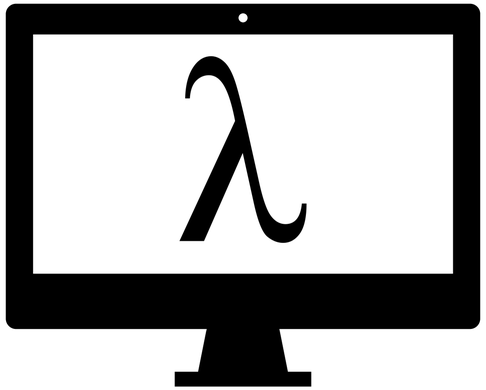
프로그래밍 언어와 기계 학습의 만남

허 기 홍
전산학부 / 프로그래밍 시스템 연구실
제 2회 KAIST PL 워크샵

KAIST

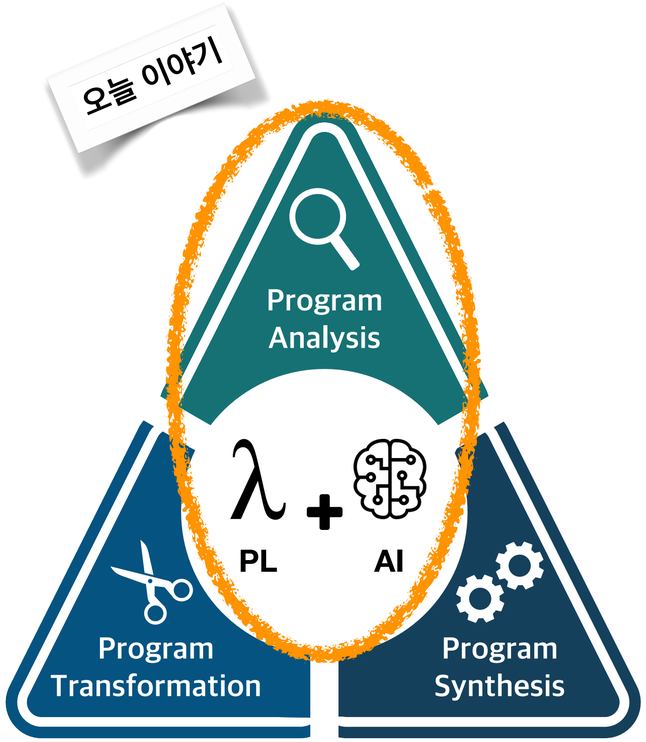
Programming Systems Laboratory

목표

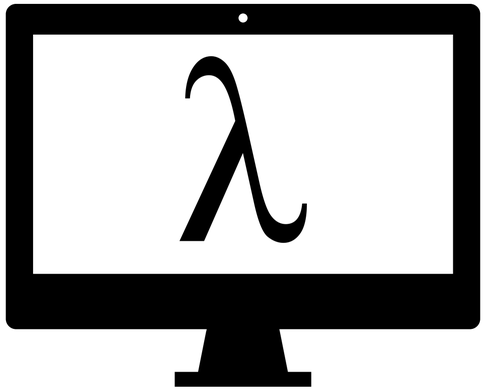


Next-generation
Programming Systems

목표

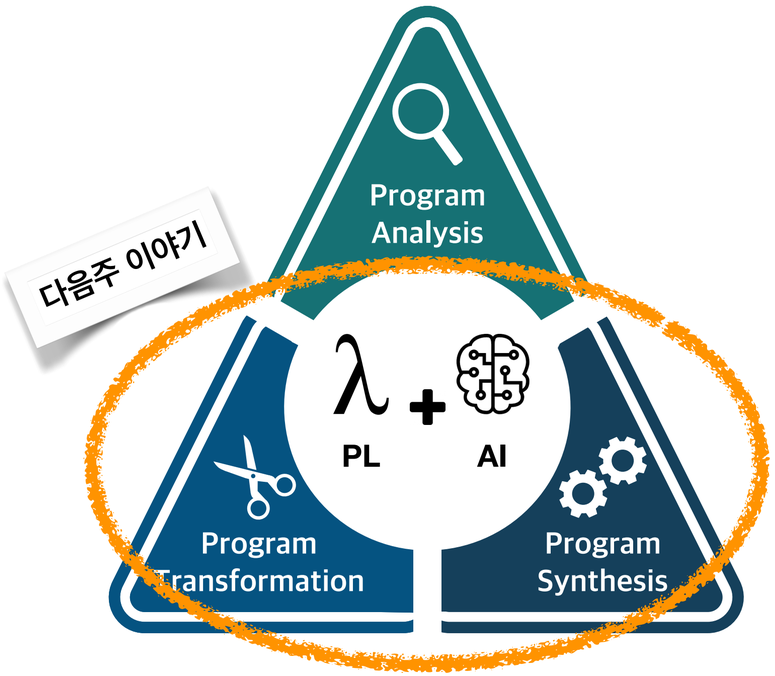


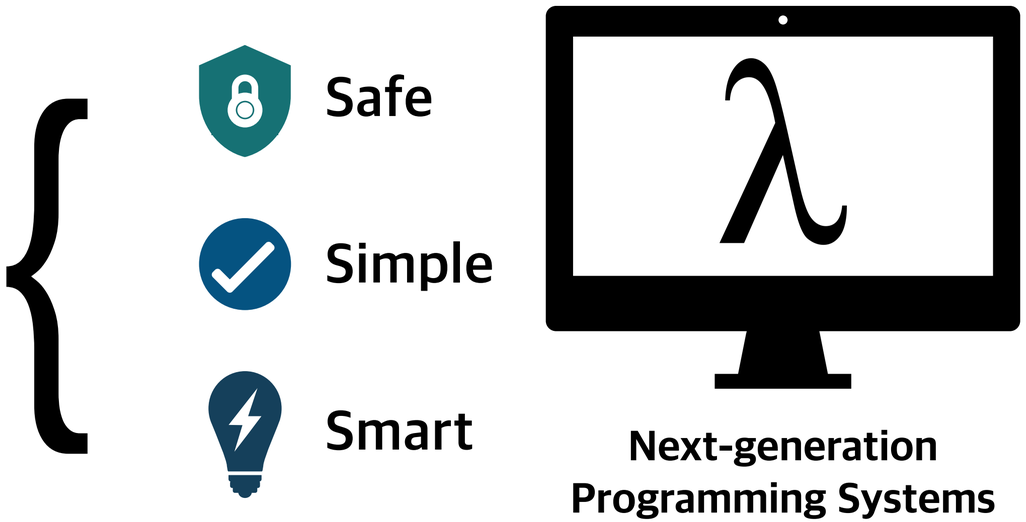
- Safe
- Simple
- Smart



Next-generation Programming Systems

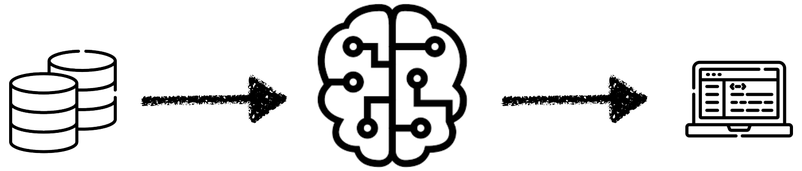
목표


다음주 이야기
Program Analysis
λ +
PL
AI
Program Transformation
Program Synthesis


Safe
Simple
Smart
λ
Next-generation Programming Systems

PL vs ML

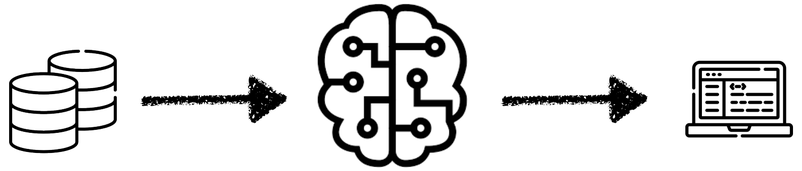
- 논리 기반
- 엄밀함 (문법, 실행 의미, 타입 등)
- 증명 가능, 해석 가능
- 올바른 답을 내지만 오래 걸릴 수도
- 못 푸는 문제도 있다 (예: 결정불가능)
- 사칙 연산, 논리 증명 등 잘함 (직렬순차)



- 확률 기반
- 유연함 (모든 것이 숫자, 벡터)
- 증명과 해석이 어려움
- 그럴듯한 답을 빨리 내지만 틀릴 수도
- 못 푸는 문제는 없다 (정확도가 낮을 뿐)
- 그림 맞추기, 번역하기 등 잘함 (병렬동시)

PL + ML?

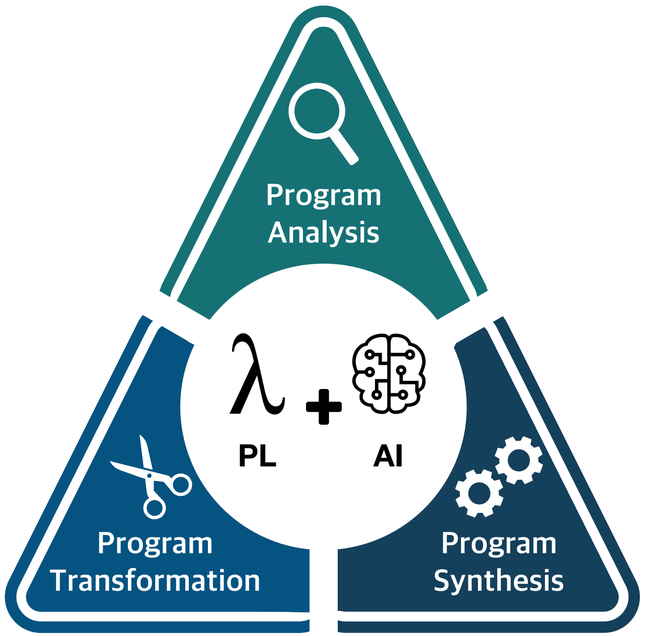
- 논리 기반
- 엄밀함 (문법, 실행 의미, 타입 등)
- 증명 가능, 해석 가능
- 올바른 답을 내지만 오래 걸릴 수도
- 못 푸는 문제도 있다 (예: 결정불가능)
- 사칙 연산, 논리 증명 등 잘함 (직렬순차)



- 확률 기반
- 유연함 (모든 것이 숫자, 벡터)
- 증명과 해석이 어려움
- 그럴듯한 답을 빨리 내지만 틀릴 수도
- 못 푸는 문제는 없다 (정확도가 낮을 뿐)
- 그림 맞추기, 번역하기 등 잘함 (병렬동시)

몇 가지 경험과 사례

- 프로그램 분석
  - 유연한 프로그램 분석을 위한 지도 학습, 강화 학습
  - 편리한 프로그램 분석을 위한 베이지안 추론
- 프로그램 변환과 합성
  - 빠른 합성을 위한 언어 모델
  - 빠른 변환을 위한 강화 학습

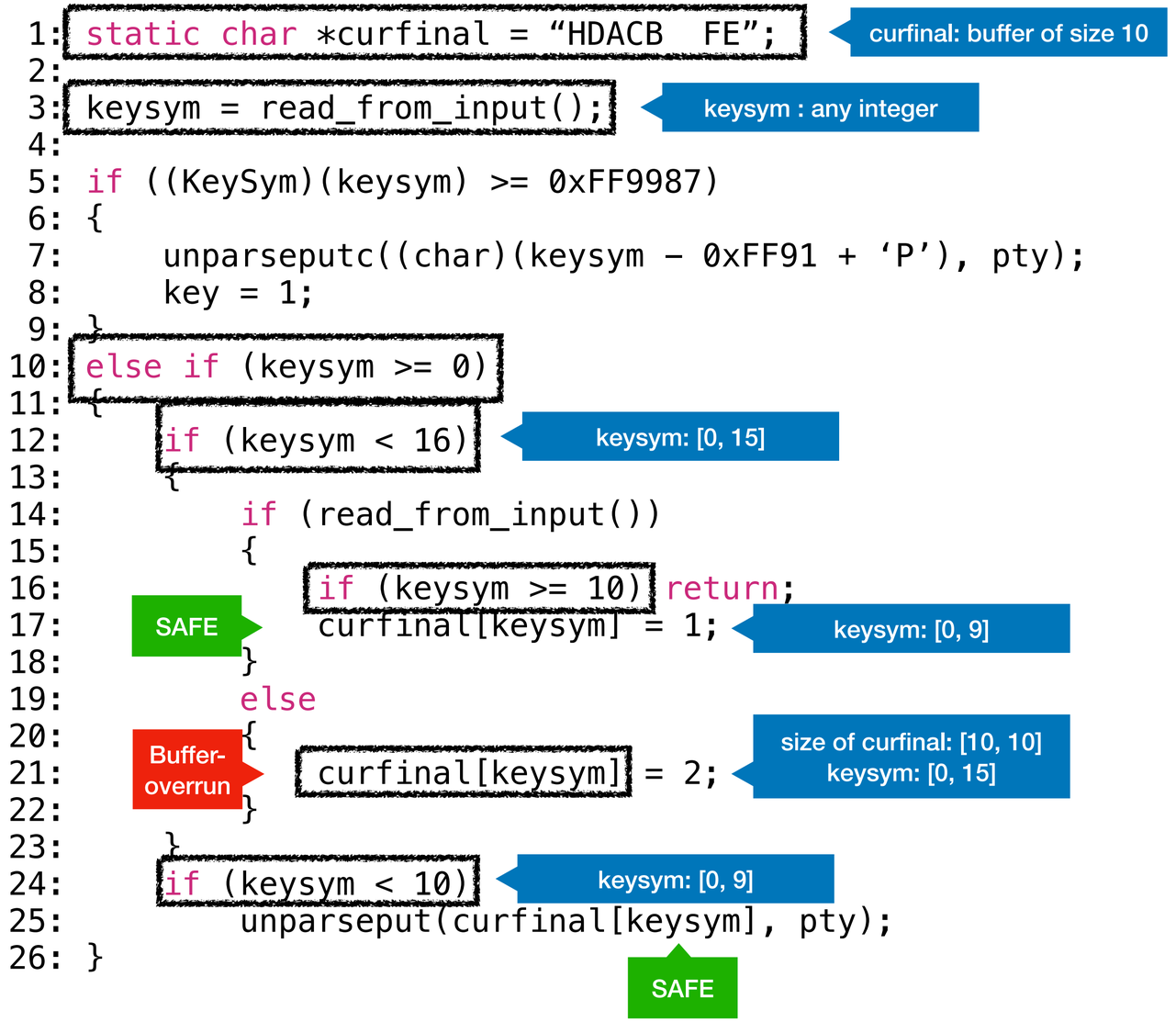


프로그램 정적 분석

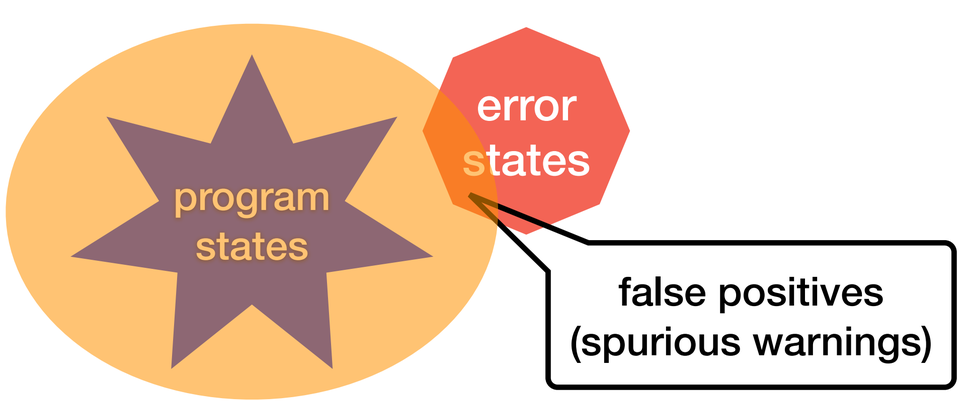
“SW MRI”

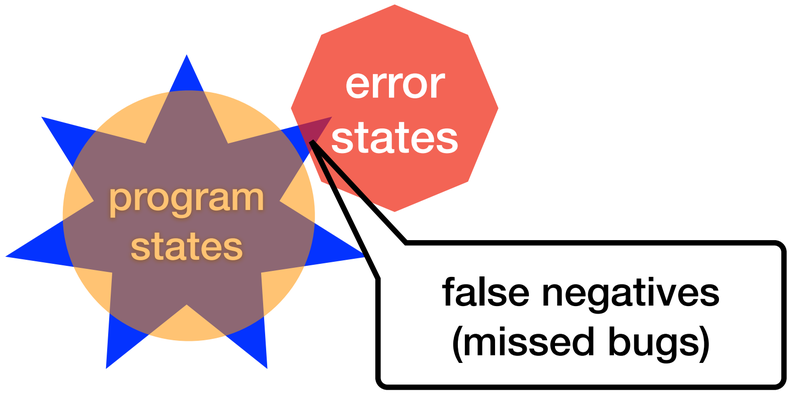
- SW 의 동작을 자동으로 예측하는 일반적이고 체계적인 방법
  - 예측: 실행하기 전에 미리 (정적)
  - 자동: 소프트웨어를 분석하는 소프트웨어 (“분석기”)
  - 일반적: 언어와 성질에 국한되지 않음
  - 체계적 : 요약 해석 (abstract interpretation) 이라는 이론에 기반
- 응용: 오류 검출, 검증, 보안, 코드 유지보수, 코드 최적화 등

예제


1: static char *curfinal = “HDACB FE”;
curfinal: buffer of size 10
2:
3: keysym = read_from_input();
keysym : any integer
4:
5: if ((KeySym)(keysym) >= 0xFF9987)
6: {
7: unparseputc((char)(keysym – 0xFF91 + ‘P’), pty);
8: key = 1;
9: }
10: else if (keysym >= 0)
11: {
12: if (keysym < 16)
keysym: [0, 15]
13: {
14: if (read_from_input())
15: {
16: if (keysym >= 10) return;
SAFE
17: curfinal[keysym] = 1;
keysym: [0, 9]
18: }
19: else
20: {
Buffer-overrun
21: curfinal[keysym] = 2;
size of curfinal: [10, 10]
keysym: [0, 15]
22: }
23: }
24: if (keysym < 10)
keysym: [0, 9]
25: unparseput(curfinal[keysym], pty);
26: }
SAFE

프로그램 분석의 고질적 문제



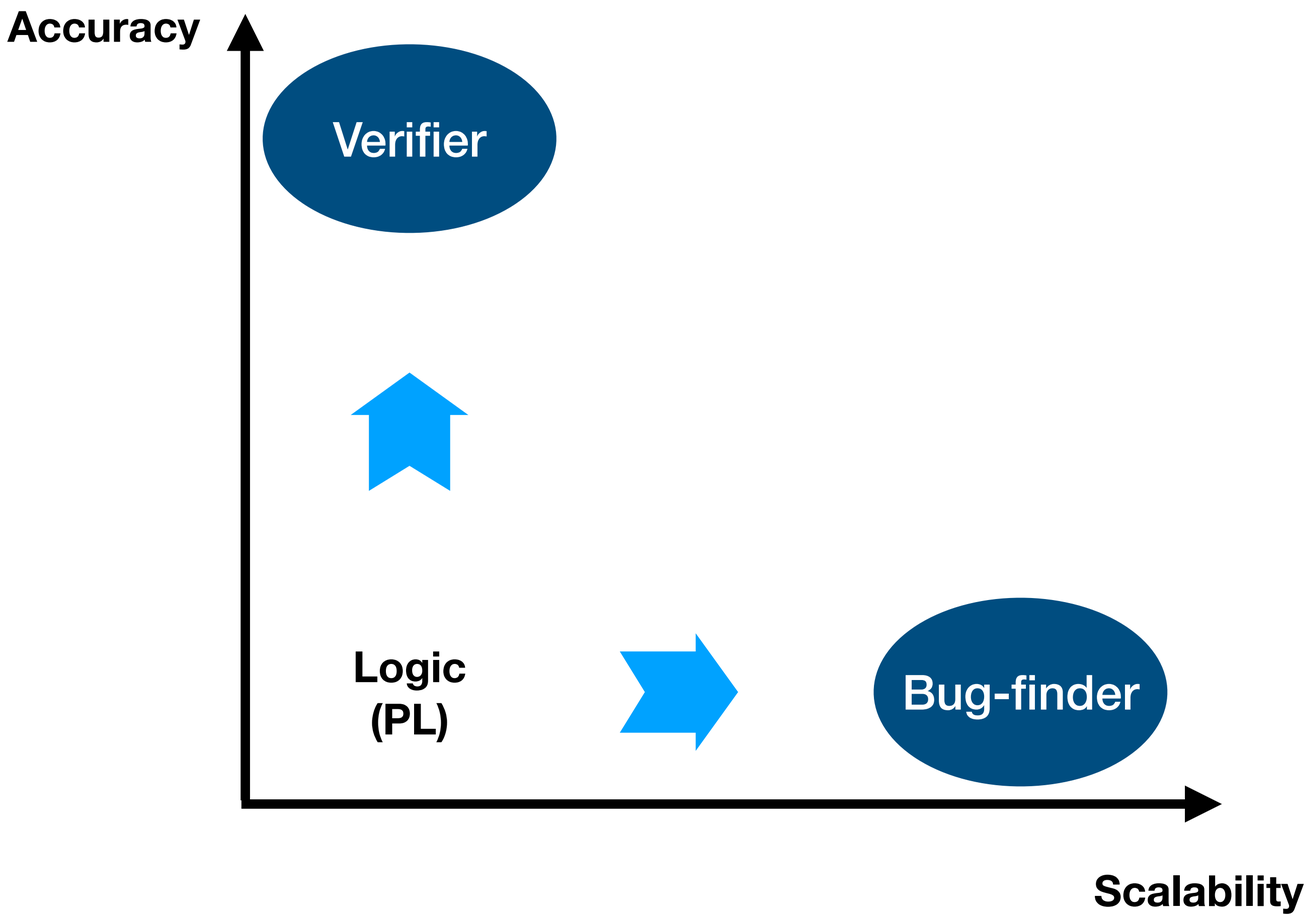


OpenSSL
Cryptography and SSL/TLS Toolkit

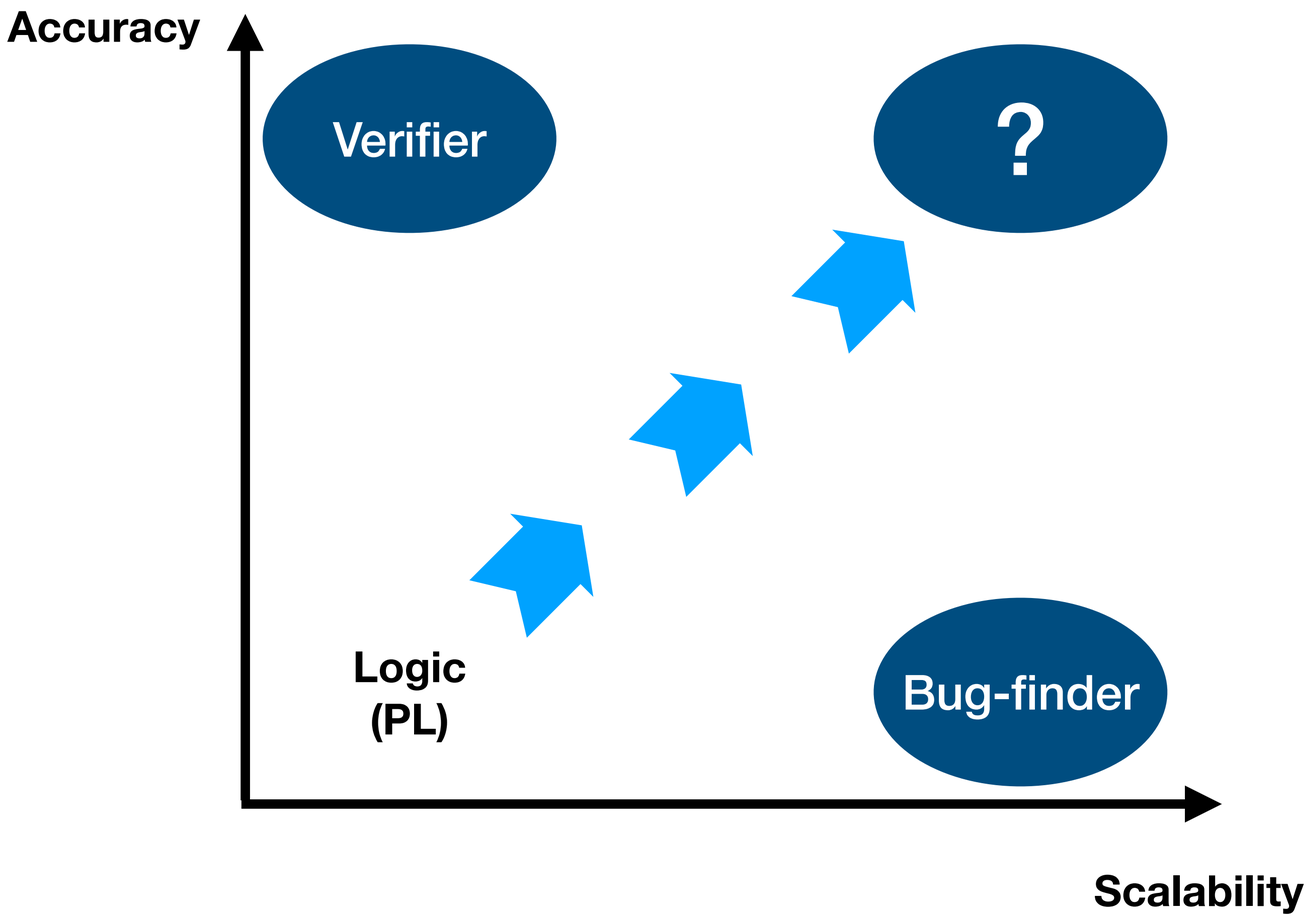
"… can be difficult to do without introducing large numbers of ***false positives***, or scaling ***performance*** exponentially poorly. In this case, ***balancing*** these … caused us to ***miss the defect***."

