|         | Input     | Semantic | Binary | Whether a parameters are determined by external inputs |
|         | FinInterval | Semantic | Binary | Whether a parameter have a finite interval value |
|         | #AbsLoc   | Semantic | Numeric | The (normalized) number of abstract locations accessed in the arguments |
|         | #ArgString | Semantic | Numeric | The (normalized) number of string arguments |
실험 결과

• 23개 프로그램, 버퍼오버런 분석

• 대상: 순환문, 라이브러리 호출

![차트1](chart1.png)

![차트2](chart2.png)
실험 결과

• 13 개 프로그램, 포맷 스트링 분석

• 대상: 라이브러리 호출

#True Alarm

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Selective</th>
<th>Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

#False Alarm

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Selective</th>
<th>Uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>75</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
후속 연구

· 기계 학습을 위한 특질 자동 생성

· 아이디어: 해당 분석의 핵심을 드러내는 (작은) 프로그램

```c
/* real code */
pos = 0;
path = malloc(temp);
path_len = tmp;
file = str_make(0);
if (*){
    while(pos < path_len){
        path[pos] = 1;
        str_add_char(path,pos);
        str_buf = nstr_to_str(path);
        pos++;
    }
}
```

```c
/* example */
arr = malloc(size);
i = 0;
while(i < size){
    arr[i] = 0;
    i++;
}
```
후속 연구

- 제한시간동안 최대한 선별적으로 정확하게 분석

- 대부분은 “싹수”가 노란 알람 (초반에 발생해서 끝까지)

- 아이디어: 분석중 알람이 난 부분은 그때그때 과감히 요약
결론

• 안전, 정확, 빠른 정적 분석의 핵심: 선별적

• 예비 분석을 이용하여 선별적으로 정확하게 [PLDI’14, TOPLAS’16]

• 기계 학습을 이용하여 선별적으로 정확하게 [SAS’16]

• 기계 학습을 이용하여 선별적으로 안전하게 [ICSE’17]
결론

- 안전, 정확, 빠른 정적 분석의 핵심: 선별적

  - 예비 분석을 이용하여 선별적으로 정확하게 [PLDI’14, TOPLAS’16]
  - 기계 학습을 이용하여 선별적으로 정확하게 [SAS’16]
  - 기계 학습을 이용하여 선별적으로 안전하게 [ICSE’17]