— *On Detecting Heartbleed with Static Analysis, (Coverity, 2014)*

전통적인 프로그램 분석



우리의 목표



전통적인 프로그램 분석의 문제점

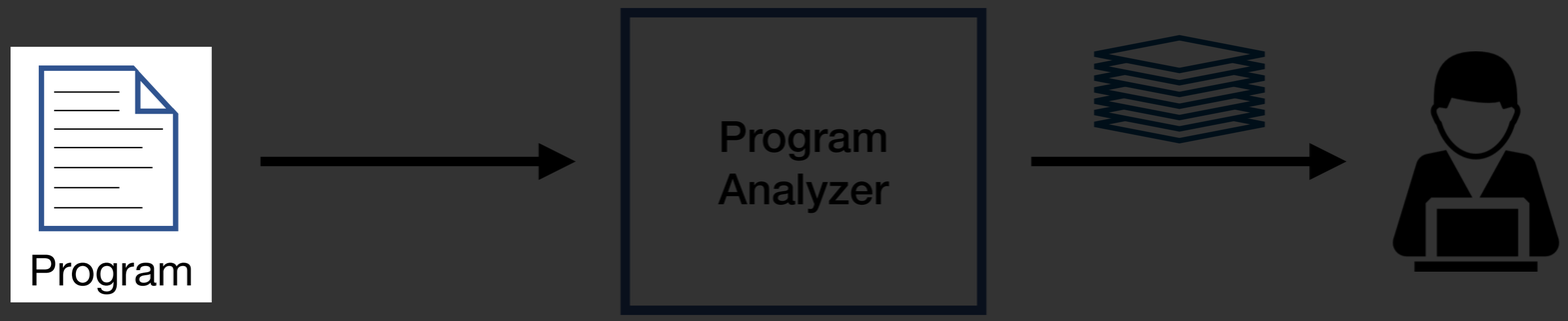
전통적인 프로그램 분석의 문제점

1. Inflexible

전통적인 프로그램 분석의 문제점

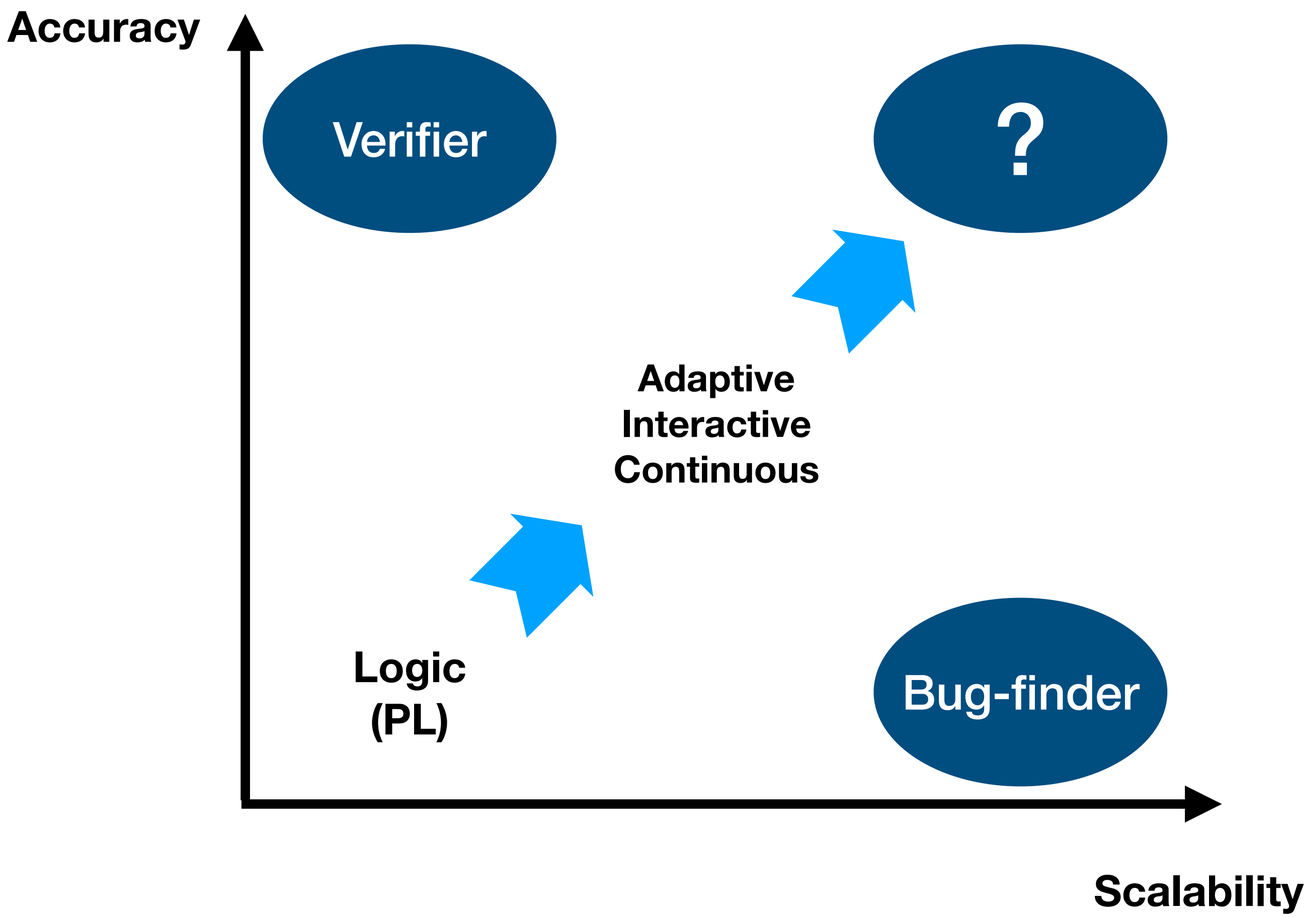
2. Unidirectional

전통적인 프로그램 분석의 문제점

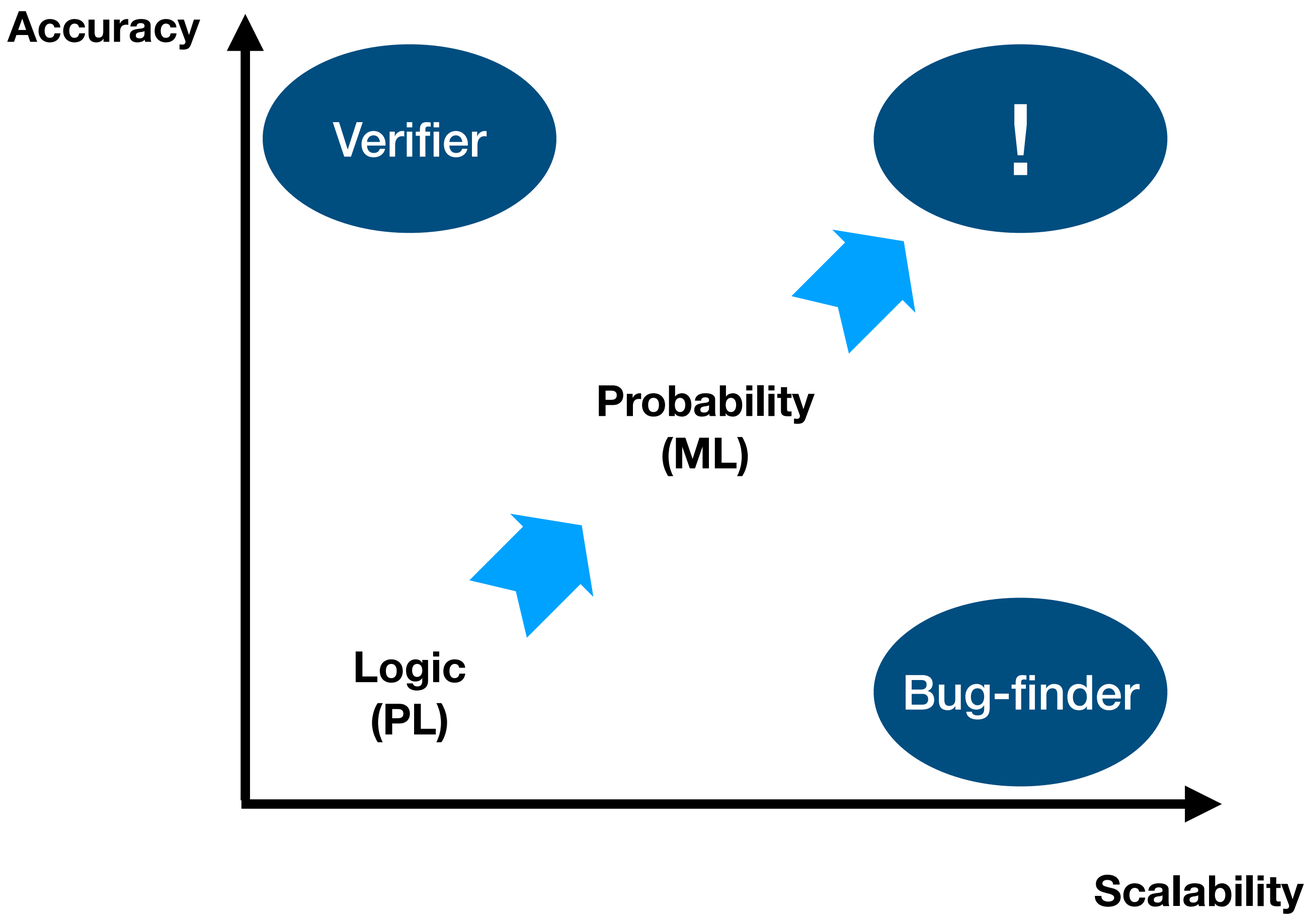


3. Narrow-sighted

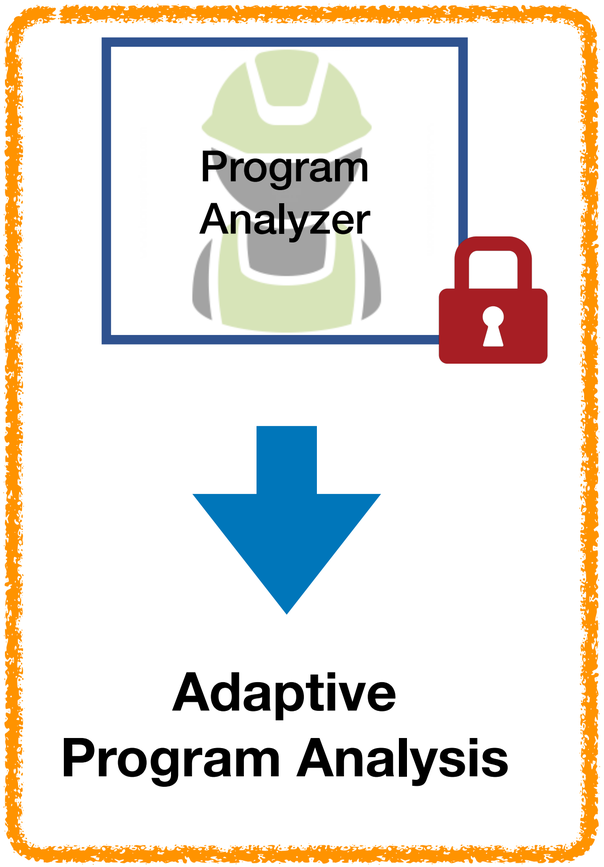
우리의 목표


Accuracy
Verifier
?
Adaptive
Interactive
Continuous
Logic
(PL)
Bug-finder
Scalability

ML 을 이용한 정적 분석

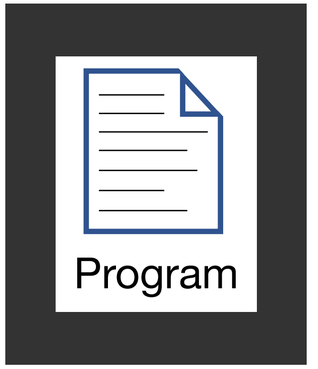

Accuracy
Verifier
!
Probability
(ML)
Logic
(PL)
Bug-finder
Scalability

세 가지 소원



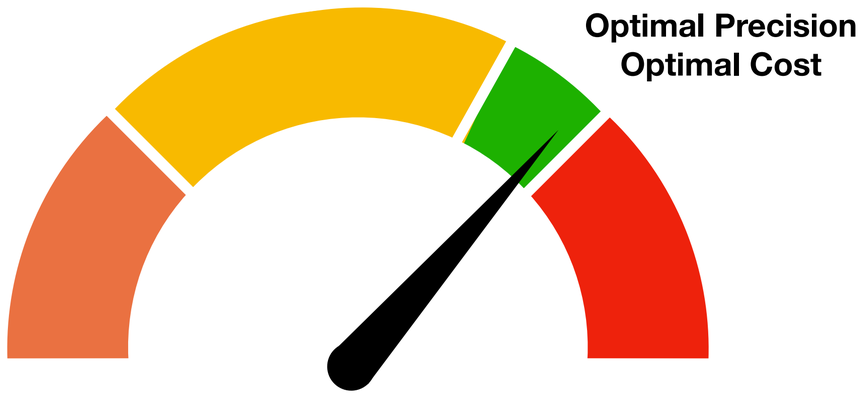
Adaptive Program Analysis

Interactive Program Analysis

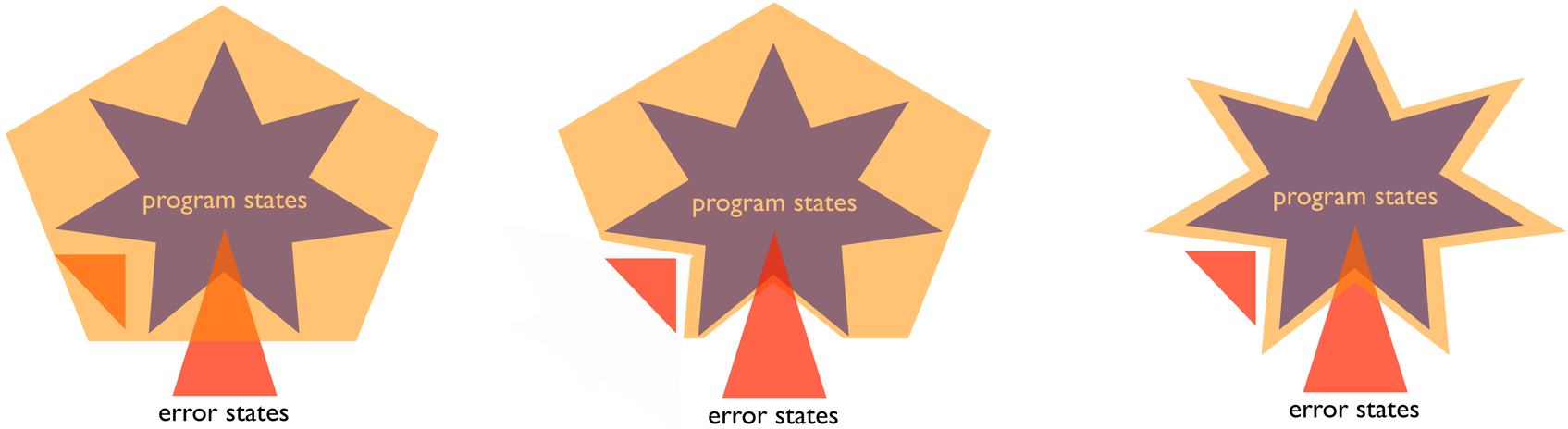


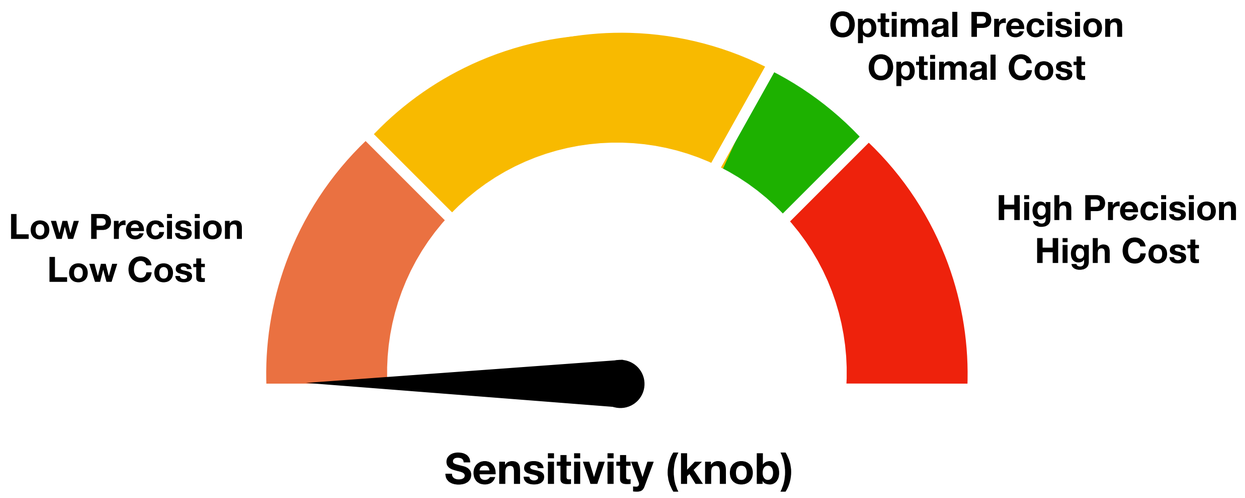
Continuous Program Analysis

Adaptive Program Analysis
[SAS'16, OOPSLA'17, ICSE'17, ICSE'19]

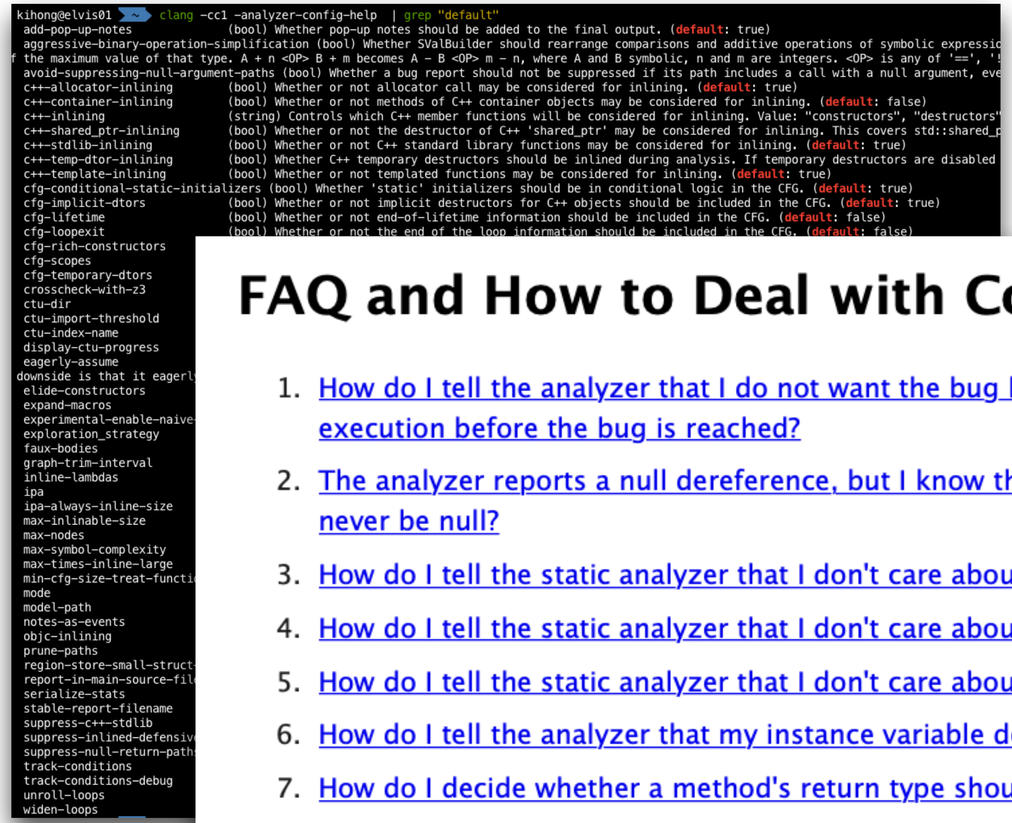


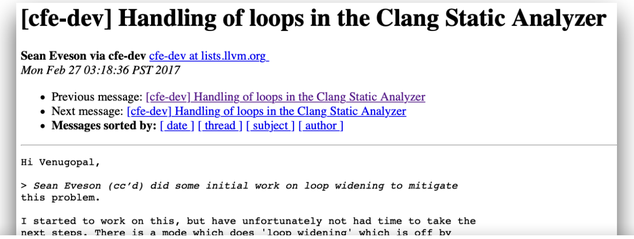
유연한 정적 분석


program states
error states
program states
error states
program states
error states


Optimal Precision
Optimal Cost
Low Precision
Low Cost
High Precision
High Cost
Sensitivity (knob)

정적 분석기 길들이기




[cfe-dev] Handling of loops in the Clang Static Analyzer

Sean Eveson via cfe-dev cfe-dev at lists.llvm.org
Mon Feb 27 03:18:36 PST 2017

- Previous message: [cfe-dev] Handling of loops in the Clang Static Analyzer
- Next message: [cfe-dev] Handling of loops in the Clang Static Analyzer
- **Messages sorted by:** [date] [thread] [subject] [author]

Hi Venugopal,

> *Sean Eveson (cc'd) did some initial work on loop widening to mitigate*
this problem.

I started to work on this, but have unfortunately not had time to take the
next steps. There is a mode which does 'loop widening' which is off by

FAQ and How to Deal with Common False Positives

1. How do I tell the analyzer that I do not want the bug being reported here since my custom error handler will safely end the execution before the bug is reached?
2. The analyzer reports a null dereference, but I know that the pointer is never null. How can I tell the analyzer that a pointer can never be null?
3. How do I tell the static analyzer that I don't care about a specific dead store?
4. How do I tell the static analyzer that I don't care about a specific unused instance variable in Objective C?
5. How do I tell the static analyzer that I don't care about a specific unlocalized string?
6. How do I tell the analyzer that my instance variable does not need to be released in -dealloc under Manual Retain/Release?
7. How do I decide whether a method's return type should be _Nullable or _Nonnull?
8. How do I tell the analyzer that I am intentionally violating nullability?
9. The analyzer assumes that a loop body is never entered. How can I tell it that the loop body will be entered at least once?
10. How can I suppress a specific analyzer warning?
11. How can I selectively exclude code the analyzer examines?

세상에 나쁜 정적 분석은 없다

공간탐색

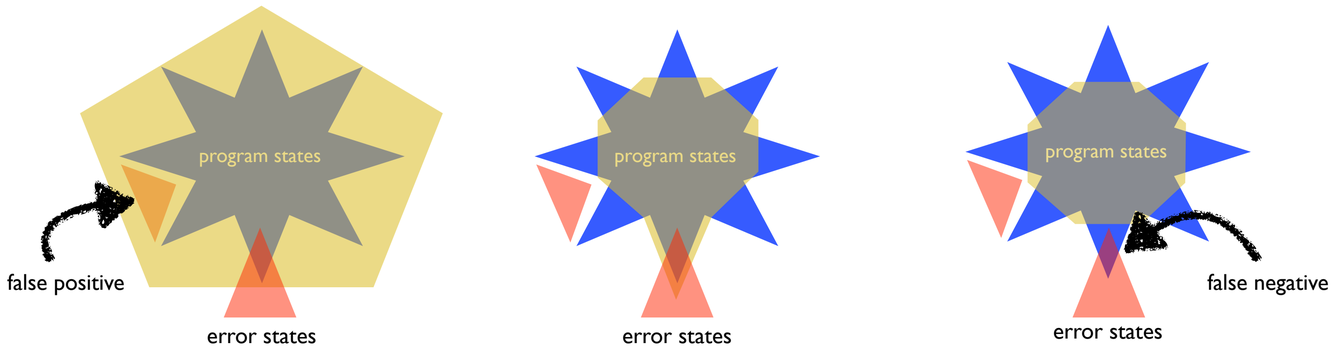
여러 사례

Abstraction (Knob)	Cost	Online/Offline	Method	Result
Variable Relationship	Running Time	Offline	Supervised Learning	[SAS'16]
Statement Order Variable Relationship	Running Time	Offline	Supervised Learning	[OOPSLA'17]
Loop Unrolling Library Call Handling	Missed Bugs	Offline	Supervised Learning	[ICSE'17]
Statement Order	Memory Consumption	Online	Reinforcement Learning	[ICSE'19]

선별적으로 안전한 분석

- 정확도를 높이기 위해 안전성 (soundness) 을 포기하는 여러 전략
  - 예: 유한번 순환문 풀기, 복잡한 라이브러리 함수 무시

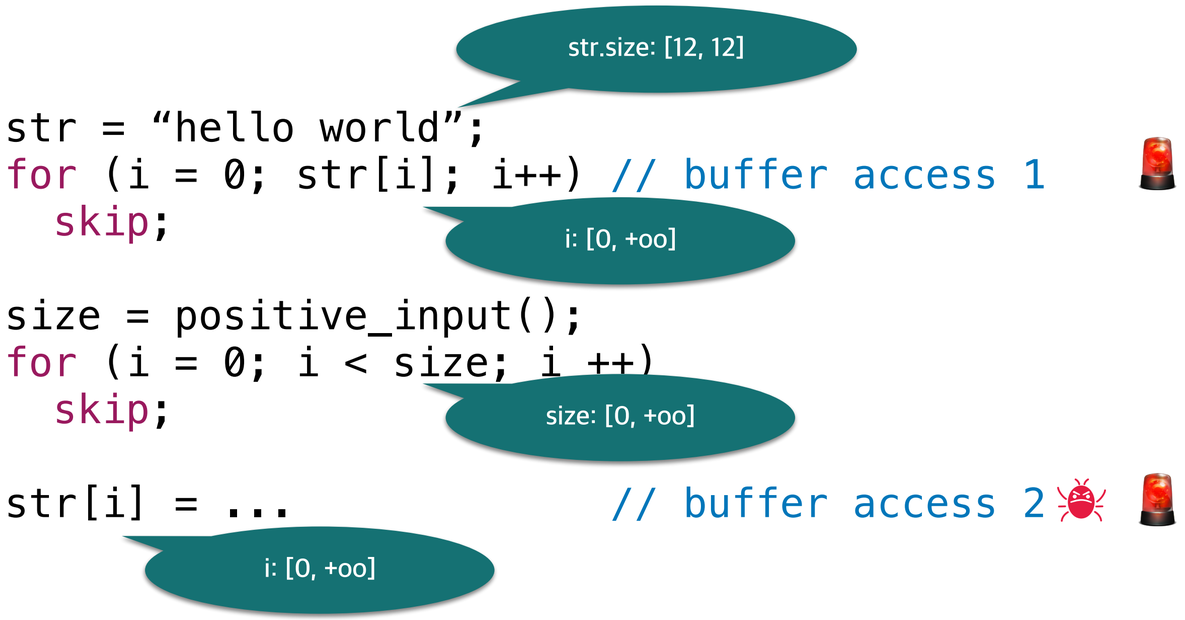
```
while(e){ C }  ▶  if(e){ C }      A;lib();B;  ▶  A;B;
```



Uniformly Sound　　Selectively Unsound　　Uniformly Unsound

예제

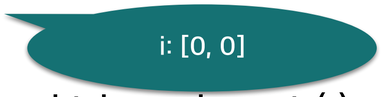
- 인터벌 도메인을 이용한 버퍼 오버런 분석기: 모든 순환문을 **안전하게** 분석

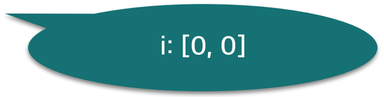


예제

- 인터벌 도메인을 이용한 버퍼 오버런 분석기: 모든 순환문을 **안전하지 않게** 분석

```
str = “hello world”;
i = 0;
if (str[i])                    // buffer access 1
  skip;

size = positive_input();
i = 0;
if (i < size)
  skip;

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





예제

- 인터벌 도메인을 이용한 버퍼 오버런 분석기: **필요할 때 적절히** 안전성을 포기하는 분석

```
str = “hello world”;
i = 0;
if (str[i])                    // buffer access 1
  skip;

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

str[i] = ...                   // buffer access 2 🐞🚨
```

i: [0, 0]

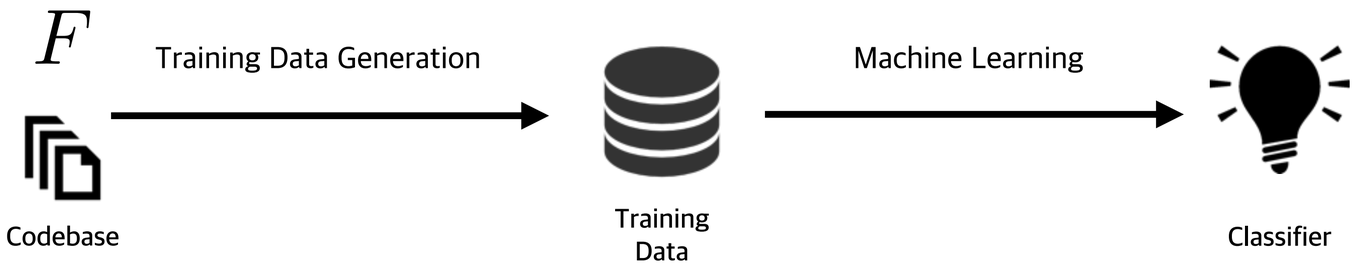
i: [0, +oo]

문제 정의

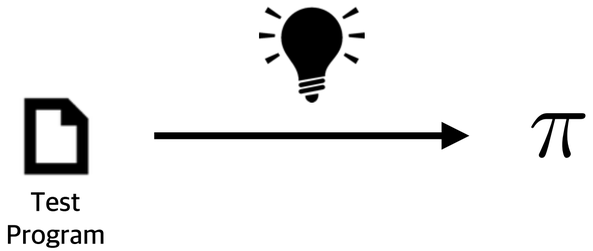
$$F \in Pgm \times \Pi \rightarrow \mathcal{A}$$

- 목표: 안전성 포기 대상의 집합 $\pi \in \Pi$ 찾기
  - 최대한 많은 오류를 찾아내면서, 거짓 경보는 최소화
  - 예: 순환문 ($\Pi = 2^{Loop}$), 라이브러리 호출 ($\Pi = 2^{Lib}$)
- 분석: 해당 집합의 원소들에 안전성 포기 전략을 적용

조감도



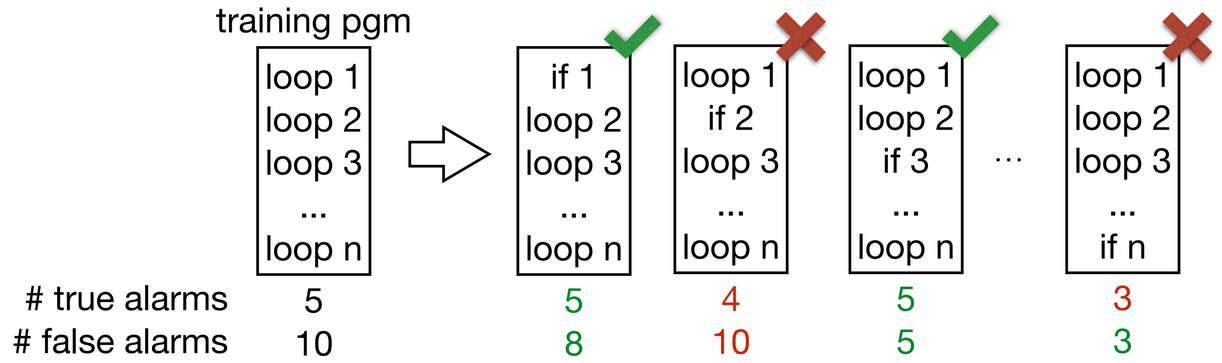
Training Harmless Unsoundness



Inferring Harmless Unsoundness

학습 데이터 수집

- 오류 지점이 알려져 있는 프로그램 집합 + 안전성을 조절할 수 있는 분석기
- 안전성 포기 전략이 적용되었을 때 1) 거짓 경보는 줄어들고, 2) 오류는 하나도 놓치지 않는 대상



특징 벡터와 학습

- 각 데이터를 특징 벡터 (feature vector) 로 표현

$$f(x) = \langle f_1(x), f_2(x), \ldots, f_n(x) \rangle$$

$$f(loop_1) = \langle 1, 0, \ldots, 1 \rangle$$
$$f(loop_2) = \langle 0, 1, \ldots, 1 \rangle$$

$$f(lib_1) = \langle 0, 1, \ldots, 0 \rangle$$
$$f(lib_2) = \langle 1, 1, \ldots, 1 \rangle$$

- 데이터를 토대로 분류기 (classifier) 를 학습
  - 널리 알려진 학습 알고리즘 이용 (예: SVM)

순환문의 특징 벡터

- 22 가지 구문적, 의미적 특징

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

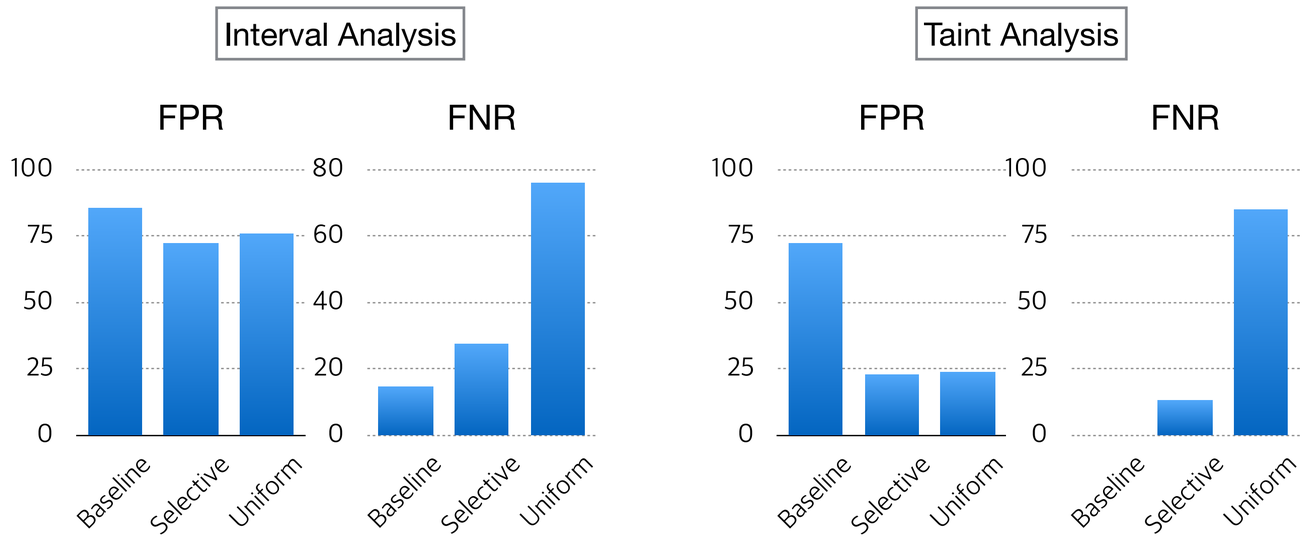
라이브러리 호출의 특징 벡터

- 15 가지 구문적, 의미적 특징

Feature	Property	Type	Description
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

성능

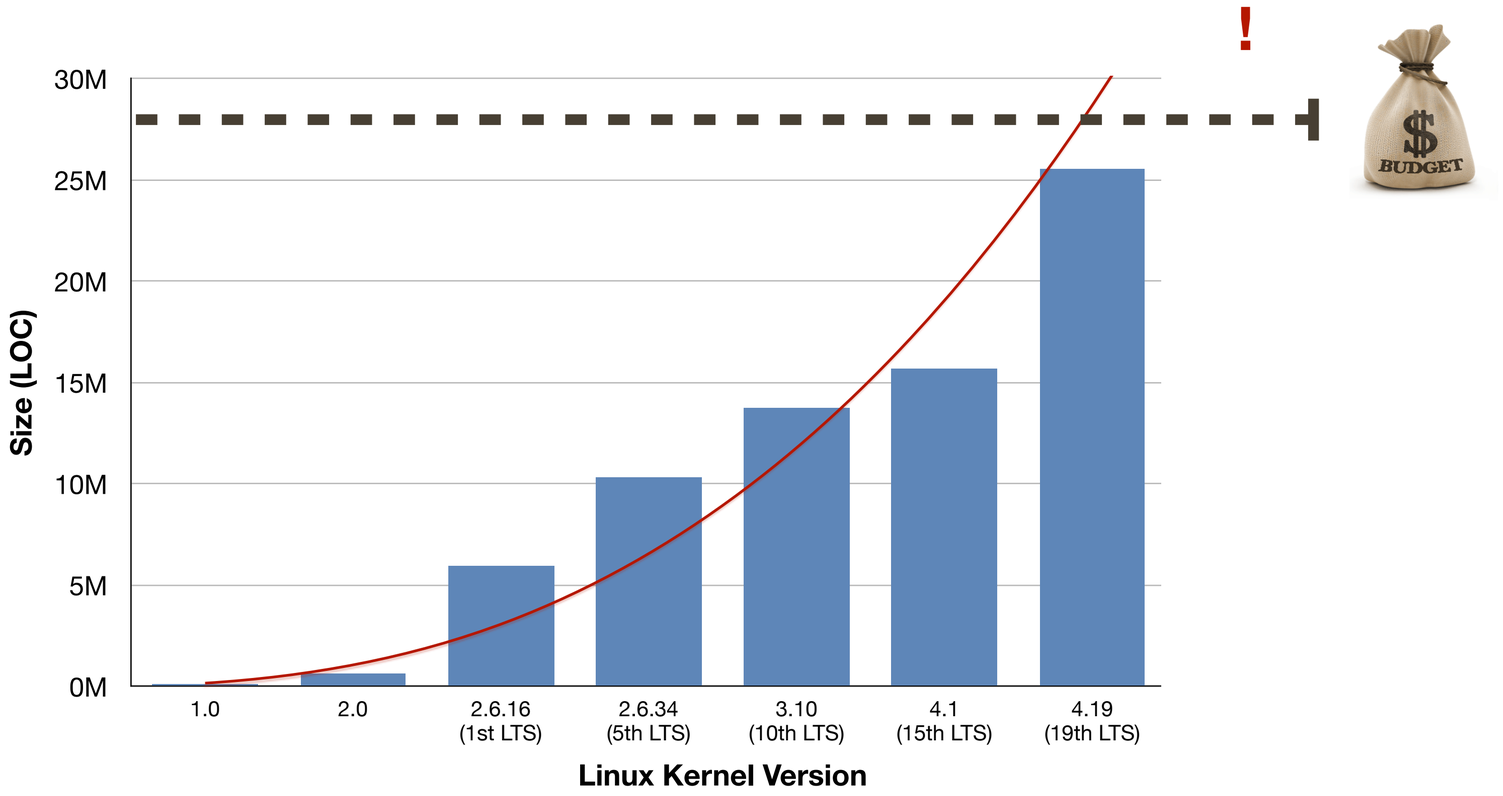
- 두 가지 분석
  - 정수 구간 (interval) 분석: 138 버퍼 오버런 오류 / 23 프로그램
  - 오염 (taint) 분석: 106 포맷 스트링 오류 / 13 프로그램



여러 사례

Abstraction (Knob)	Cost	Online/Offline	Method	Result
Variable Relationship	Running Time	Offline	Supervised Learning	[SAS'16]
Statement Order Variable Relationship	Running Time	Offline	Supervised Learning	[OOPSLA'17]
Loop Unrolling Library Call Handling	Missed Bugs	Offline	Supervised Learning	[ICSE'17]
Statement Order	Memory Consumption	Online	Reinforcement Learning	[ICSE'19]

어제, 오늘 그리고



*https://www.linuxcounter.net

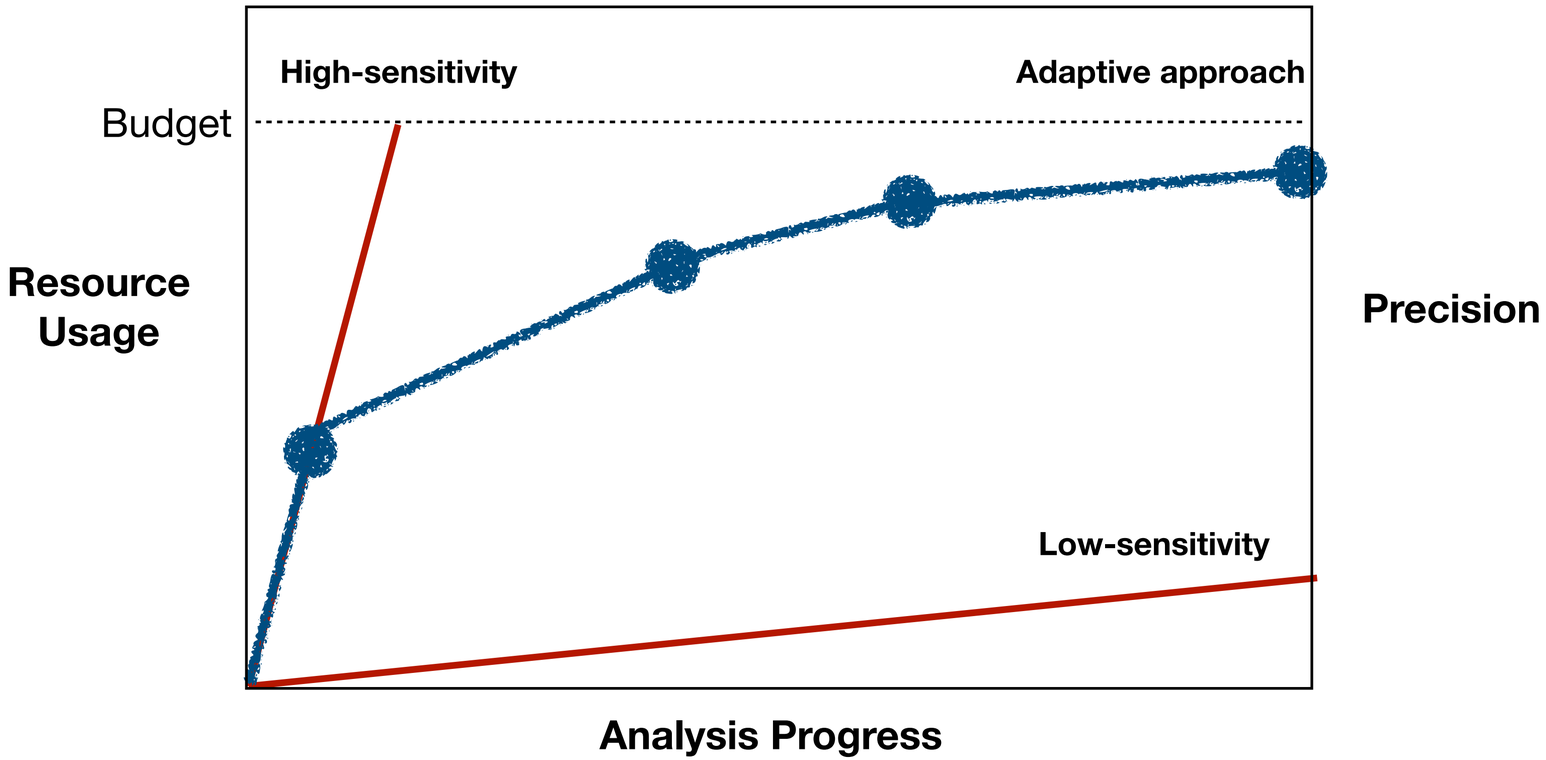
분석기 사용자의 어려움

- 분석기 동작을 예측하기가 상당히 어려움
  - 단순한 구문적 특징 (예: 소스 코드 크기) 으로는 예측하기 어려움
  - 복잡한 의미 구조가 결정하므로 분석 전문가가 아니면 이해하기 어려움

Sensitivity: 0% vim60 (227KLOC)	<	Sensitivity: 0% emacs-26.0.91 (503KLOC)	<	Sensitivity: 5% emacs-26.0.91 (503KLOC)
Memory: 51GB	>	Memory: 18GB	<<	Memory: > 128GB

자원 소모량을 스스로 고려하는 정적 분석

- 주어진 **자원 한도 내**에서 가능한 **최대 정확도를 자동**으로 달성하는 기술
  - 예: 128GB 메모리



요약 정도 조절 장치

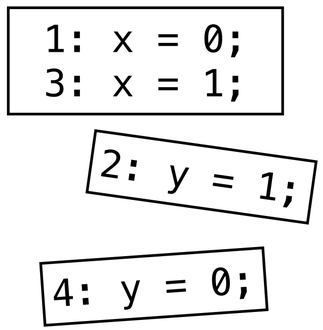
- Flow-sensitivity: 구문의 순서를 요약하는 정도

Flow-sensitive

```
1: x = 0;
2: y = 1;
3: x = 1;
4: y = 0;
```

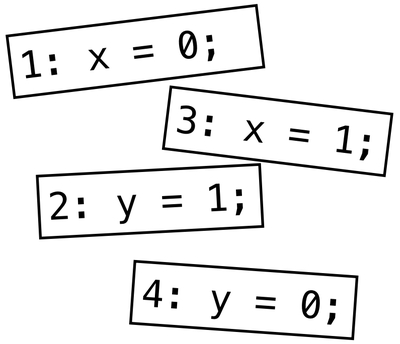
Line	State
1	{x = [0,0]}
2	{x = [0,0], y = [1,1]}
3	{x = [1,1], y = [1,1]}
4	{x = [1,1], y = [0,0]}

Partially Flow-sensitive



Line	FS	FI
1	{x = [0,0]}	{y = [0, 1]}
2	{x = [0,0]}	
3	{x = [1,1]}	
4	{x = [1,1]}	

Flow-insensitive



Line	State
*	{x = [0, 1], y = [0, 1]}

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Line	Flow-Sensitive Abstract State
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤, w = ⊤}

3 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Line	Flow-Sensitive Abstract State
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤, w = ⊤}
2	{x = [1,1], y = [0,0], z = [1,1], v = ⊤, w = ⊤}

6 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Line	Flow-Sensitive Abstract State
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤, w = ⊤}
2	{x = [1,1], y = [0,0], z = [1,1], v = ⊤, w = ⊤}
3	{x = [1,1], y = [0,0], z = [2,2], v = ⊤, w = ⊤}
4	{**x = [1,1], y = [1,1], z = [2,2]**, v = ⊤, w = ⊤}

12 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

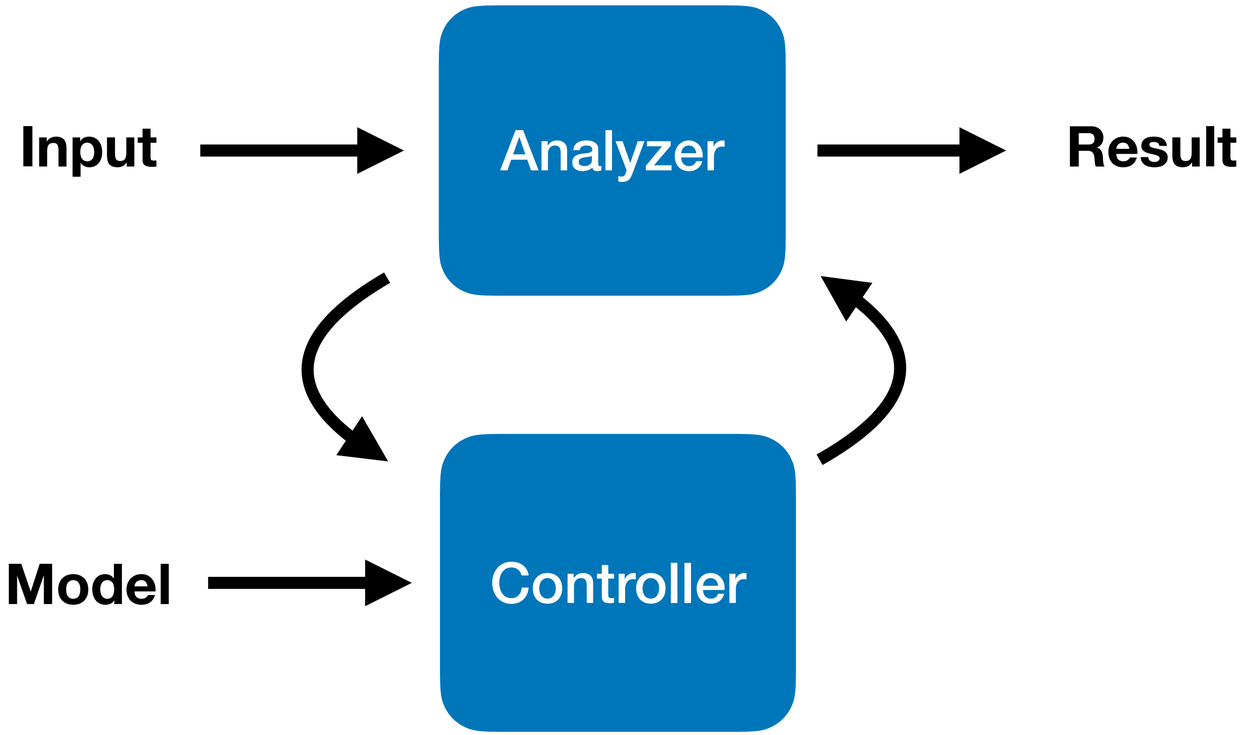
```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Line	Flow-Insensitive Abstract State
*	{**x = [0,+∞], y = [0,+∞], z = [1,+∞]**, v = ⊤, w = ⊤}

3 Intervals

유연한 정적 분석

- 분석 도중에 요약 정도를 스스로 조절
  - 여러 분석 데이터를 기반으로 확률 모델과 조절 장치를 스스로 학습



변수에 관한 확률 모델

- Model *M :* Variable → [0, 1]
- Importance of each variable in terms of flow-sensitivity
- Learned using Bayesian Optimization
  - represent variables as feature vectors and learn weights of features

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

M(x) > M(y) > M(z) > M(v) > M(w)

요약 조절 장치

- 조절 함수 $\boldsymbol{\pi} : \mathbf{F} \rightarrow \mathbf{Pr(A)}$ 이고 $\mathbf{A} = \{0, \cdots, 100\}$
- 입력: 현재 상태를 나타내는 특징 벡터
  - 예: 메모리 사용량, 분석 진행도
- 출력: 지금부터 몇 % 변수를 순서 무관하게 분석할지에 관한 확률 분포
- 강화 학습 방법을 이용하여 학습

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) > M(z) > M(v) > M(w)$

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) > M(z) > M(v) > M(w)$

Line	Flow-Sensitive Abstract State
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤, w = ⊤}
2	{x = [1,1], y = [0,0], z = [1,1], v = ⊤, w = ⊤}

6 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) > M(z) > M(v) > \cancel{M(w)}$

Line	Flow-Sensitive	Flow-Insensitive
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤}	{w = ⊤}
2	{x = [1,1], y = [0,0], z = [1,1], v = ⊤}	

6 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) > M(z) > M(v) > \cancel{M(w)}$

Line	Flow-Sensitive	Flow-Insensitive
1	{x = [0,0], y = [0,0], z = [1,1], v = ⊤}	
2	{x = [1,1], y = [0,0], z = [1,1], v = ⊤}	{w = ⊤}
3	{x = [1,1], y = [0,0], z = [2,2], v = ⊤}	

9 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) \not> M(z) \not> M(v) \not> M(w)$

Line	Flow-Sensitive	Flow-Insensitive
1	{x = [0,0], y = [0,0]}	{z = [1,+∞], v = ⊤, w = ⊤}
2	{x = [1,+∞], y = [0,0]}	
3	{x = [1,+∞], y = [0,0]}	

6 Intervals

예제

- Partially flow-sensitive interval analysis (budget: 10 intervals)

```
1: x = 0; y = 0; z = 1; v = input(); w = input();
2: x = z;
3: z = z + 1;
4: y = x;
5: assert(y > 0);     // Query 1 (hold)
6: assert(z > 0);     // Query 2 (hold)
7: assert(v == w);    // Query 3 (may fail)
```

Model: $M(x) > M(y) > M(z) > M(v) > M(w)$

Line	Flow-Sensitive	Flow-Insensitive
1	{x = [0,0], y = [0,0]}	
2	{x = [1,+∞], y = [0,0]}	{**z = [1,+∞]**, v = ⊤, w = ⊤}
3	{x = [1,+∞], y = [0,0]}	
4	**{x = [1,+∞], y = [1,+∞]}**	

8 Intervals

요약 조절 장치 학습

- 조절 함수 $\boldsymbol{\pi} : \mathbf{F} \rightarrow \mathbf{Pr(A)}$ 이고 $\mathbf{A} = \{0, \cdots, 100\}$
- 입력: 현재 상태를 나타내는 특징 벡터
  - 예: 메모리 사용량, 분석 진행도
- 출력: 지금부터 몇 % 변수를 순서 무관하게 분석할지에 관한 확률 분포
- Value function $Q : \mathbf{F} \times \mathbf{A} \rightarrow$ [0, 1] : 각 특징 벡터와 행동에 관한 점수
- $\boldsymbol{\pi}_Q$(f)(a) = $\dfrac{Q(f, a)}{\sum_{a' \in A} Q(f, a')}$

Value Function

$$Q : \mathbf{F} \times \mathbf{A} \rightarrow [0, 1]$$

- 특징 요약 함수 α : State $\rightarrow \mathbf{F}$ 이고 $\mathbf{F} = [0, 1]^4$

  1. The inverse of memory budget
  2. Current memory consumption divided by the total budget
  3. Current lattice position divided by the lattice height
  4. Current workset size divided by the total workset size

- 보상 함수: FI 와 FS 사이에서 표준화한 상대적 알람 개수
  - 0 if #alarms = #flow-insensitive alarms
  - 1 if #alarms = #flow-sensitive alarms

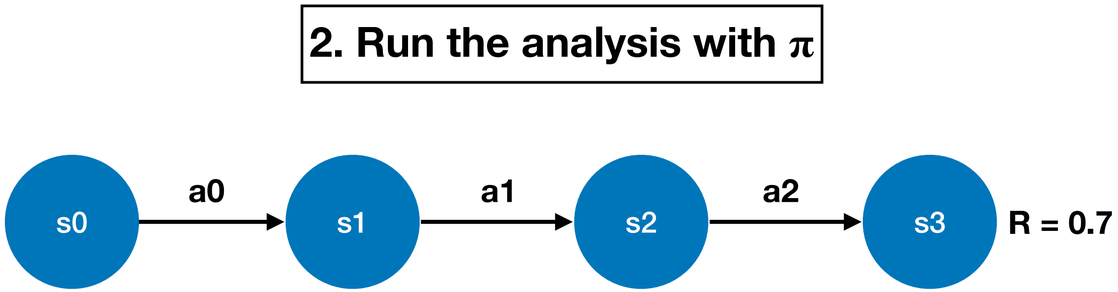
학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용

1. Initialize π with a random policy

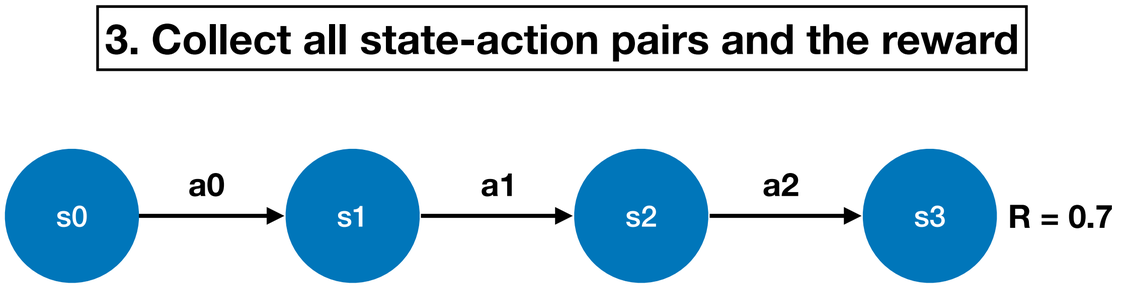
학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



학습 알고리즘

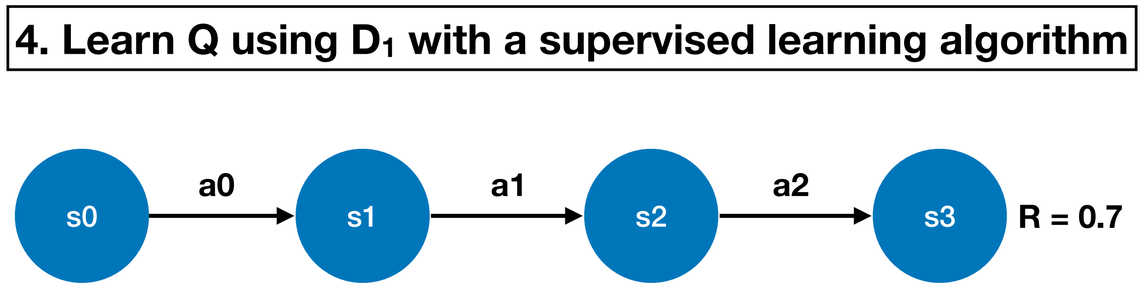
- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



$D_1 = \{(<\alpha(s_0), a_0>, 0.7), (<\alpha(s_1), a_1>, 0.7), (<\alpha(s_2), a_2>, 0.7)\}$

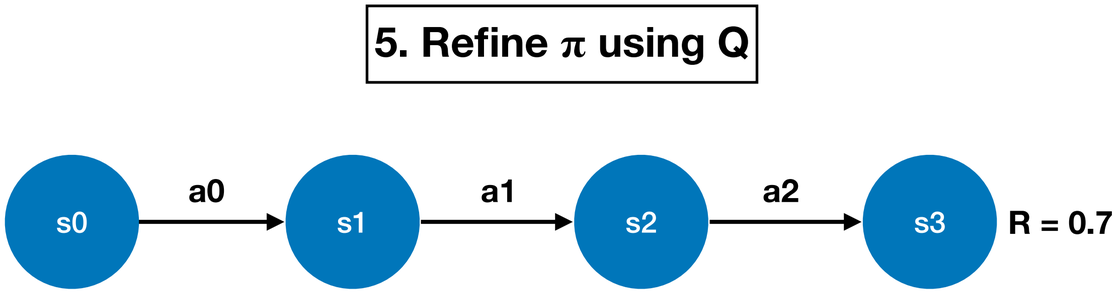
학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



학습 알고리즘

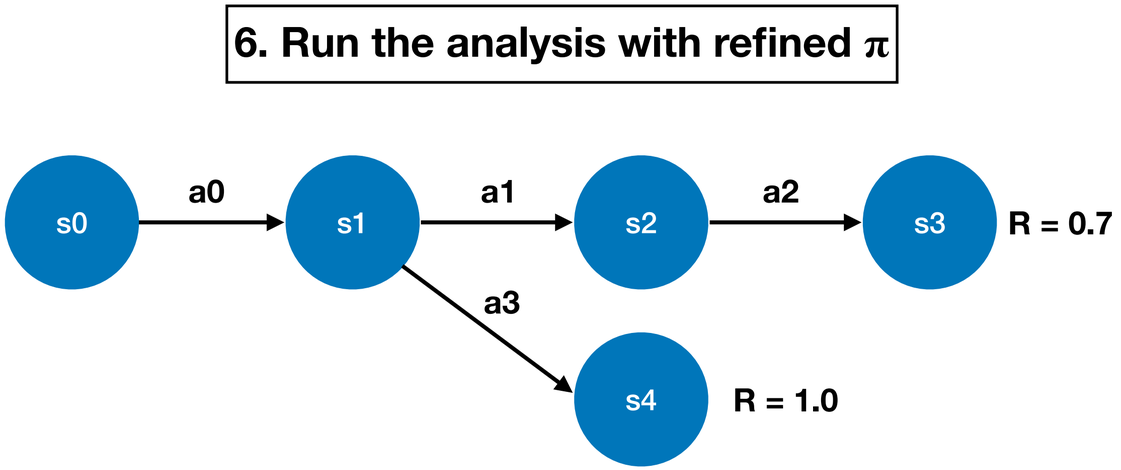
- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



$$\pi_Q(f)(a) = \frac{Q(f,a)}{\sum_{a' \in A} Q(f,a')}$$

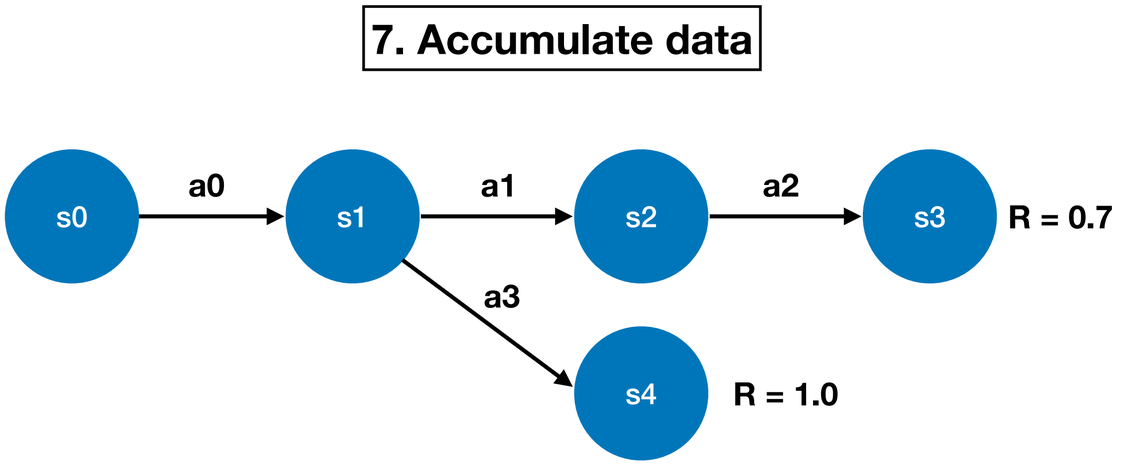
학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



학습 알고리즘

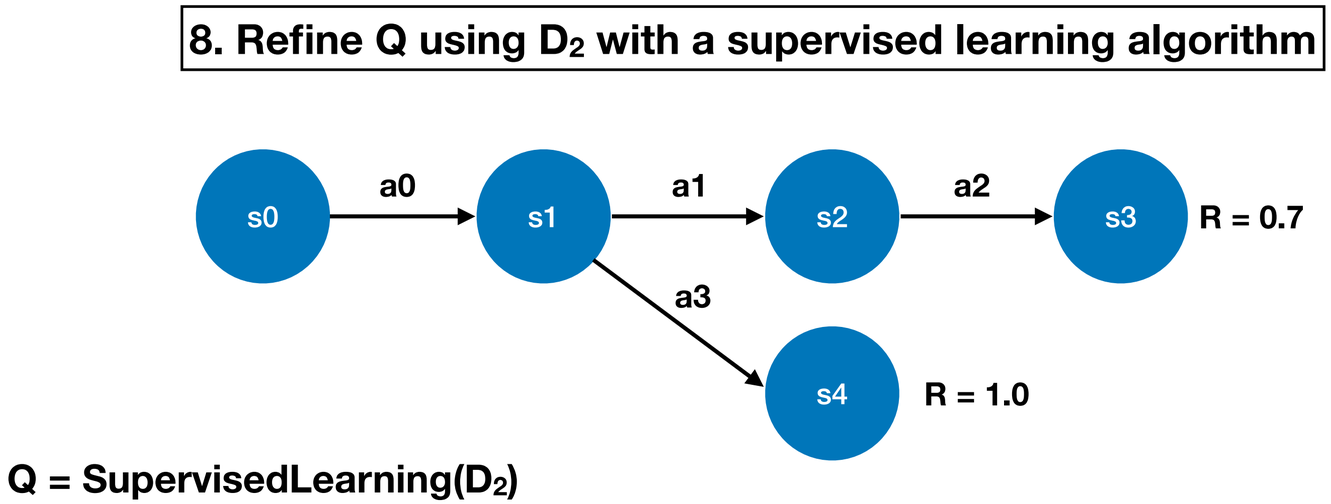
- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



$D_2 = D_1 \cup \{(<\alpha(s_0), a_0>, 1.0), (<\alpha(s_4), a_4>, 1.0)\}$

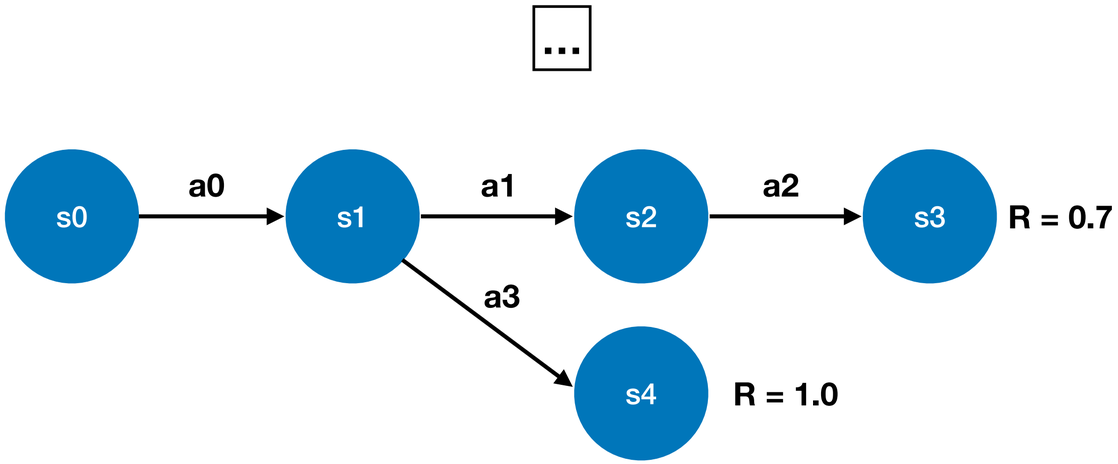
학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용

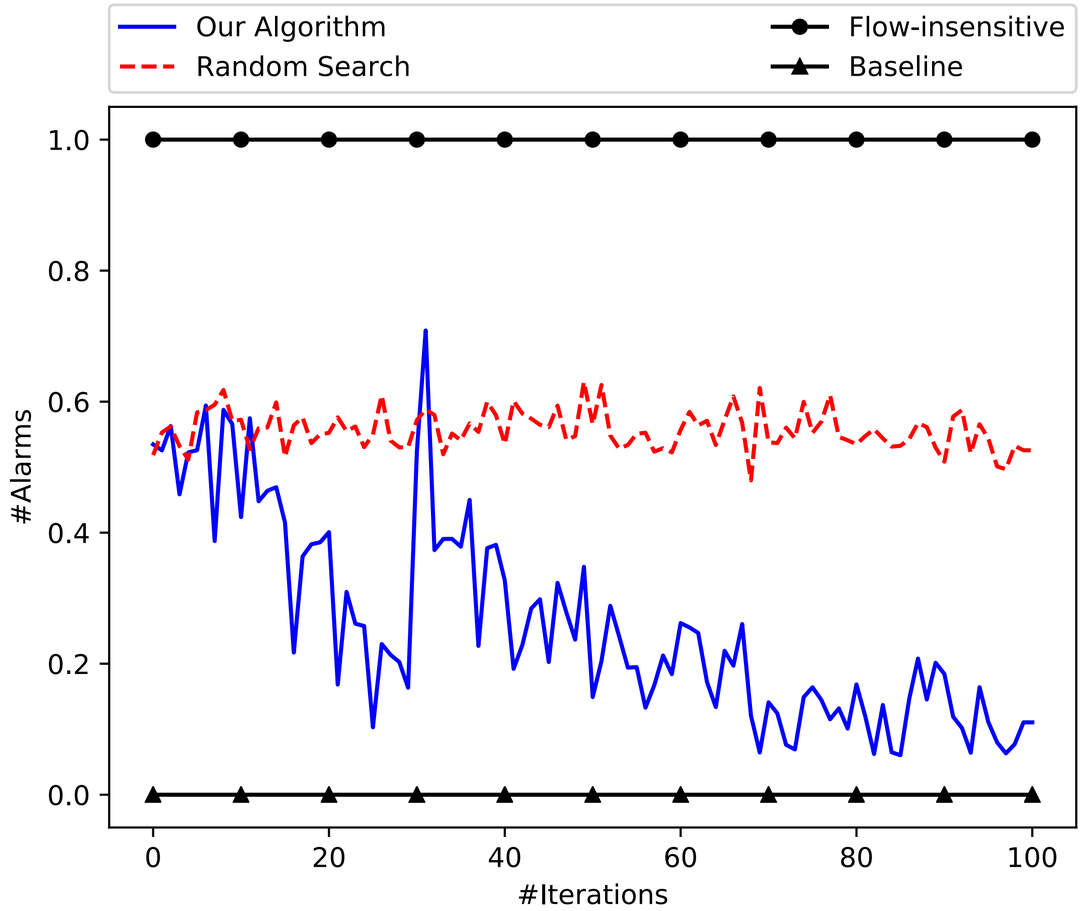


학습 알고리즘

- 미리 제공된 학습 데이터와 강화 학습 알고리즘 이용



학습 진행


Our Algorithm
Random Search
Flow-insensitive
Baseline
1.0
0.8
0.6
0.4
0.2
0.0
#Alarms
0
20
40
60
80
100
#Iterations

효과

- Achieving **maximum precision** within a given **resource budget**
  - e.g., within 128GB of memory

기존 방식*
(10% flow-sensitivity)

- **3/8** run out of memory (128GB)
- **27%** of buffer overrun alarms ↓
- **30%** of null dereference alarms ↓

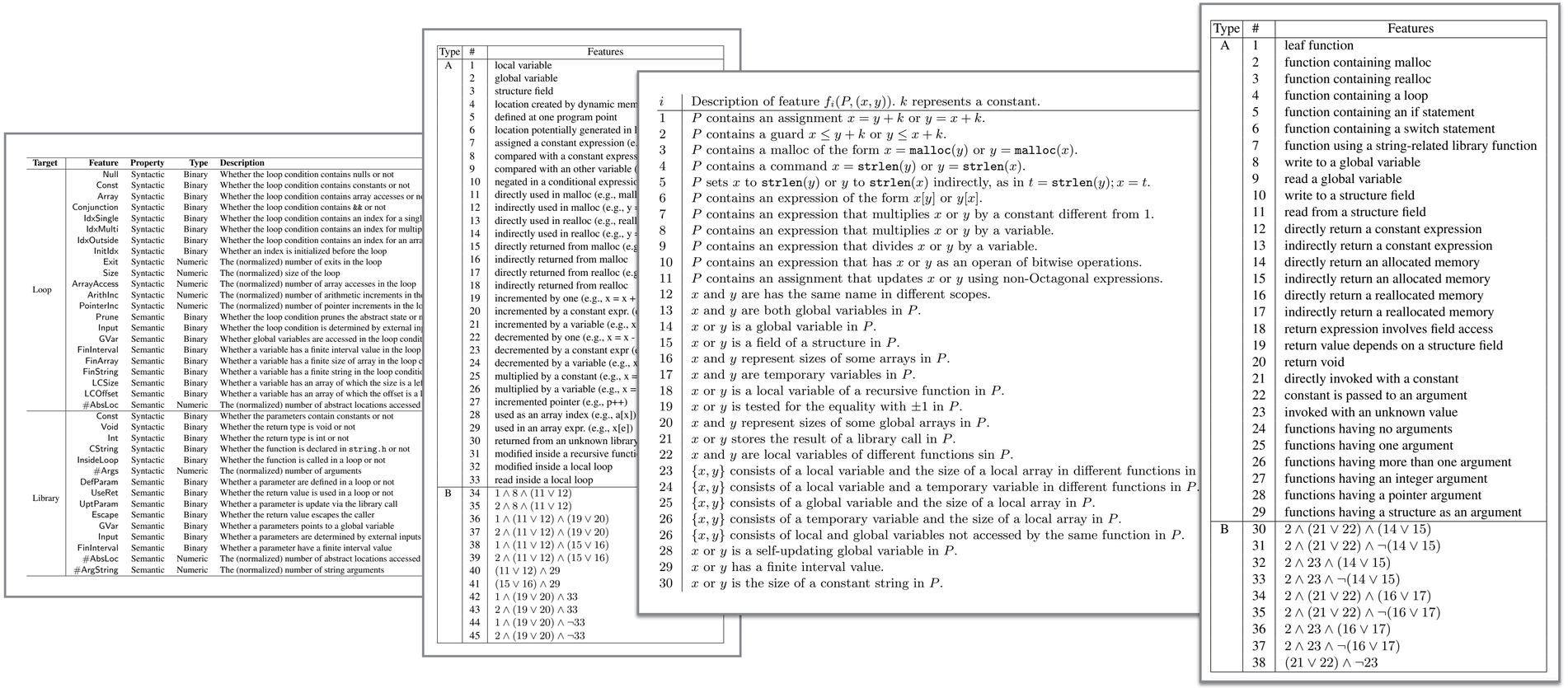
자동 조절 장치 기반

- **0/8** run out of memory (64 / 128GB)
- **28—32%** of buffer overrun alarms ↓
- **33—41%** of null dereference alarms ↓

*Kihong Heo, Hakjoo Oh, Hongseok Yang, Kwangkeun Yi. Adapting Static Analysis via Learning with Bayesian Optimization. ACM TOPLAS, 2018

한계점

- 각 대상의 특징 (feature) 을 수동으로 정의해야 함



Target	Feature	Property	Type	Description
Loop	Null	Syntactic	Binary	Whether the loop condition contains nulls or not
	Const	Syntactic	Binary	Whether the loop condition contains constants or not
	Array	Syntactic	Binary	Whether the loop condition contains array accesses or n
	Conjunction	Syntactic	Binary	Whether the loop condition contains && or not
	IdxSingle	Syntactic	Binary	Whether the loop condition contains an index for a singl
	IdxMulti	Syntactic	Binary	Whether the loop condition contains an index for multip
	IdxOutside	Syntactic	Binary	Whether the loop condition contains an index for an arra
	InitIdx	Syntactic	Binary	Whether an index is initialized before the loop
	Exit	Syntactic	Numeric	The (normalized) number of exits in the loop
	Size	Syntactic	Numeric	The (normalized) size of the loop
	ArrayAccess	Syntactic	Numeric	The (normalized) number of array accesses in the loop
	ArithInc	Syntactic	Numeric	The (normalized) number of arithmetic increments in th
	PointerInc	Syntactic	Numeric	The (normalized) number of pointer increments in the lo
	Prune	Semantic	Binary	Whether the loop condition prunes the abstract state or r
	Input	Semantic	Binary	Whether the loop condition is determined by external in
	GVar	Semantic	Binary	Whether global variables are accessed in the loop condit
	FinInterval	Semantic	Binary	Whether a variable has a finite interval value in the loop
	FinArray	Semantic	Binary	Whether a variable has a finite size of array in the loop c
	FinString	Semantic	Binary	Whether a variable has a finite string in the loop conditio
	LCSize	Semantic	Binary	Whether a variable has an array of which the size is a let
	LCOffset	Semantic	Binary	Whether a variable has an array of which the offset is a l
	#AbsLoc	Semantic	Numeric	The (normalized) number of abstract locations accessed
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
	#ArgString	Semantic	Numeric	The (normalized) number of string arguments

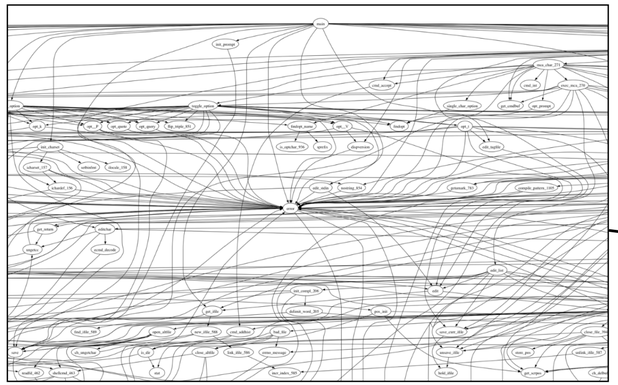
Type	#	Features
A	1	local variable
	2	global variable
	3	structure field
	4	location created by dynamic mem
	5	defined at one program point
	6	location potentially generated in l
	7	assigned a constant expression (e.
	8	compared with a constant express
	9	compared with an other variable (
	10	negated in a conditional expressic
	11	directly used in malloc (e.g., mall
	12	indirectly used in malloc (e.g., y =
	13	directly used in realloc (e.g., reall
	14	indirectly used in realloc (e.g., y =
	15	directly returned from malloc (e.g
	16	indirectly returned from malloc
	17	directly returned from realloc (e.g
	18	indirectly returned from realloc
	19	incremented by one (e.g., x = x +
	20	incremented by a constant expr. (
	21	incremented by a variable (e.g., x
	22	decremented by one (e.g., x = x -
	23	decremented by a constant expr (
	24	decremented by a variable (e.g., x
	25	multiplied by a constant (e.g., x =
	26	multiplied by a variable (e.g., x =
	27	incremented pointer (e.g., p++)
	28	used as an array index (e.g., a[x])
	29	used in an array expr. (e.g., x[e])
	30	returned from an unknown library
	31	modified inside a recursive functi
	32	modified inside a local loop
	33	read inside a local loop
B	34	$1 \wedge 8 \wedge (11 \vee 12)$
	35	$2 \wedge 8 \wedge (11 \vee 12)$
	36	$1 \wedge (11 \vee 12) \wedge (19 \vee 20)$
	37	$2 \wedge (11 \vee 12) \wedge (19 \vee 20)$
	38	$1 \wedge (11 \vee 12) \wedge (15 \vee 16)$
	39	$2 \wedge (11 \vee 12) \wedge (15 \vee 16)$
	40	$(11 \vee 12) \wedge 29$
	41	$(15 \vee 16) \wedge 29$
	42	$1 \wedge (19 \vee 20) \wedge 33$
	43	$2 \wedge (19 \vee 20) \wedge 33$
	44	$1 \wedge (19 \vee 20) \wedge \neg 33$
	45	$2 \wedge (19 \vee 20) \wedge \neg 33$

i	Description of feature $f_i(P,(x,y))$. k represents a constant.
1	P contains an assignment $x = y + k$ or $y = x + k$.
2	P contains a guard $x \leq y + k$ or $y \leq x + k$.
3	P contains a malloc of the form $x = \texttt{malloc}(y)$ or $y = \texttt{malloc}(x)$.
4	P contains a command $x = \texttt{strlen}(y)$ or $y = \texttt{strlen}(x)$.
5	P sets x to $\texttt{strlen}(y)$ or y to $\texttt{strlen}(x)$ indirectly, as in $t = \texttt{strlen}(y); x = t$.
6	P contains an expression of the form $x[y]$ or $y[x]$.
7	P contains an expression that multiplies x or y by a constant different from 1.
8	P contains an expression that multiplies x or y by a variable.
9	P contains an expression that divides x or y by a variable.
10	P contains an expression that has x or y as an operan of bitwise operations.
11	P contains an assignment that updates x or y using non-Octagonal expressions.
12	x and y are has the same name in different scopes.
13	x and y are both global variables in P.
14	x or y is a global variable in P.
15	x or y is a field of a structure in P.
16	x and y represent sizes of some arrays in P.
17	x and y are temporary variables in P.
18	x or y is a local variable of a recursive function in P.
19	x or y is tested for the equality with ± 1 in P.
20	x and y represent sizes of some global arrays in P.
21	x or y stores the result of a library call in P.
22	x and y are local variables of different functions sin P.
23	$\{x, y\}$ consists of a local variable and the size of a local array in different functions in
24	$\{x, y\}$ consists of a local variable and a temporary variable in different functions in P.
25	$\{x, y\}$ consists of a global variable and the size of a local array in P.
26	$\{x, y\}$ consists of a temporary variable and the size of a local array in P.
26	$\{x, y\}$ consists of local and global variables not accessed by the same function in P.
28	x or y is a self-updating global variable in P.
29	x or y has a finite interval value.
30	x or y is the size of a constant string in P.

Type	#	Features
A	1	leaf function
	2	function containing malloc
	3	function containing realloc
	4	function containing a loop
	5	function containing an if statement
	6	function containing a switch statement
	7	function using a string-related library function
	8	write to a global variable
	9	read a global variable
	10	write to a structure field
	11	read from a structure field
	12	directly return a constant expression
	13	indirectly return a constant expression
	14	directly return an allocated memory
	15	indirectly return an allocated memory
	16	directly return a reallocated memory
	17	indirectly return a reallocated memory
	18	return expression involves field access
	19	return value depends on a structure field
	20	return void
	21	directly invoked with a constant
	22	constant is passed to an argument
	23	invoked with an unknown value
	24	functions having no arguments
	25	functions having one argument
	26	functions having more than one argument
	27	functions having an integer argument
	28	functions having a pointer argument
	29	functions having a structure as an argument
B	30	$2 \wedge (21 \vee 22) \wedge (14 \vee 15)$
	31	$2 \wedge (21 \vee 22) \wedge \neg(14 \vee 15)$
	32	$2 \wedge 23 \wedge (14 \vee 15)$
	33	$2 \wedge 23 \wedge \neg(14 \vee 15)$
	34	$2 \wedge (21 \vee 22) \wedge (16 \vee 17)$
	35	$2 \wedge (21 \vee 22) \wedge \neg(16 \vee 17)$
	36	$2 \wedge 23 \wedge (16 \vee 17)$
	37	$2 \wedge 23 \wedge \neg(16 \vee 17)$
	38	$(21 \vee 22) \wedge \neg 23$

특징 자동 생성

- 핵심 아이디어: 특징을 나타내는 프로그램
  - 왜 특정 분석 기술 (flow-sensitivity, relational analysis, etc) 이 먹히는가?
- 특징 프로그램 생성: 이미 존재하는 프로그램 축약기 이용



“FS works but FI fails”

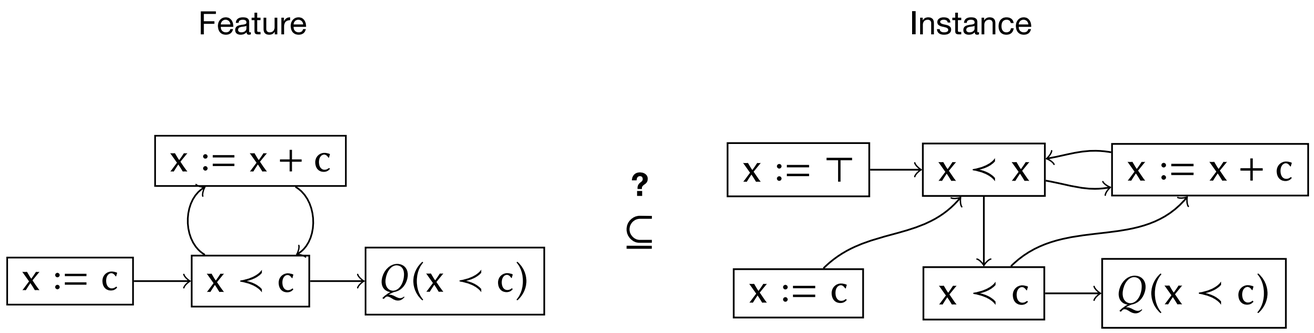
C-Reduce*

```
/* feature program */
a = 0;
while(1) {
  if (a < 3)
    assert(a < 5);
  a++;
}
```

*https://embed.cs.utah.edu/creduce

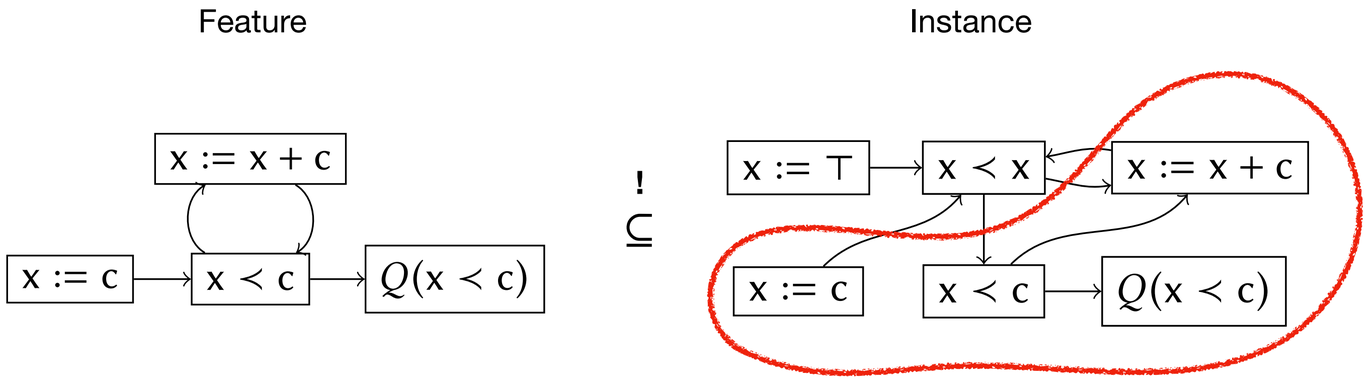
특징 추출

- 특징 프로그램과 대상 프로그램을 그래프로 표현하여 포함관계를 계산
  - 그래프: 요약된 데이터 흐름을 표현



특징 추출

- 특징 프로그램과 대상 프로그램을 그래프로 표현하여 포함관계를 계산
  - 그래프: 요약된 데이터 흐름을 표현

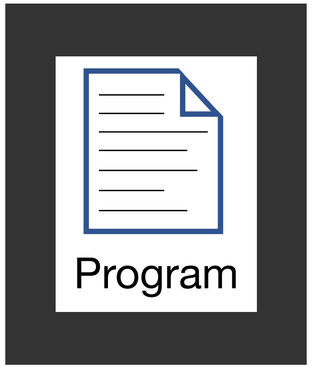


$$Node_F \subseteq Node_I \ \wedge \ Edge_F \subseteq Edge_I^*$$

Outline

Adaptive Program Analysis

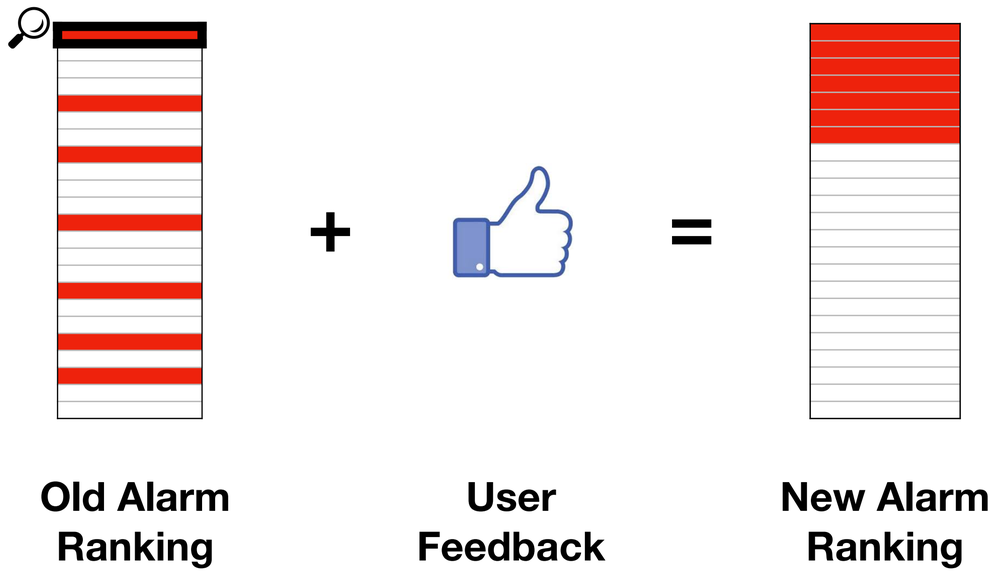
Interactive Program Analysis



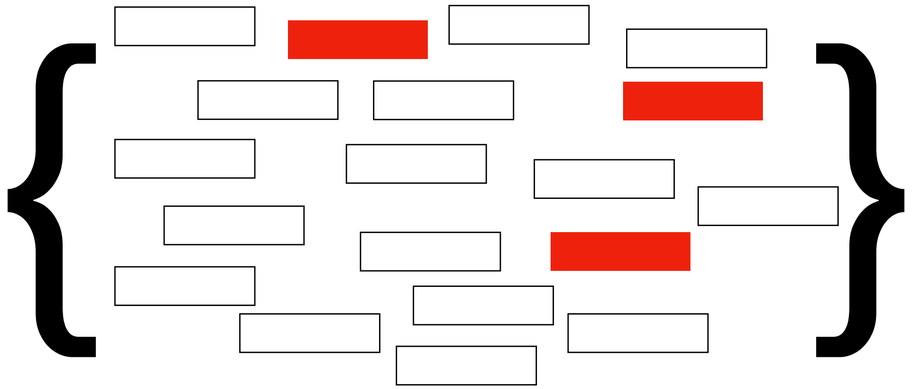
Continuous Program Analysis

Outline

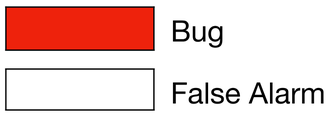
Bingo: Interactive Alarm Ranking System [PLDI'18]



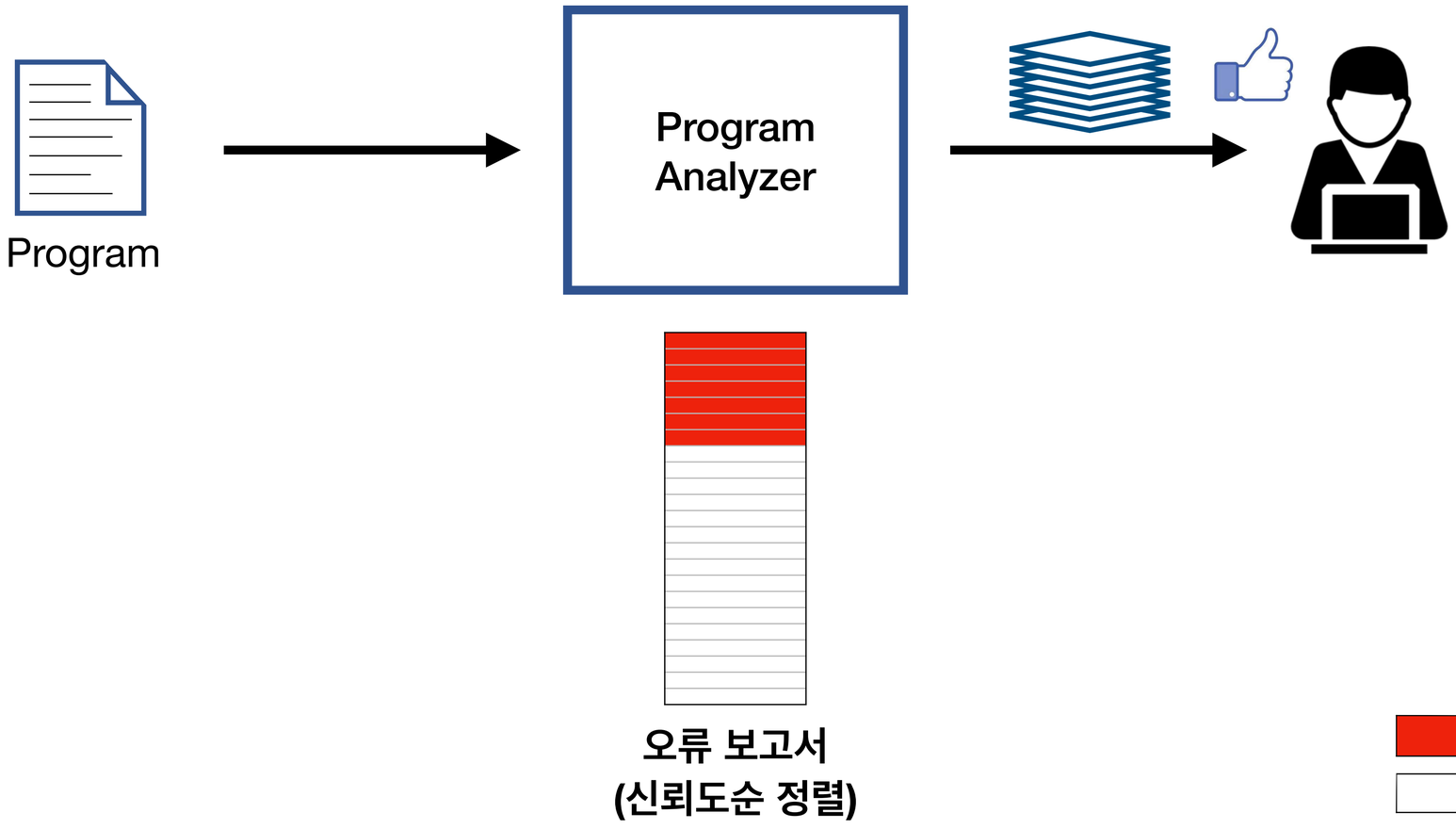
문제점



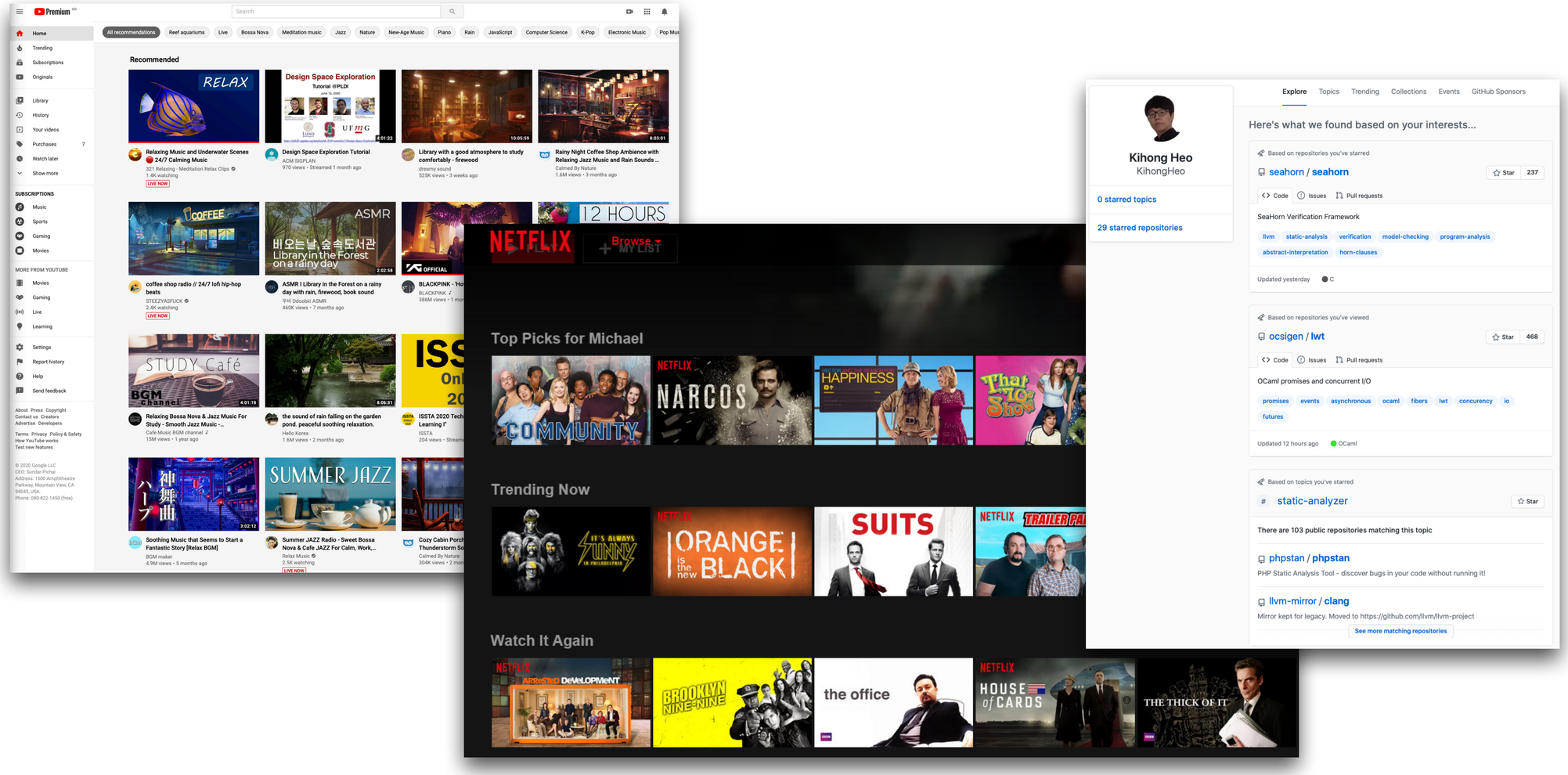
오류 보고서
(뒤죽박죽, 중구난방)



목표

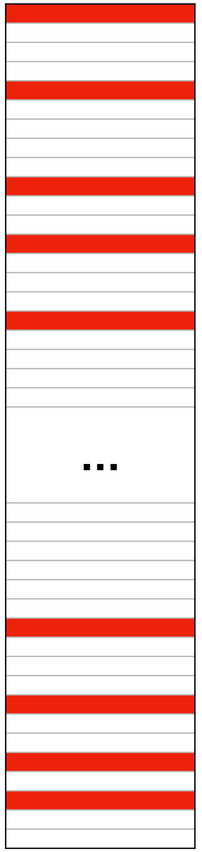


상호작용 시스템 (다른 분야)



반응형 오류 보고

Rank 1



Rank 522

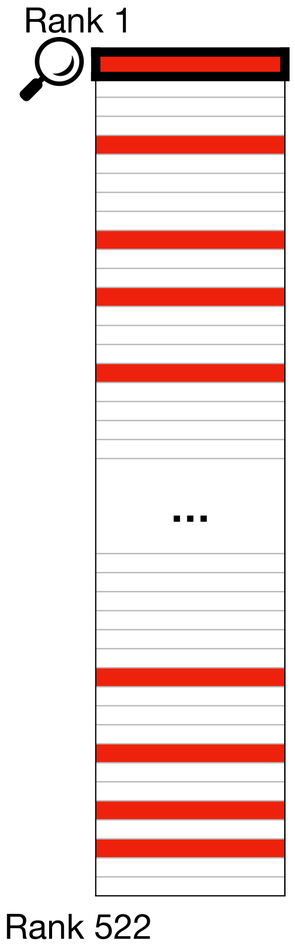
Apache FtpServer

152KLOC
75 Datarace Bugs
522 Total Alarms

Bug

False Alarm

반응형 오류 보고



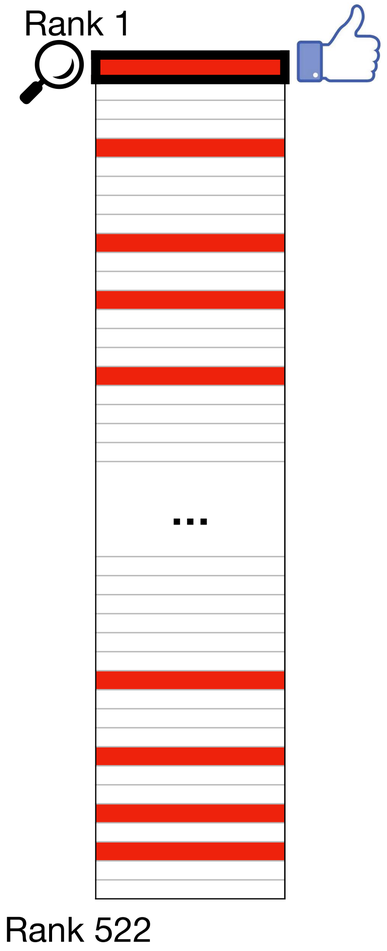
Apache FtpServer

152KLOC
75 Datarace Bugs
522 Total Alarms

Bug

False Alarm

반응형 오류 보고

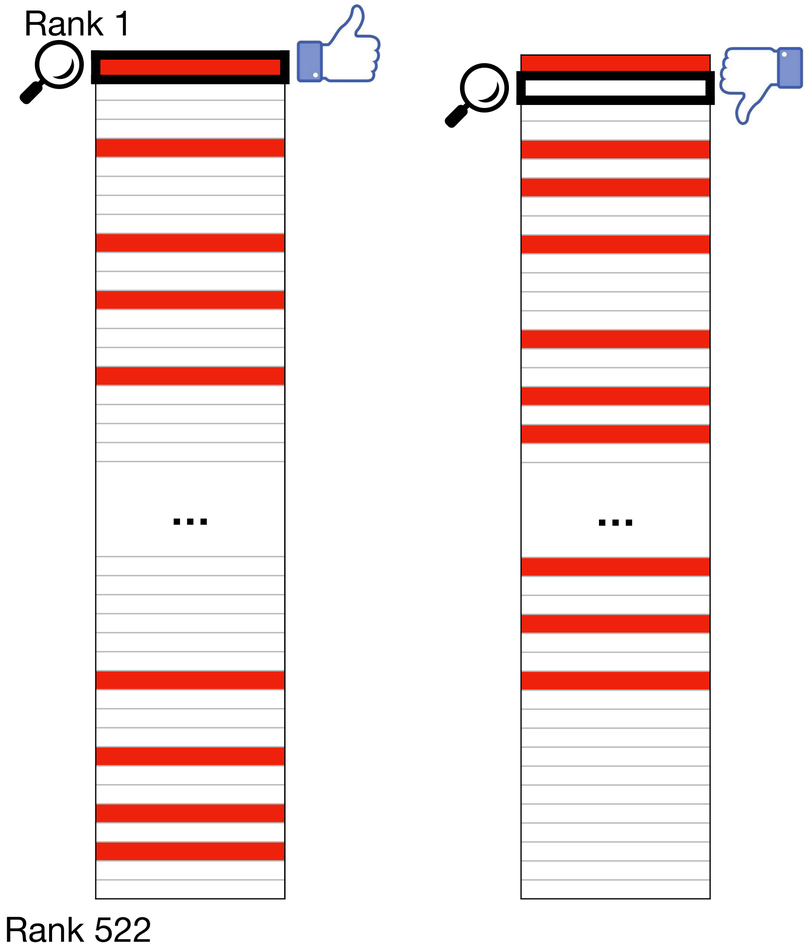


152KLOC
75 Datarace Bugs
522 Total Alarms

Bug

False Alarm

반응형 오류 보고

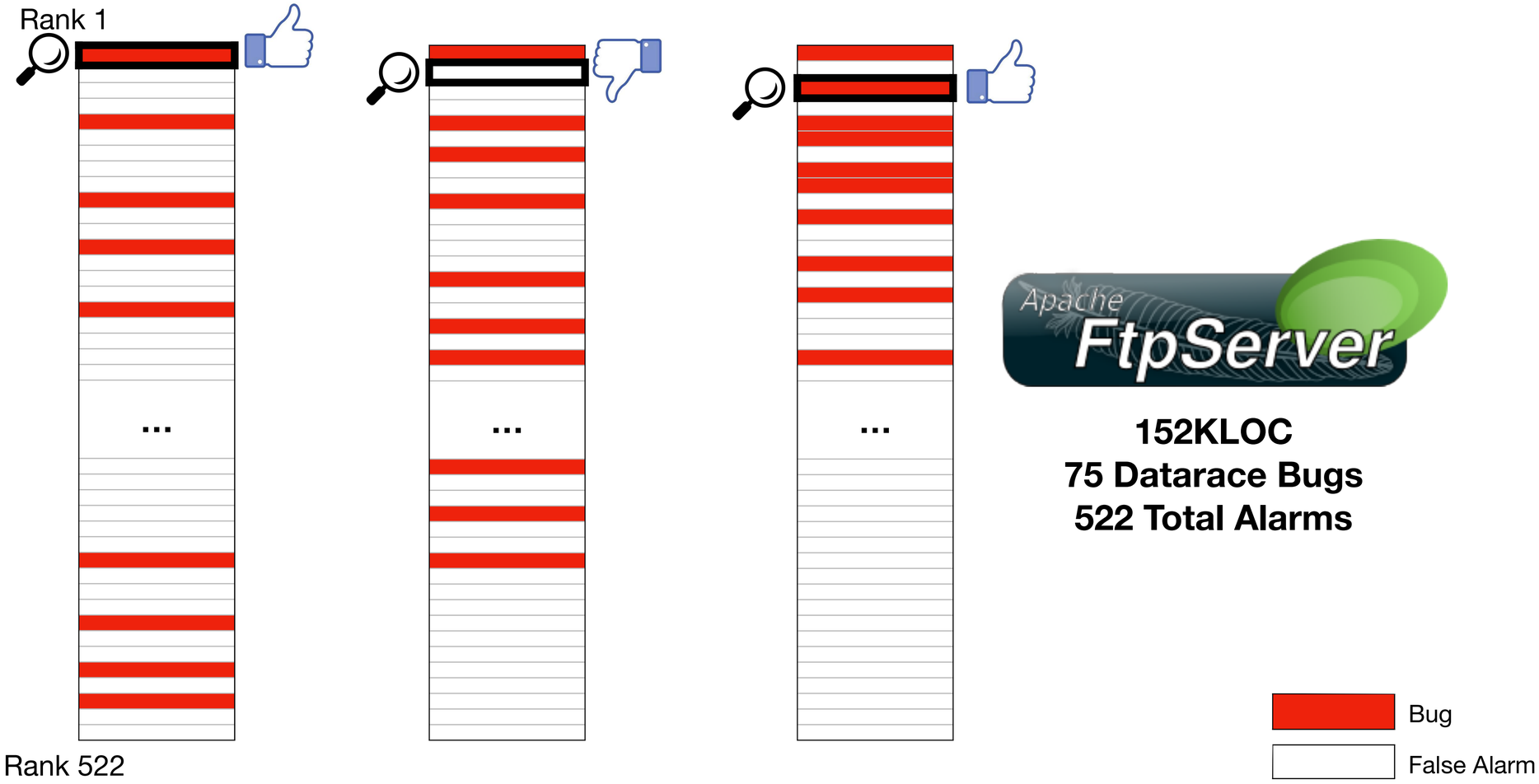


152KLOC
75 Datarace Bugs
522 Total Alarms

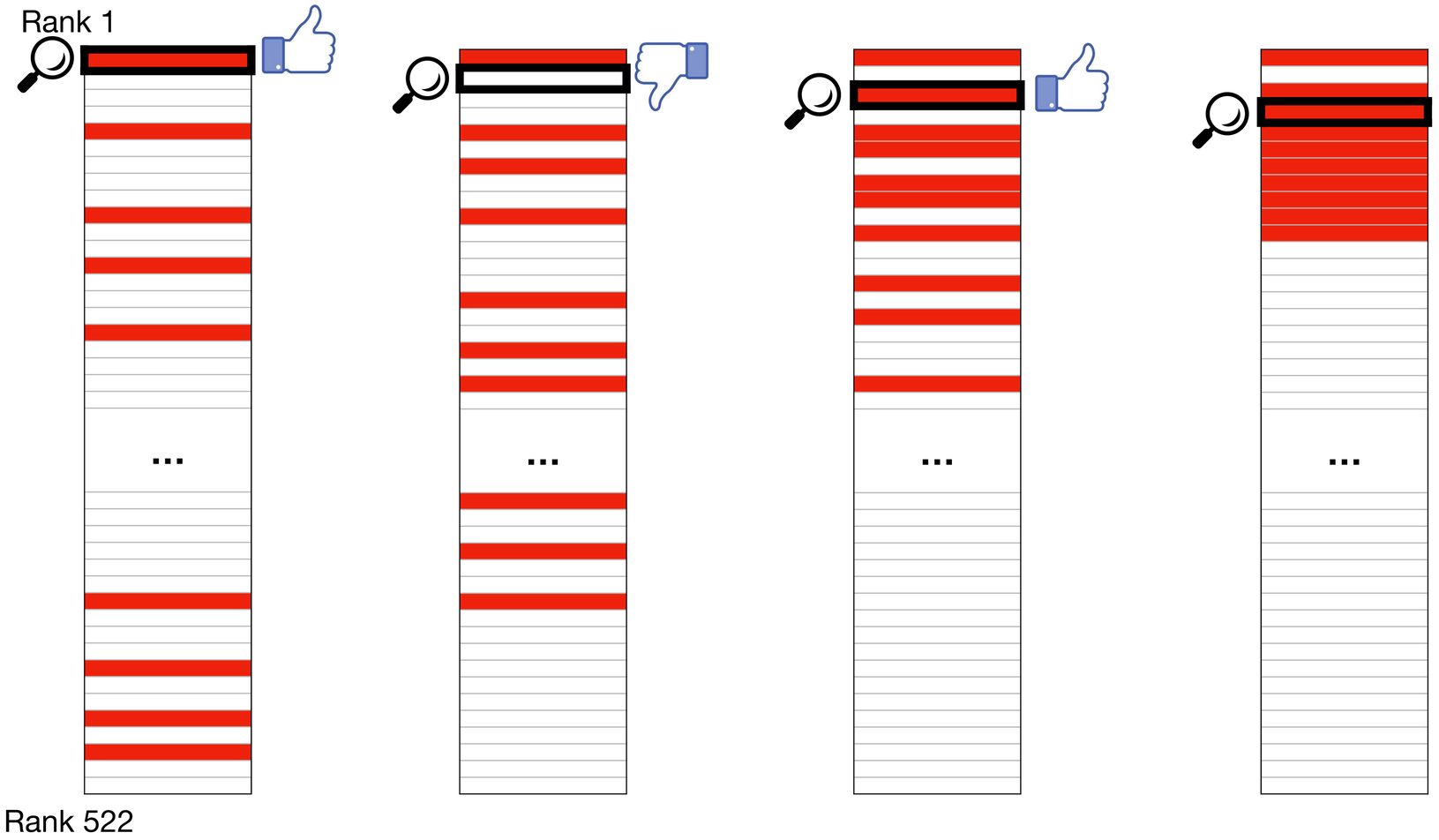
Bug

False Alarm

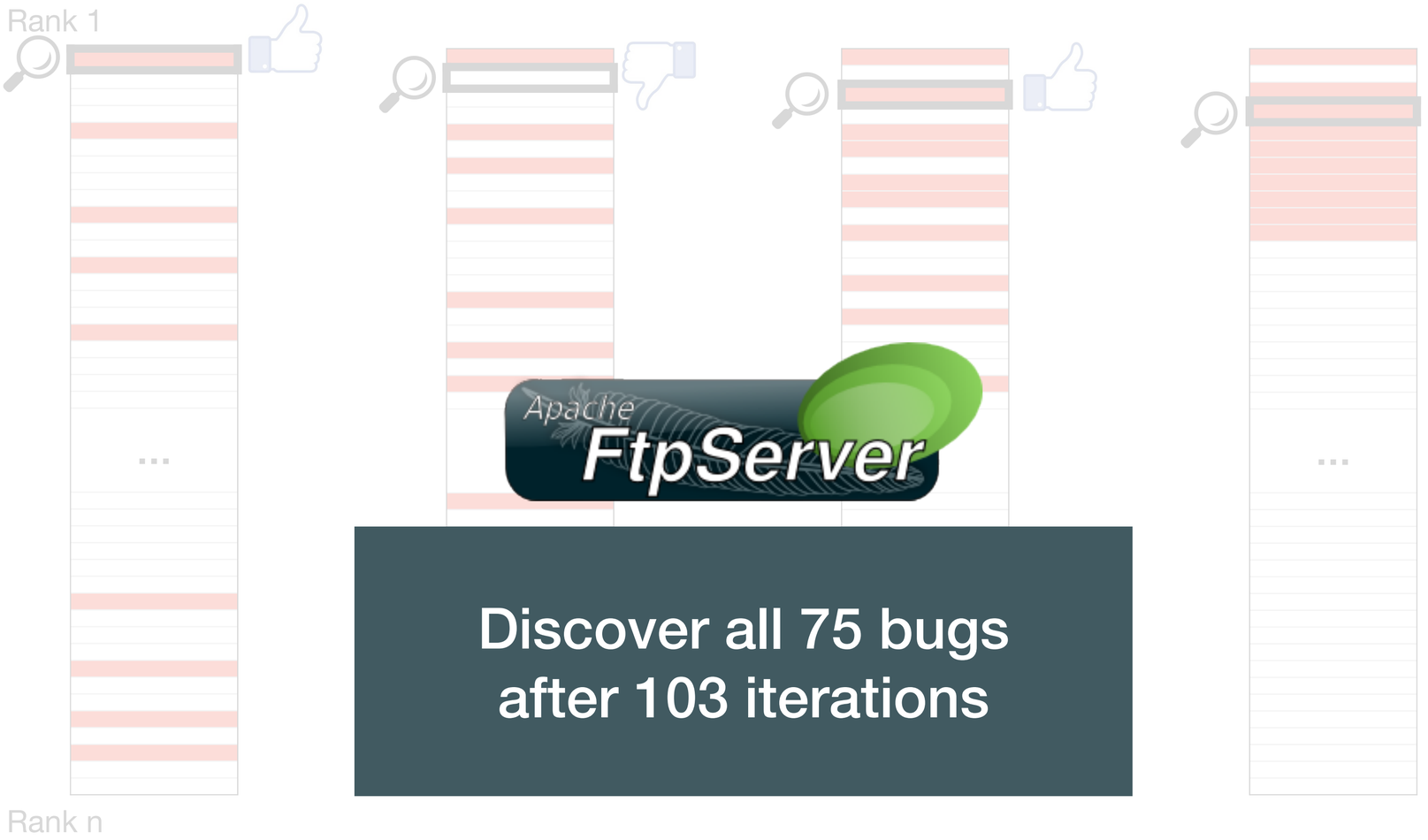
반응형 오류 보고


Rank 1
...
...
...
Rank 522
Apache
FtpServer
152KLOC
75 Datarace Bugs
522 Total Alarms
Bug
False Alarm

반응형 오류 보고

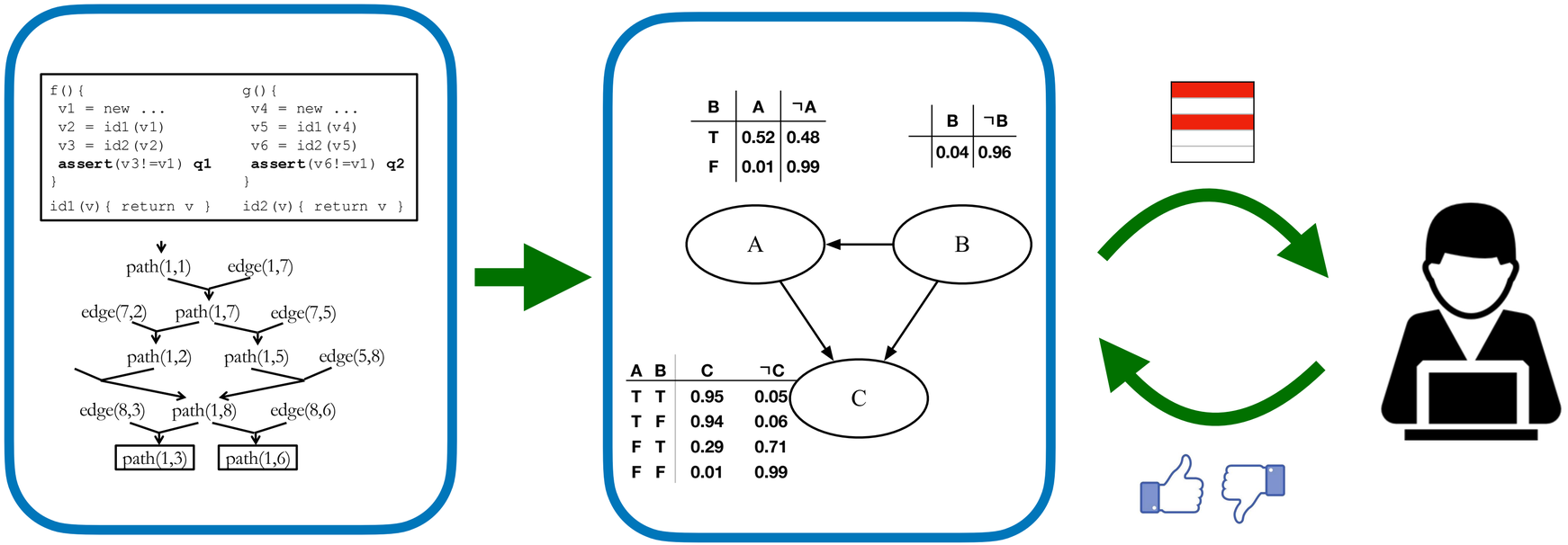


반응형 오류 보고


Rank 1
...
Rank n
Apache
FtpServer
...
Discover all 75 bugs
after 103 iterations

핵심 기술

인간-분석 상호 작용 + 베이지안 추론 (Bayesian inference)



분석 결과

확률 모델
(e.g., Bayesian Network)

사용자

분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

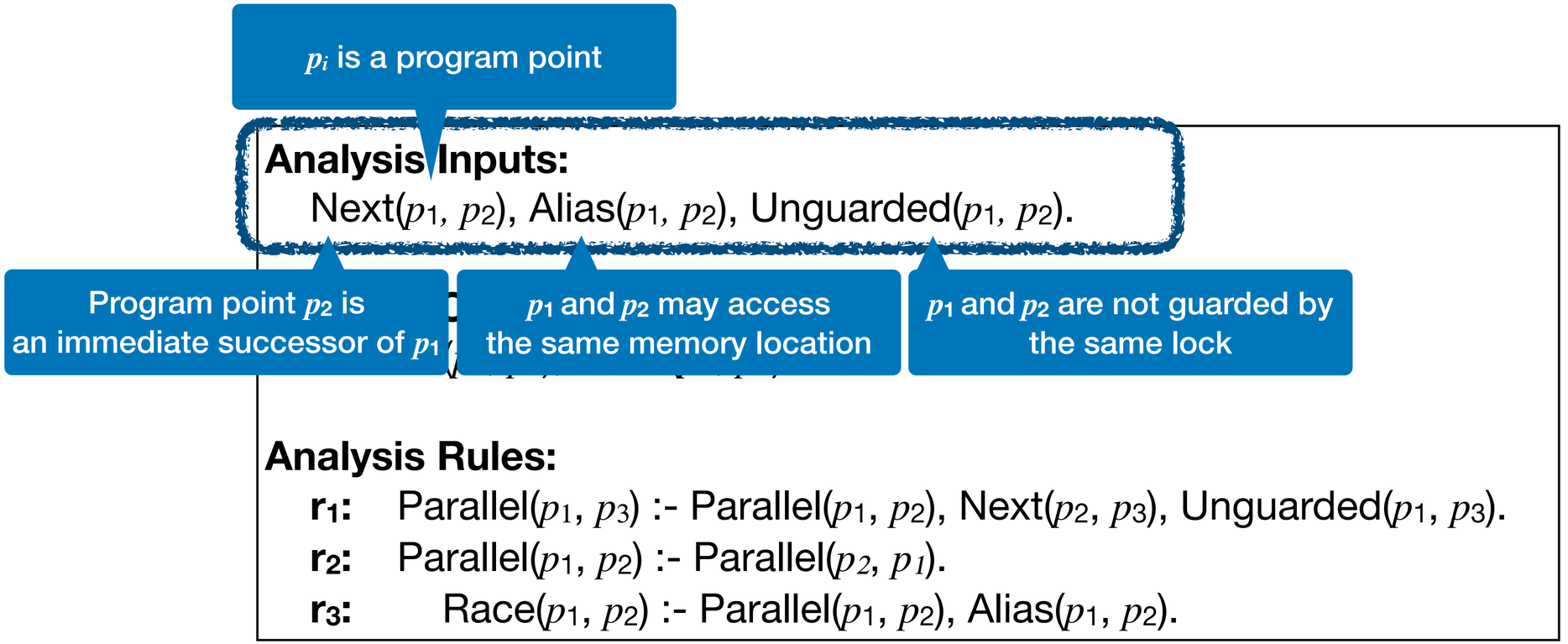
Analysis Rules:

r_1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r_2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r_3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

분석 예제: Datarace



분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

Program point p_1 and p_2 can be executed in parallel

Race condition between p_1 and p_2

r1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

Analysis Rules:

r_1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r_2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r_3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

Thread 1

```
…
x = y + 1; // L1
…
```

Thread 2

```
…
z = y + 1; // L2
x = z + 1; // L3
…
```

분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

Analysis Rules:

r_1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r_2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r_3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

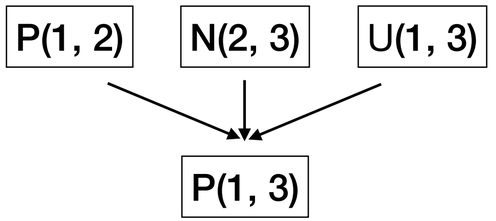
Thread 1

```
…
x = y + 1; // L1
…
```

Thread 2

```
…
z = y + 1; // L2
x = z + 1; // L3
…
```

Derivation



분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

Analysis Rules:

r_1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r_2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r_3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

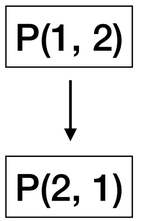
Thread 1

```
…
x = y + 1; // L1
…
```

Thread 2

```
…
z = y + 1; // L2
x = z + 1; // L3
…
```

Derivation



분석 예제: Datarace

Analysis Inputs:

Next(p_1, p_2), Alias(p_1, p_2), Unguarded(p_1, p_2).

Analysis Outputs:

Parallel(p_1, p_2), Race(p_1, p_2)

Analysis Rules:

r_1: Parallel(p_1, p_3) :- Parallel(p_1, p_2), Next(p_2, p_3), Unguarded(p_1, p_3).

r_2: Parallel(p_1, p_2) :- Parallel(p_2, p_1).

r_3: Race(p_1, p_2) :- Parallel(p_1, p_2), Alias(p_1, p_2).

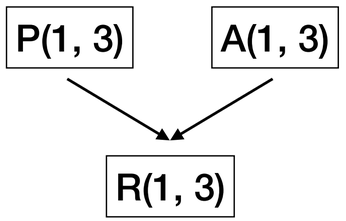
Thread 1

```
…
x = y + 1; // L1
…
```

Thread 2

```
…
z = y + 1; // L2
x = z + 1; // L3
…
```

Derivation



예제 프로그램

```
public class RequestHandler {
 private FtpRequest request;

 public FtpRequest getRequest() {
  return request;                    //L0
 }

 public void close() {
  synchronized (this) {              //L1
   if (isClosed) return;             //L2
   isClosed = true;                  //L3
  }
  controlSocket.close();             //L4
  controlSocket = null;              //L5
  request.clear();                   //L6
  request = null;                    //L7
 }
}
```

Analysis Rules:

- **r1:** P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
- **r2:** P(p_1, p_2) :- P(p_2, p_1).
- **r3:** R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

***Apache FTP Server**

예제 프로그램

```
public class RequestHandler {
 private FtpRequest request;

 public FtpRequest getRequest() {
  return request;                    //L0
 }

 public void close() {
  synchronized (this) {              //L1
   if (isClosed) return;             //L2
   isClosed = true;                  //L3
  }
  controlSocket.close();             //L4
  controlSocket = null;              //L5
  request.clear();                   //L6
  request = null;                    //L7
 }
}
```

Analysis Rules:

- **r1:** P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
- **r2:** P(p_1, p_2) :- P(p_2, p_1).
- **r3:** R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Datarace

***Apache FTP Server**

예제 프로그램

```
public class RequestHandler {
 private FtpRequest request;

 public FtpRequest getRequest() {
  return request;                    //L0
 }

 public void close() {
  synchronized (this) {              //L1
   if (isClosed) return;             //L2
   isClosed = true;                  //L3
  }
  controlSocket.close();             //L4
  controlSocket = null;              //L5
  request.clear();                   //L6
  request = null;                    //L7
 }
}
```

Analysis Rules:

- **r1:** P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
- **r2:** P(p_1, p_2) :- P(p_2, p_1).
- **r3:** R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

False alarm

False alarm

***Apache FTP Server**

정적 분석

Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis Rules

r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

정적 분석

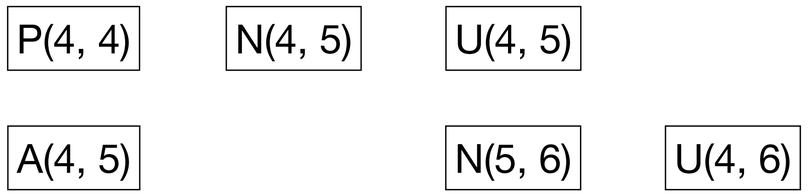
Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis Rules

r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Derivation Graph



1. 입력부터 시작

정적 분석

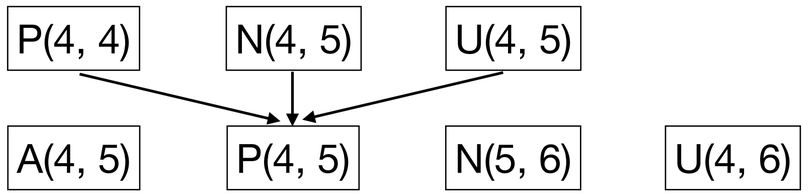
Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis Rules

r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Derivation Graph



2. 입력에 규칙을 적용

정적 분석

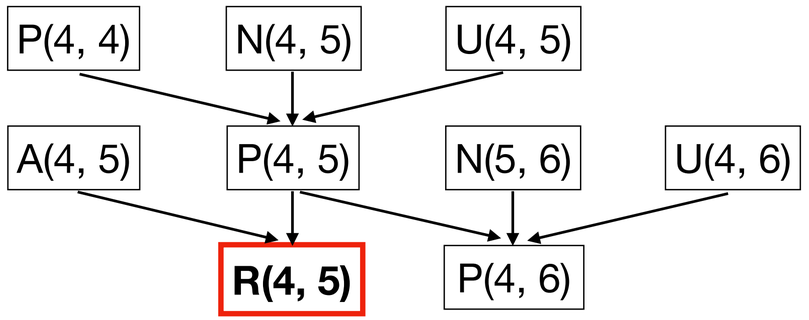
Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis Rules

r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Derivation Graph



3. 모든 중간결과에 다시 규칙을 적용

정적 분석

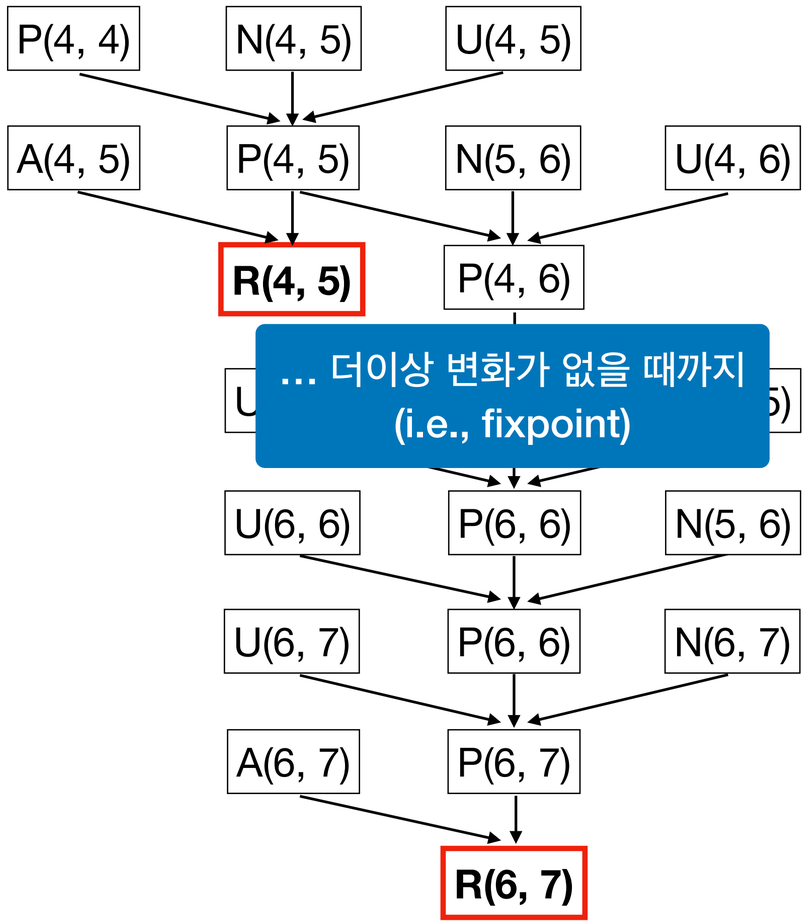
Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis Rules

r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Derivation Graph



정적 분석

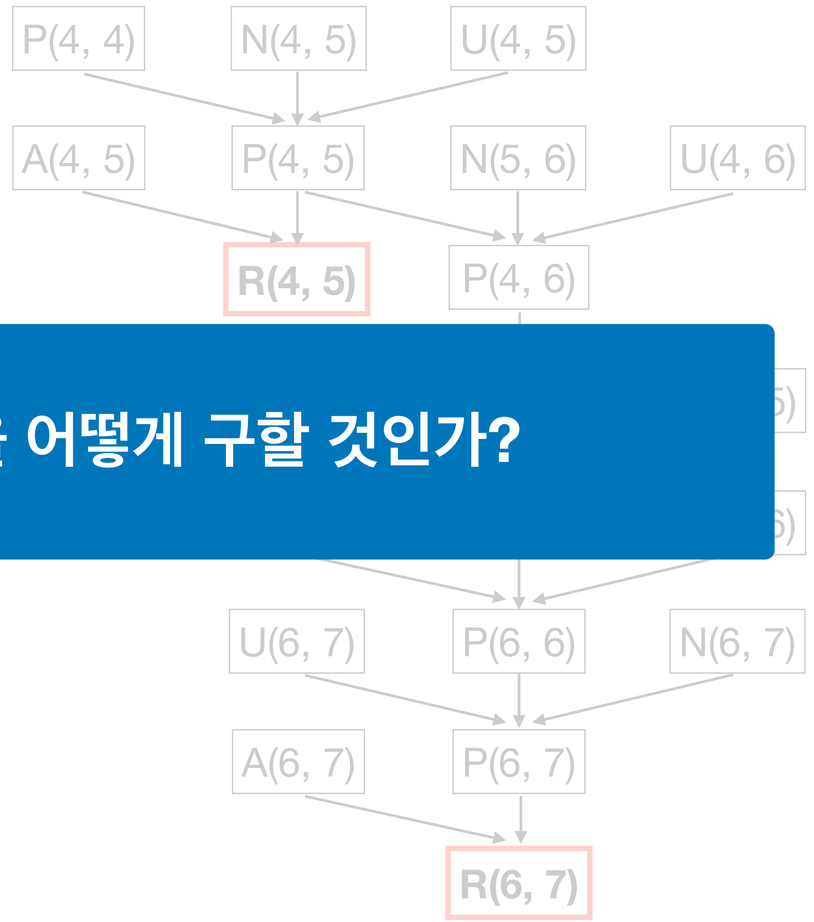
Program

```
controlSocket.close(); //L4
controlSocket = null;  //L5
request.clear();       //L6
request = null;        //L7
```

Analysis R

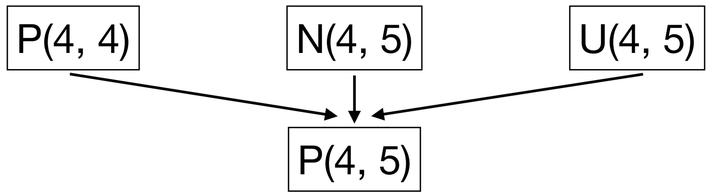
r1: P(p_1, p_3)
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Derivation Graph



Q: 경보의 확률을 어떻게 구할 것인가?

베이지안 네트워크



Logical Rule

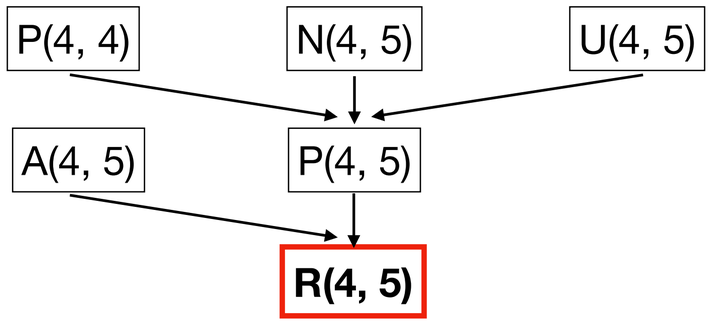
r1: P(p_1, p_3) :- P(p_1, p_2), N(p_2, p_3), U(p_1, p_3).
r2: P(p_1, p_2) :- P(p_2, p_1).
r3: R(p_1, p_2) :- P(p_1, p_2), A(p_1, p_2).

Probabilistic Rule

P(4,4)	N(4,5)	U(4,5)	*Pr*(P(4,5) | …)
TRUE	TRUE	TRUE	0.95*
TRUE	TRUE	FALSE	0
	…		
FALSE	FALSE	FALSE	0

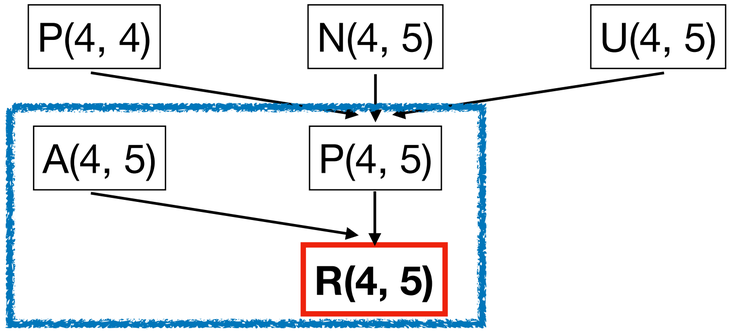
*computed by an offline learning

Marginal Inference



Pr(R(4,5)) =

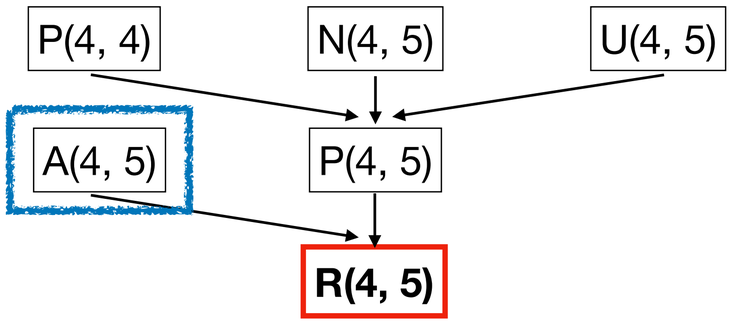
Marginal Inference



$$Pr(\text{R}(4,5)) = Pr(\text{R}(4,5) \mid \text{A}(4,5), \text{P}(4,5))$$

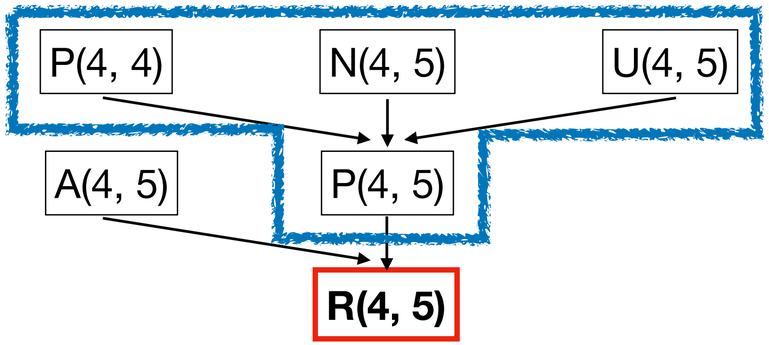
$\times$

Marginal Inference



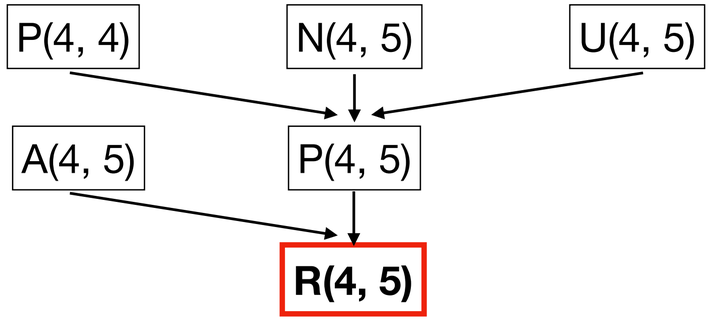
$$Pr(\text{R}(4,5)) = Pr(\text{R}(4,5) \mid \text{A}(4,5), \text{P}(4,5)) \times Pr(\text{A}(4,5)) \times$$

Marginal Inference



$$Pr(\text{R}(4,5)) = Pr(\text{R}(4,5) \mid \text{A}(4,5), \text{P}(4,5)) \times Pr(\text{A}(4,5)) \times Pr(\text{P}(4,5) \mid \ldots) \times$$

Marginal Inference



$$
\begin{aligned}
Pr(\text{R}(4,5)) &= Pr(\text{R}(4,5) \mid \text{A}(4,5), \text{P}(4,5)) \\
&\times Pr(\text{A}(4,5)) \\
&\times Pr(\text{P}(4,5) \mid \ldots) \\
&\times \ldots \\
&= 0.398
\end{aligned}
$$

by an off-the-shelf marginal inference solver

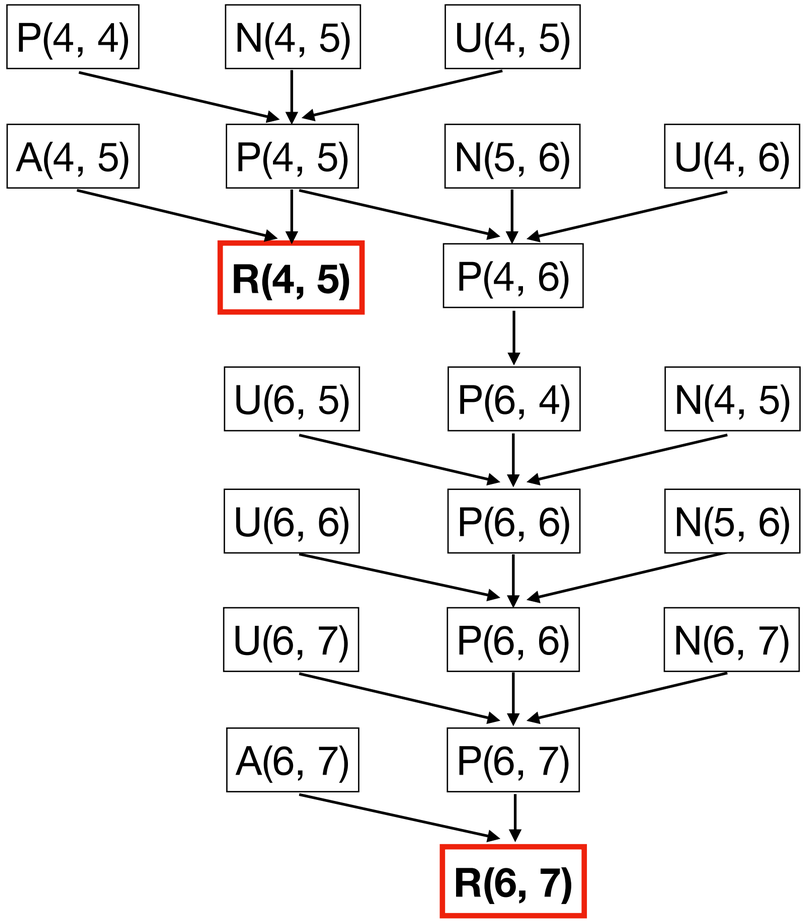
순위 기반 오류 보고서

```
public class RequestHandler {
 private FtpRequest request;

 public FtpRequest getRequest() {
  return request;                    //L0
 }

 public void close() {
  synchronized (this) {              //L1
   if (isClosed) return;             //L2
   isClosed = true;                  //L3
  }
  controlSocket.close();             //L4
  controlSocket = null;              //L5
  request.clear();                   //L6
  request = null;                    //L7
 }
}
```

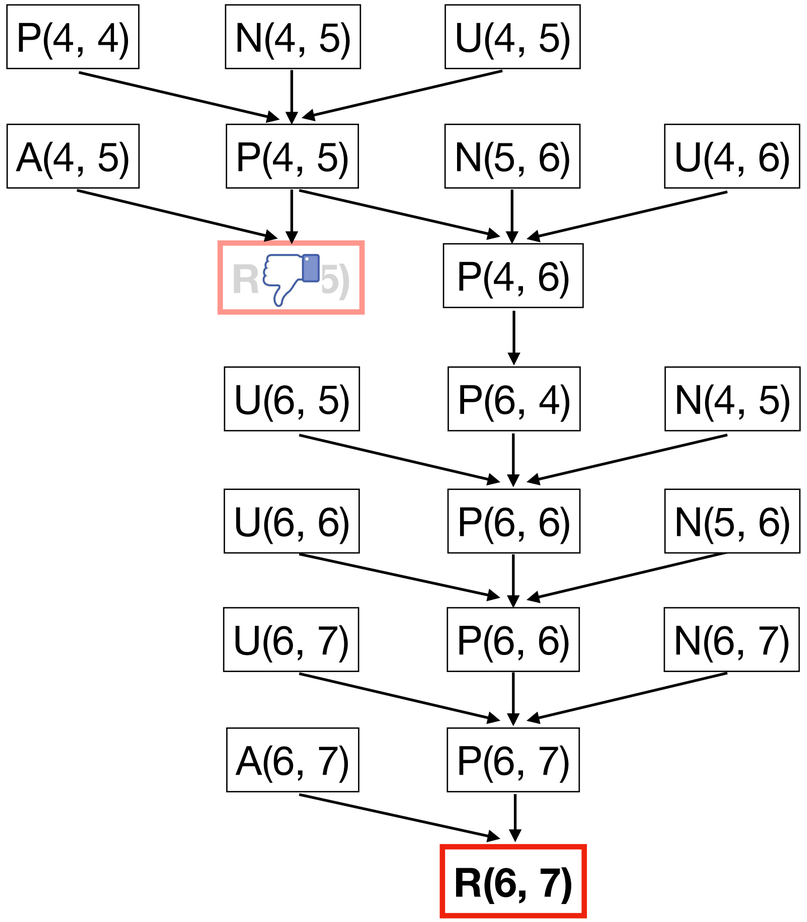
Ranking	Alarm	Confidence
1	R(4, 5)	0.398
2	R(5, 5)	0.378
3	R(6, 7)	0.324
4	R(7, 7)	0.308
5	R(0, 7)	0.279

Q: What are the probabilities of the other alarms when R(4,5) is false?

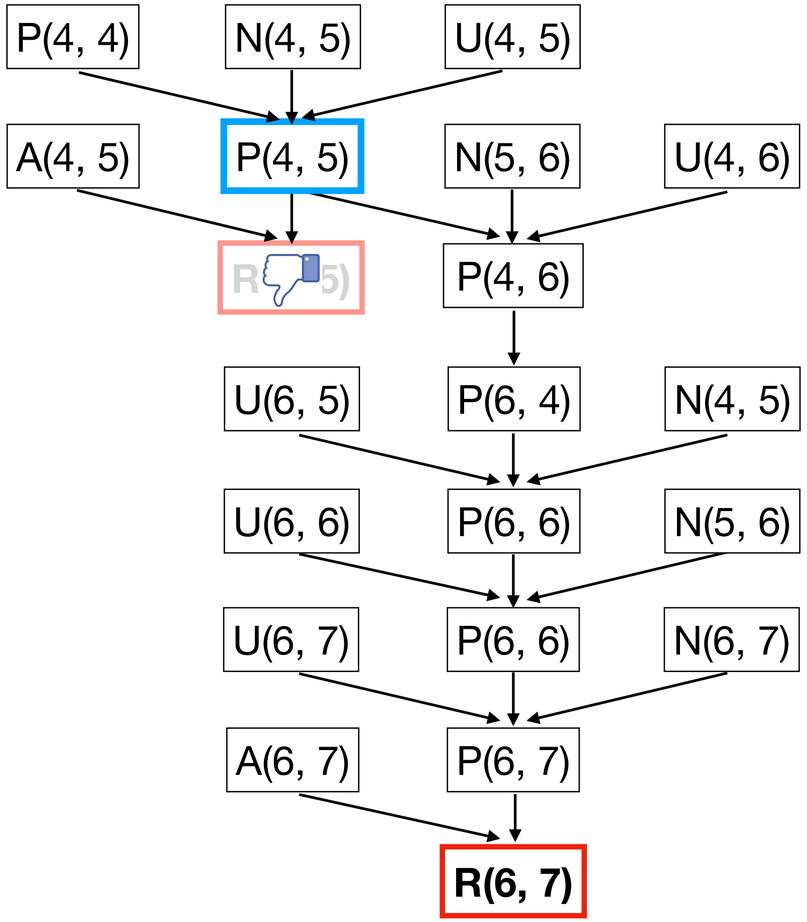
경보의 확률


P(4, 4)
N(4, 5)
U(4, 5)
A(4, 5)
P(4, 5)
N(5, 6)
U(4, 6)
R(4, 5)
P(4, 6)
U(6, 5)
P(6, 4)
N(4, 5)
U(6, 6)
P(6, 6)
N(5, 6)
U(6, 7)
P(6, 6)
N(6, 7)
A(6, 7)
P(6, 7)
R(6, 7)

경보의 확률

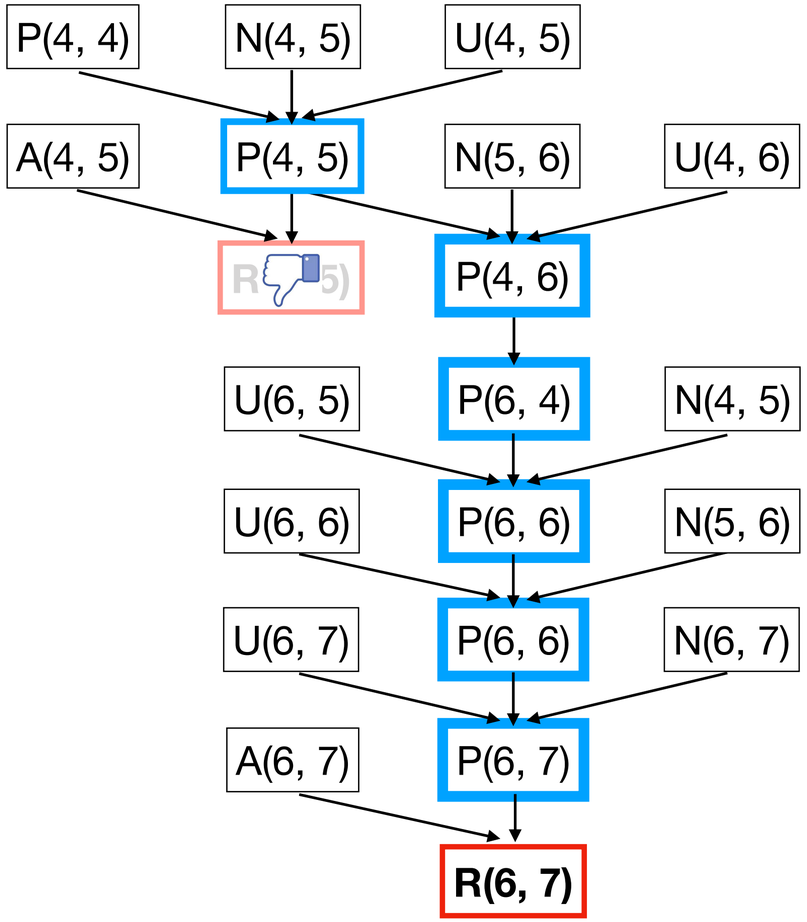

P(4, 4)
N(4, 5)
U(4, 5)
A(4, 5)
P(4, 5)
N(5, 6)
U(4, 6)
P(4, 6)
U(6, 5)
P(6, 4)
N(4, 5)
U(6, 6)
P(6, 6)
N(5, 6)
U(6, 7)
P(6, 6)
N(6, 7)
A(6, 7)
P(6, 7)
R(6, 7)

경보의 확률



Pr(P(4,5) | ¬R(4,5))
= *Pr*(¬R(4,5) | P(4,5)) *
Pr(P(4,5)) / *Pr*(¬R(4,5))
= 0.03

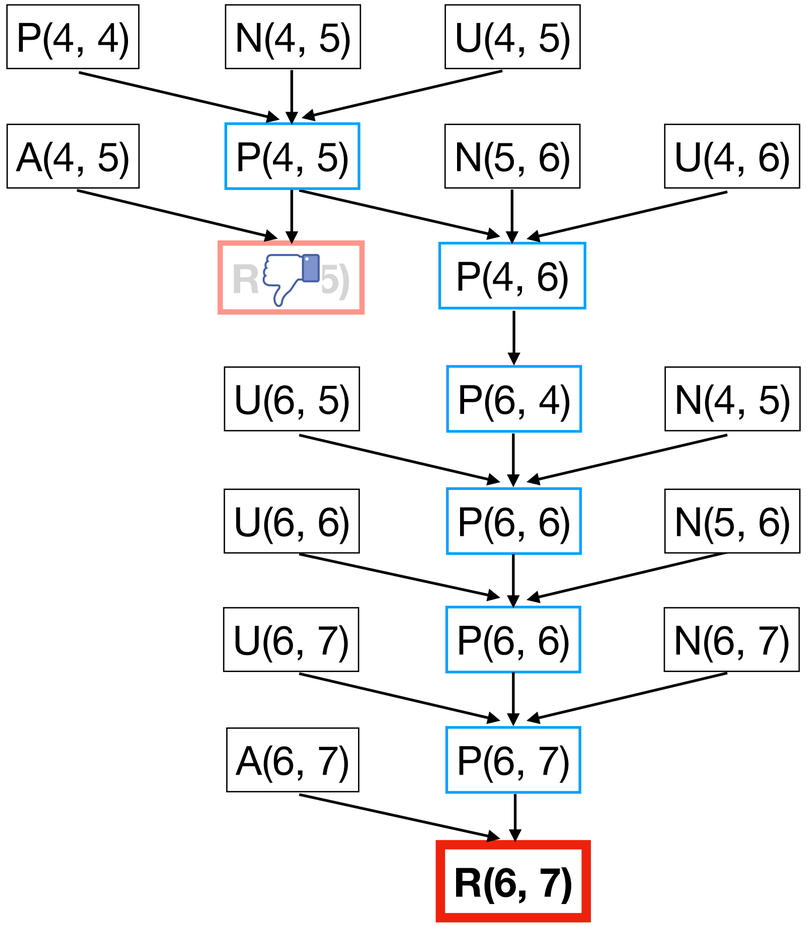
경보의 확률



Pr(P(4,5) | ¬R(4,5))
= *Pr*(¬R(4,5) | P(4,5)) *
Pr(P(4,5)) / *Pr*(¬R(4,5))
= 0.03

By Bayes's Rule:
Pr(A|B) = *Pr*(B|A) * *Pr*(A) / *Pr*(B)

경보의 확률



Pr(P(4,5) | ¬R(4,5))
= *Pr*(¬R(4,5) | P(4,5)) *
Pr(P(4,5)) / *Pr*(¬R(4,5))
= 0.03

By Bayes's Rule:
Pr(A|B) = ***Pr***(B|A) * ***Pr***(A) / ***Pr***(B)

Pr(R(6,7) | ¬R(4,5))
= *Pr*(R(6,7) | P(4,5)) *
Pr(P(4,5)) | ¬R(4,5))
= 0.03

개선된 순위

Ranking	Alarm	Confidence
1	R(4, 5)	0.398
2	R(5, 5)	0.378
3	R(6, 7)	0.324
4	R(7, 7)	0.308
5	R(0, 7)	0.279

개선된 순위

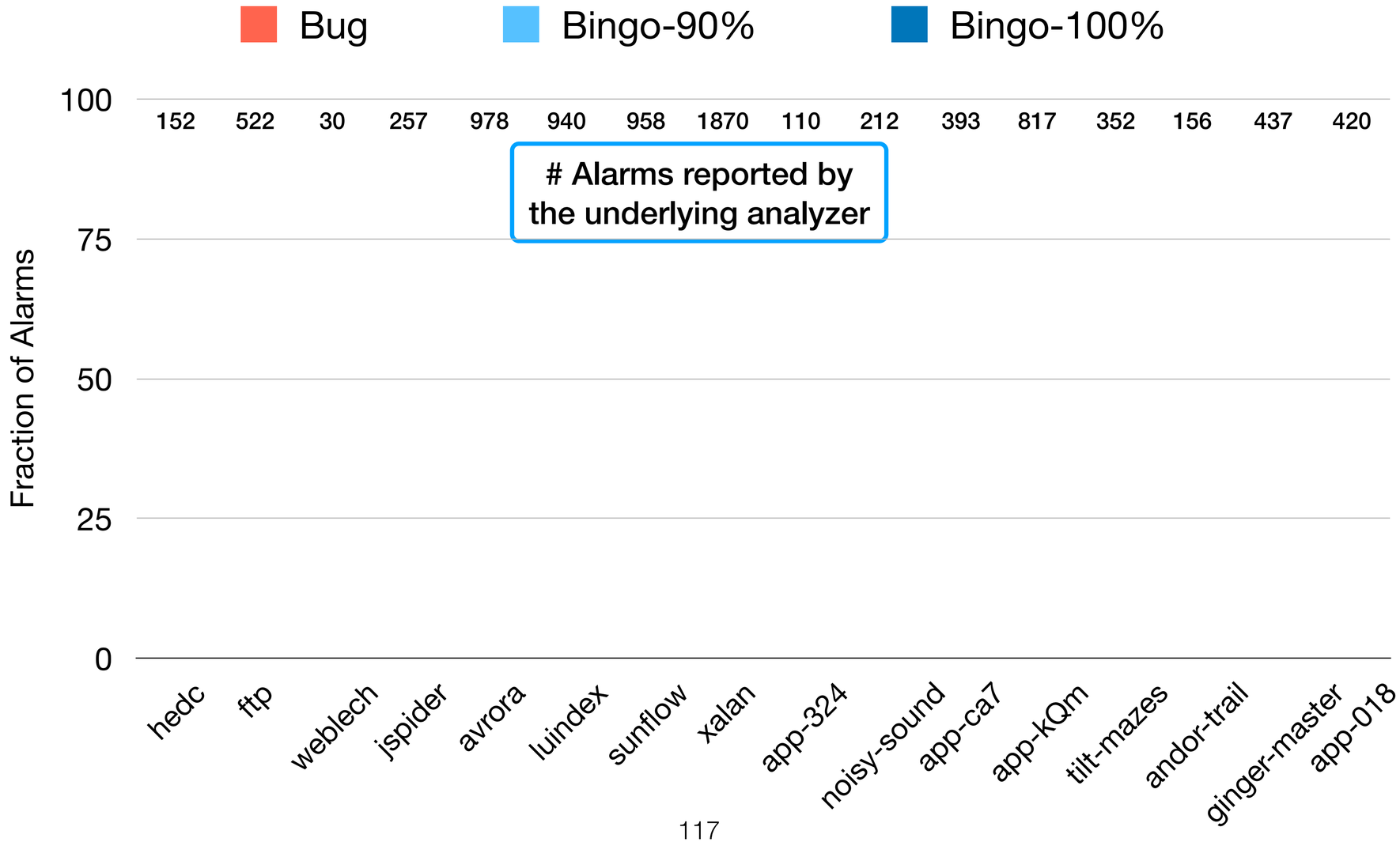
Ranking	Alarm	Confidence
1	R(0, 7)	0.279
2	R(5, 5)	0.035
3	R(6, 7)	0.030
4	R(7, 7)	0.028
5	R(4, 5)	0

효과

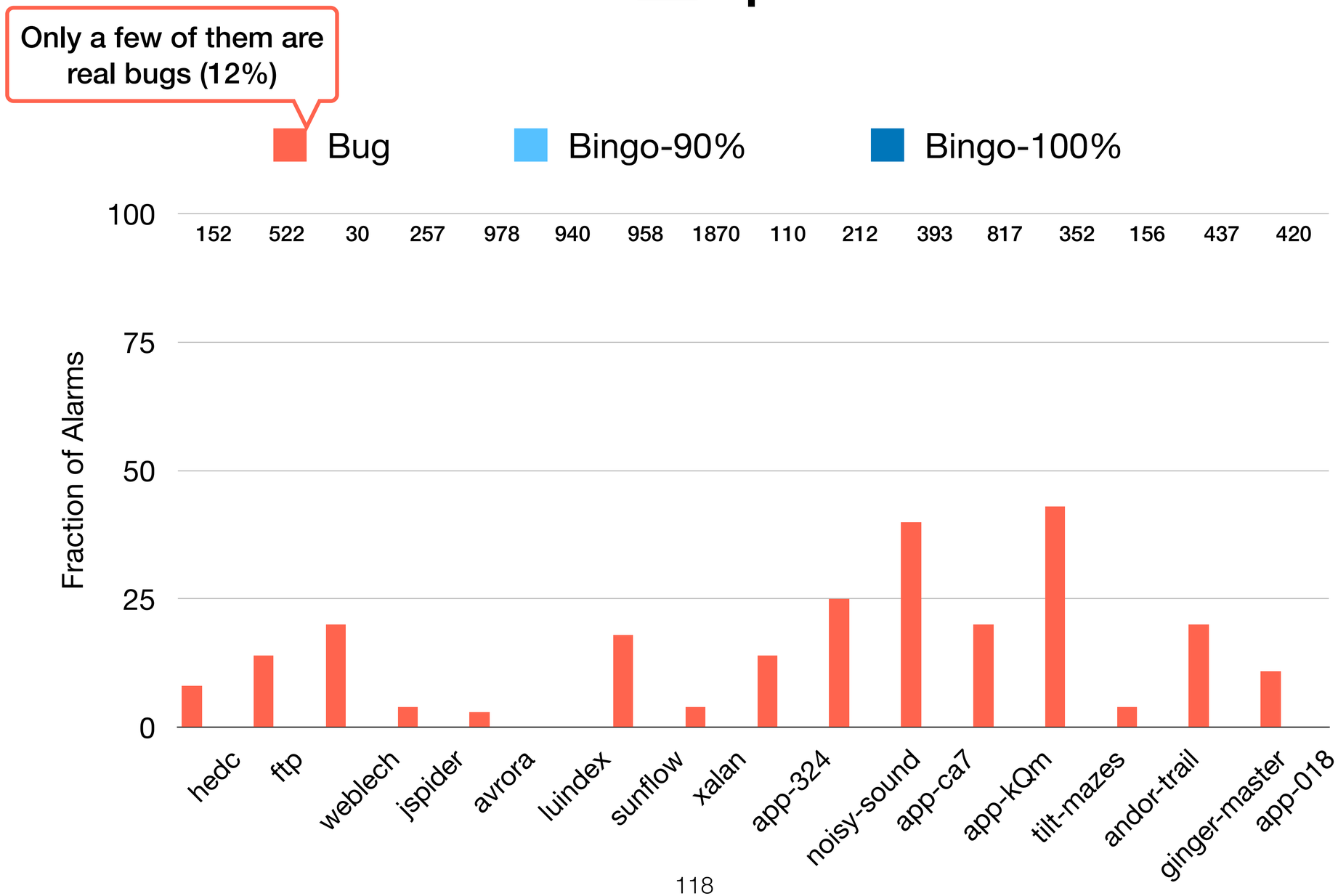
40—616KLOC JAVA Programs
Datarace and Privacy leak analyses

hedc
ftp
weblech
jspider
avrora
luindex
sunflow
xalan
app-324
noisy-sound
app-ca7
app-kQm
tilt-mazes
andor-trail
ginger-master
app-018

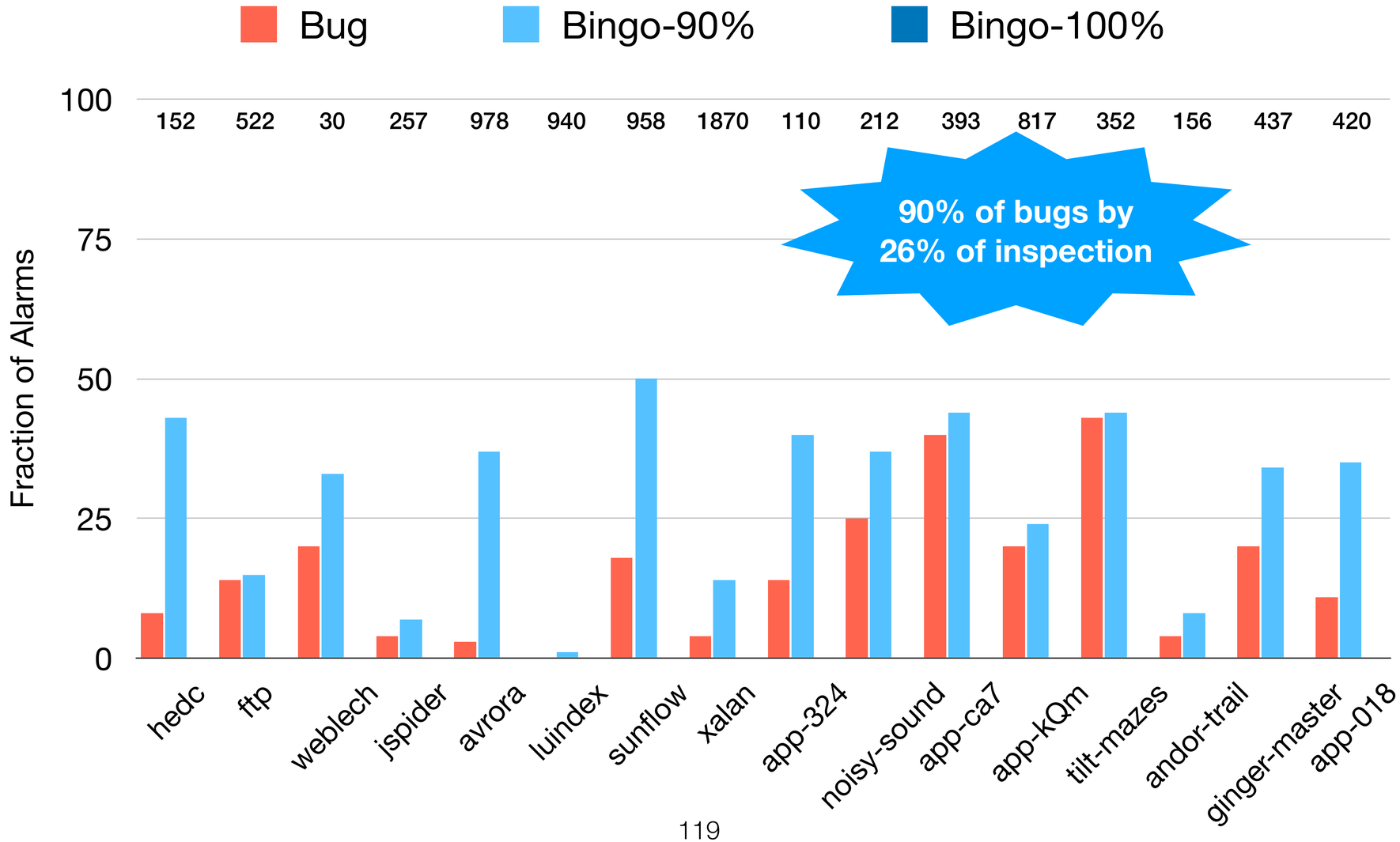
효과


Bug
Bingo-90%
Bingo-100%
Fraction of Alarms
100
75
50
25
0
152
522
30
257
978
940
958
1870
110
212
393
817
352
156
437
420
Alarms reported by the underlying analyzer
hedc
ftp
weblech
jspider
avrora
luindex
sunflow
xalan
app-324
noisy-sound
app-ca7
app-kQm
tilt-mazes
andor-trail
ginger-master
app-018

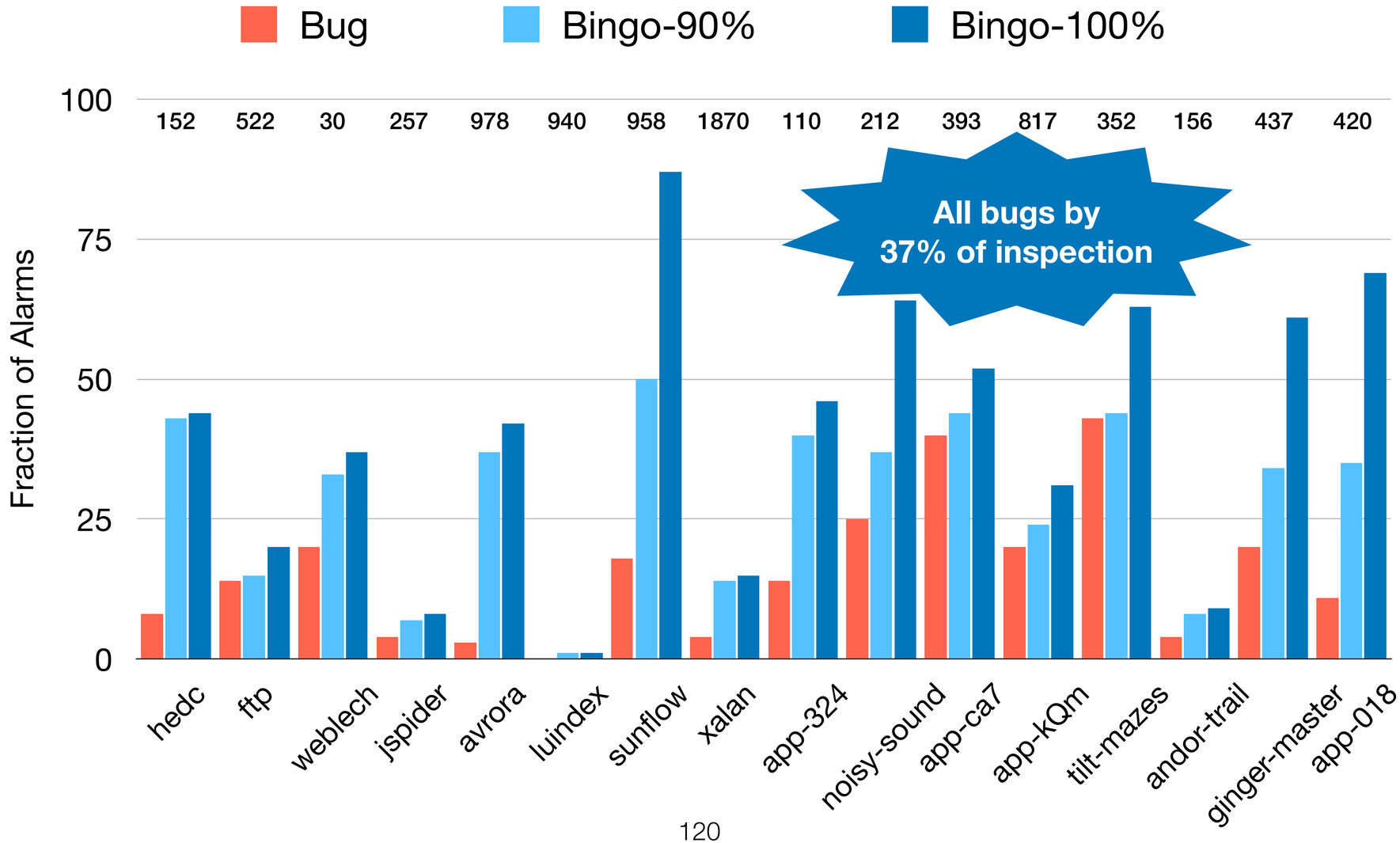
효과


Only a few of them are real bugs (12%)
Bug
Bingo-90%
Bingo-100%
Fraction of Alarms
100
75
50
25
0
152
522
30
257
978
940
958
1870
110
212
393
817
352
156
437
420
hedc
ftp
weblech
jspider
avrora
luindex
sunflow
xalan
app-324
noisy-sound
app-ca7
app-kQm
tilt-mazes
andor-trail
ginger-master
app-018

효과


Bug
Bingo-90%
Bingo-100%
Fraction of Alarms
100
75
50
25
0
152
522
30
257
978
940
958
1870
110
212
393
817
352
156
437
420
90% of bugs by
26% of inspection
hedc
ftp
weblech
jspider
avrora
luindex
sunflow
xalan
app-324
noisy-sound
app-ca7
app-kQm
tilt-mazes
andor-trail
ginger-master
app-018

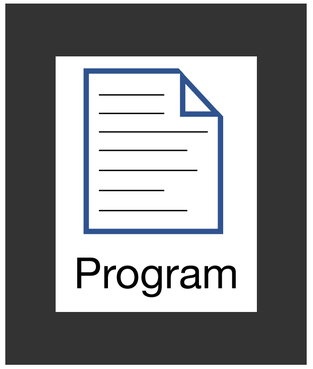
효과


Bug
Bingo-90%
Bingo-100%
Fraction of Alarms
100
75
50
25
0
152
522
30
257
978
940
958
1870
110
212
393
817
352
156
437
420
All bugs by
37% of inspection
hedc
ftp
weblech
jspider
avrora
luindex
sunflow
xalan
app-324
noisy-sound
app-ca7
app-kQm
tilt-mazes
andor-trail
ginger-master
app-018

Outline

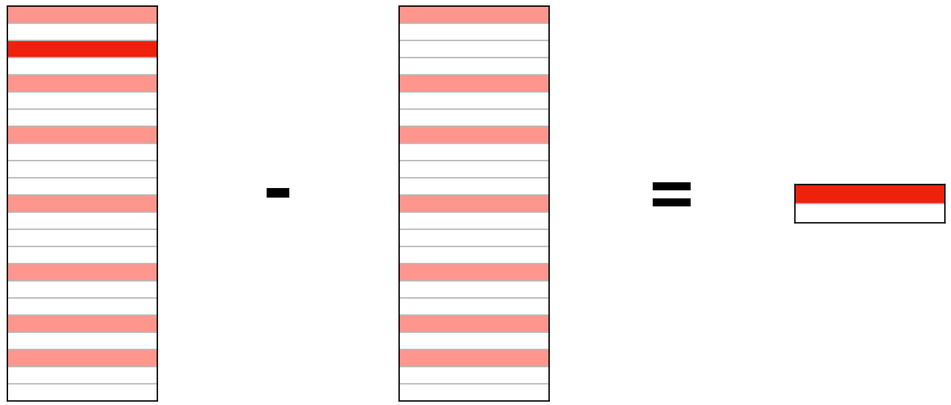
Adaptive Program Analysis

Interactive Program Analysis



Continuous Program Analysis

Drake: Continuous Alarm Masking System [PLDI'19]

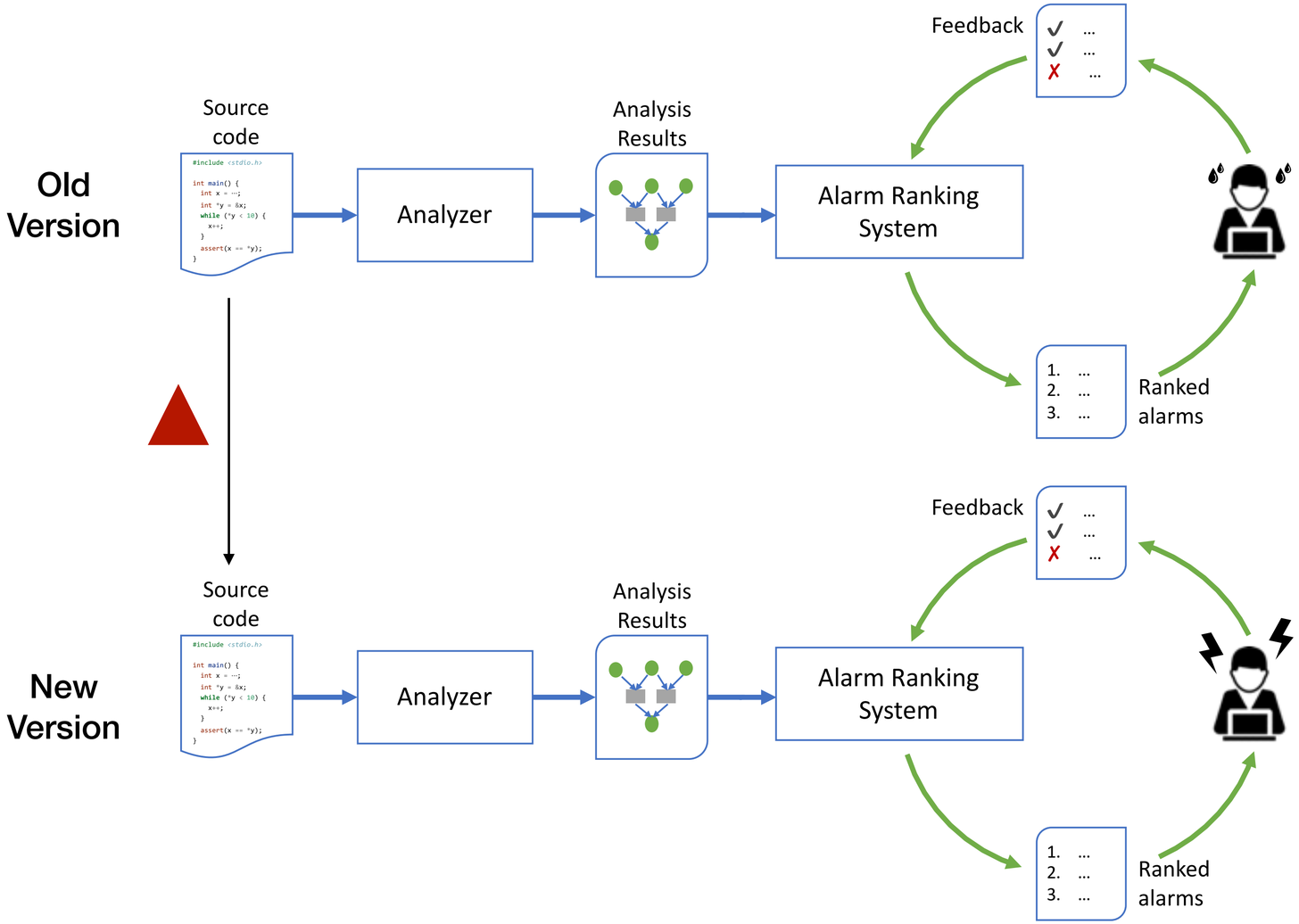


Alarms of New Version

Alarms of Old Version

Alarms Relevant to the Change

일괄형 (batch-mode) 분석



연속적 (continuous) 분석

"We only display results for most analyses on ***changed lines*** *by default; this keeps analysis results* ***relevant*** *to the code review at hand", - Google, 2015*

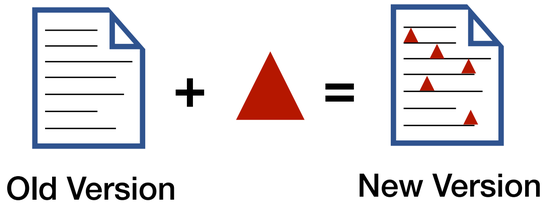
"The vast majority of Infer's impact to this point is attributable to ***continuous reasoning at diff time****", - Facebook, 2016*

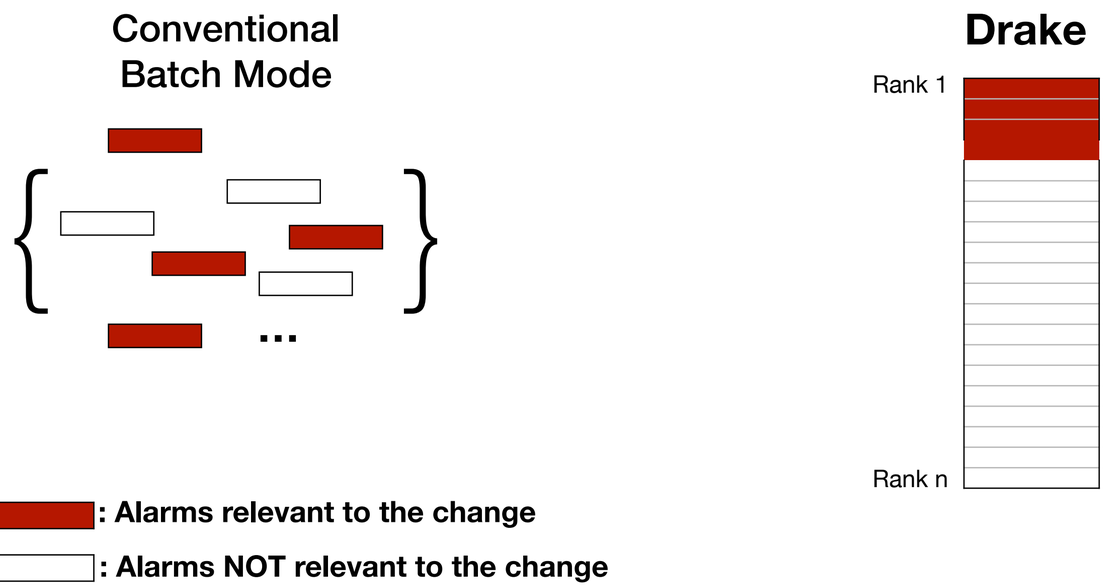
"… is the ability to analyze a ***changelist (a.k.a. a commit) rather than the entire codebase****. This functionality can help developers assess the quality and impact of a change …", - Microsoft, 2016*

"In order to realize the goal, verification must continue to work with low effort ***as developers change the code****. … Neither of these approaches would work for Amazon as s2n is under* ***continuous development****.", - Amazon, 2018*

목표

- 코드 변화와 관련이 있는 순으로 정렬해주는 오류 보고 시스템





예제

Old Version

```
1: x = input();
2: y = input();
3: x = opaque_dec(x);
4: y = opaque_dec(y);
5: x++;  // Alarm
6: y++;  // Alarm
```

x and y can be any integers

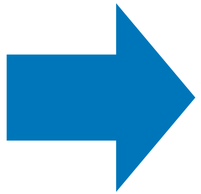
Opaque, but “--” actually

Integer overflow alarms at 5 & 6

예제

Old Version

```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
-4: y = opaque_dec(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```



New Version

```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```

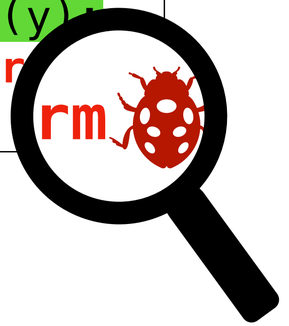
예제

Old Version

```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
-4: y = opaque_dec(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```

New Version

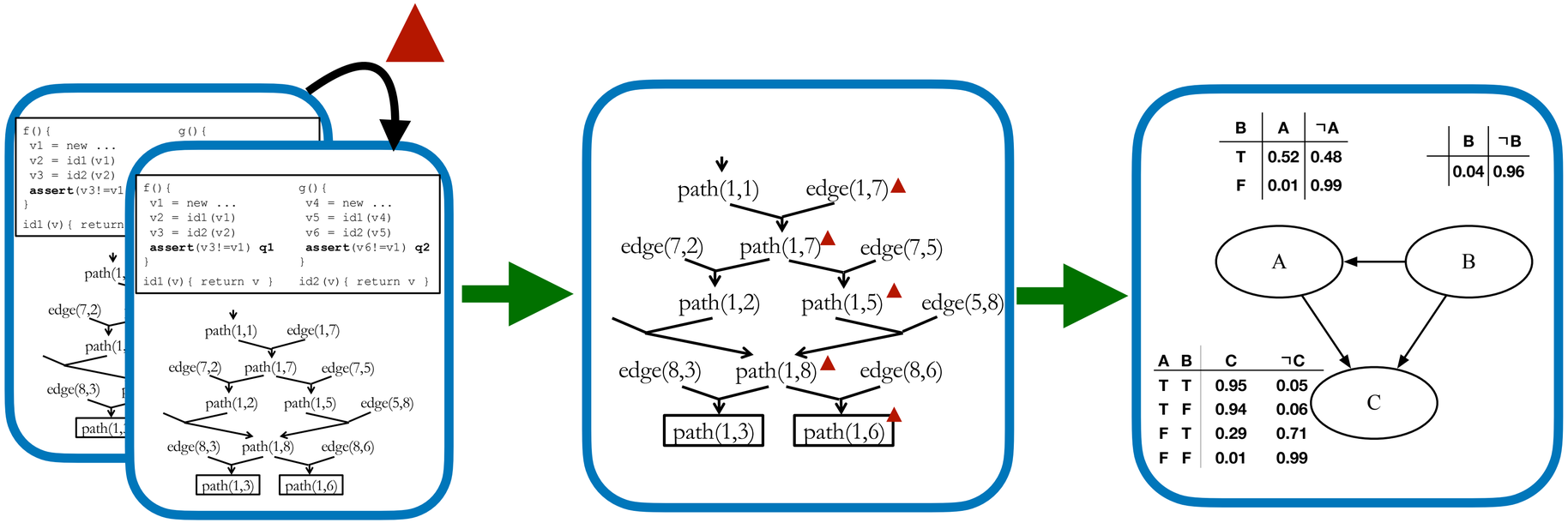
```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alar
 6: y++;  // Ala rm
```



Q: How to emphasize the alarm at 6 that is relevant to this change?

핵심 기술

변화 감지 분석 결과 + 베이지안 추론 (Bayesian inference)



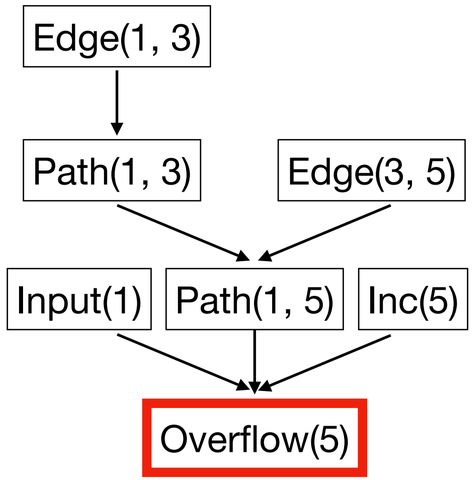
두 버전의 분석 결과

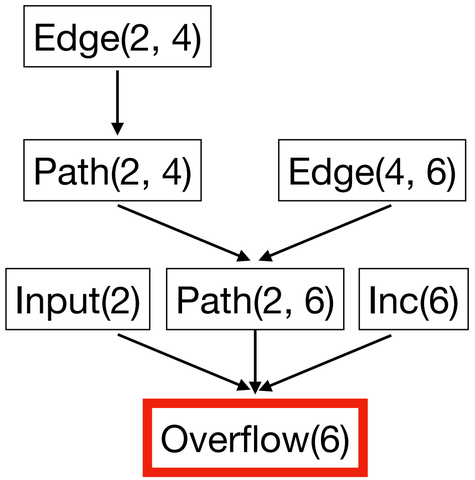
코드 변화를 감지하는 분석 결과

확률 모델
(Bayesian Network)

변화를 감지하는 분석 결과

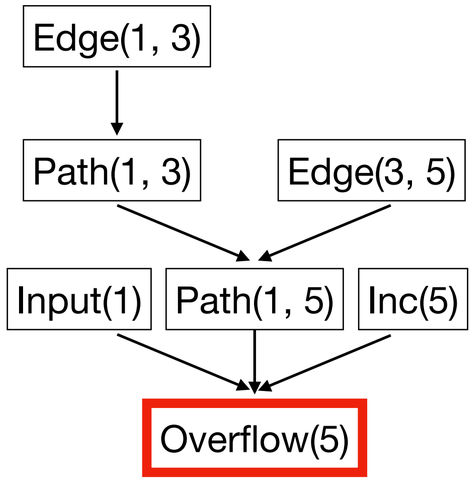
```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```

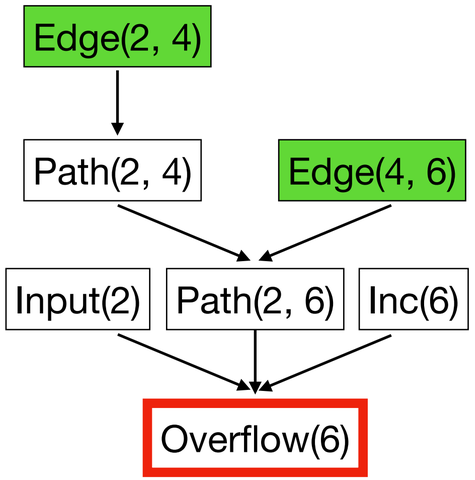




변화를 감지하는 분석 결과

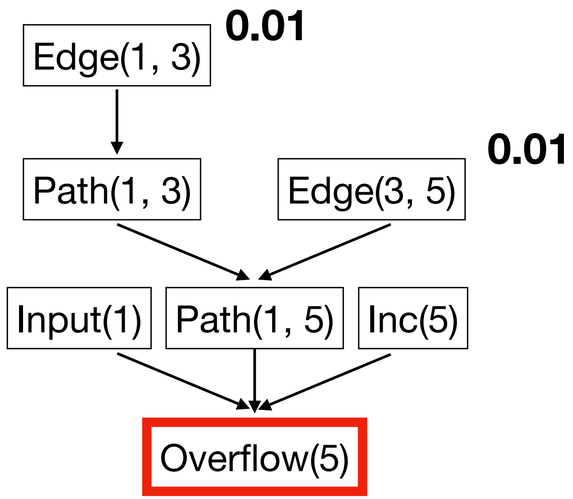
```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```

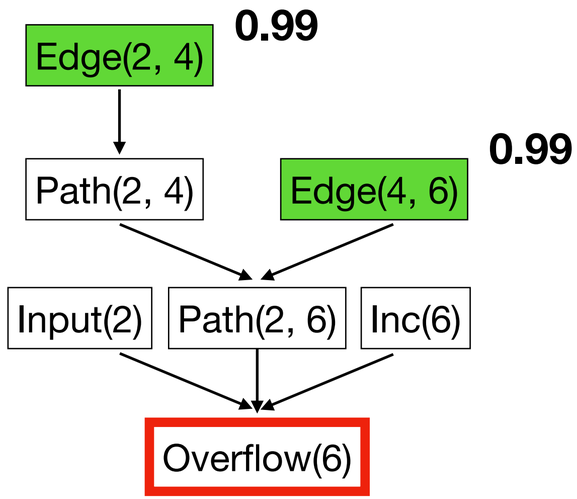




관련 점수

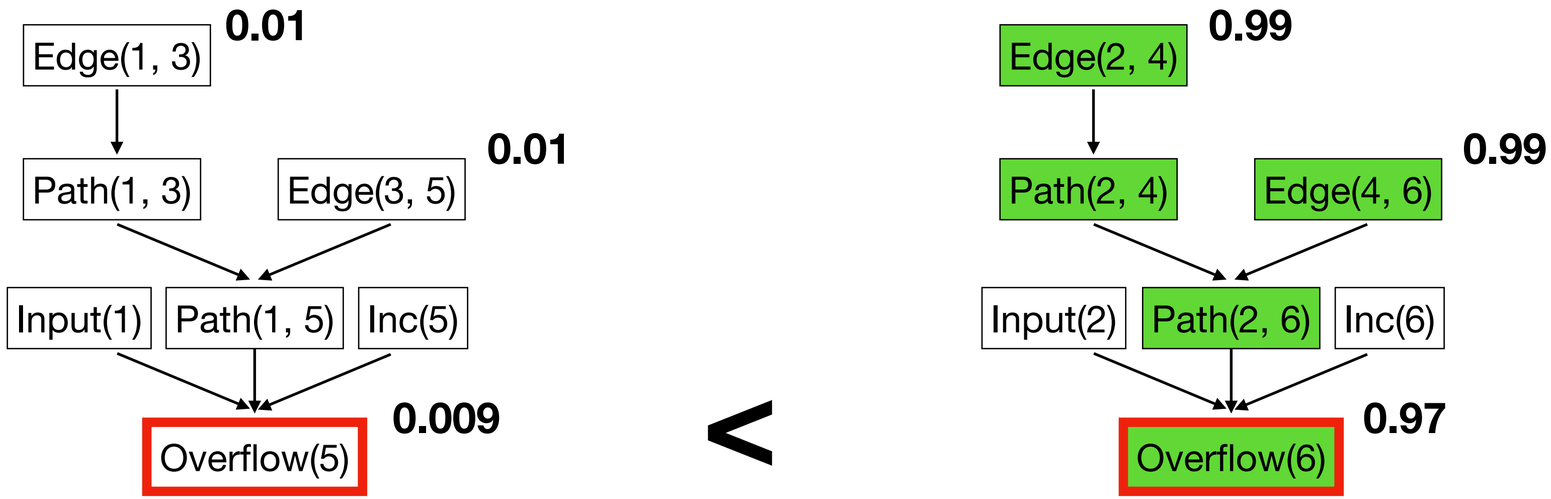
```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```





관련 점수

```
 1: x = input();
 2: y = input();
 3: x = opaque_dec(x);
+4: y = identity(y);
 5: x++;  // Alarm
 6: y++;  // Alarm
```

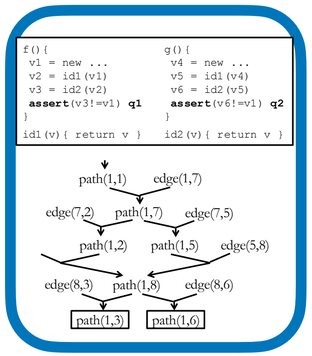


효과

13—112KLOC C Programs (old and new)
3 bugs on average
Buffer overrun and Integer overflow analyses

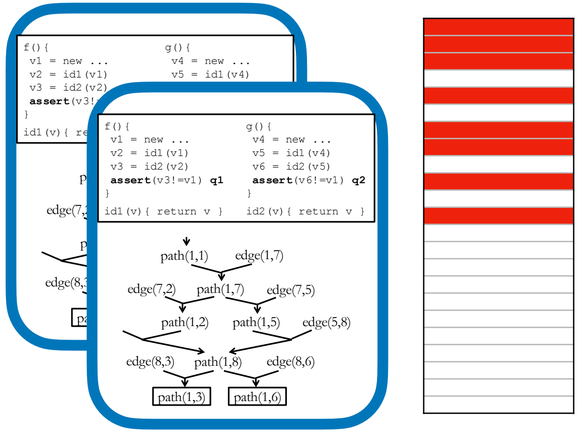
효과

평균 3개 버그를 찾기 위한 드는 노력의 평균을 측정



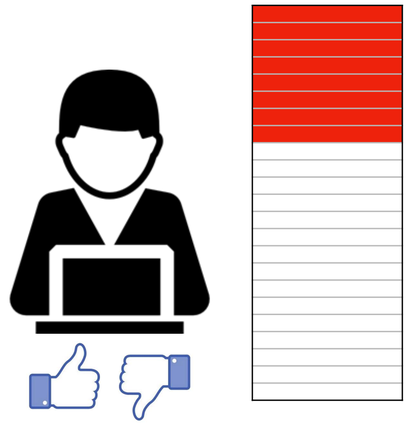
일괄형

563 개 경보 사이에
모든 버그가 산재



관련성 기반 순위

최대 순위 94 위 이내에
모든 버그 위치



+ 상호 작용 기반 순위

상호 작용 30 회 이내에
모든 버그 검출

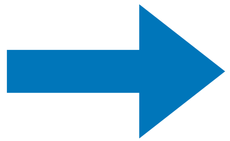
여러 응용

Pr(a | e)

문제	상호작용 대상	오류 검출에 드는 노력		확률 모델 학습 [ICSE'22]	오류 검출에 드는 노력
분석 결과 진위 여부	개발자	44% 감소	[PLDI'18]		22-33% 추가 감소
분석 결과 진위 여부	이전 버전	65% 감소	[PLDI'19]	→	54% 추가 감소
분석 결과 진위 여부	동적 분석	35% 감소	[FSE'21]		20% 추가 감소
…	…	…	…		…

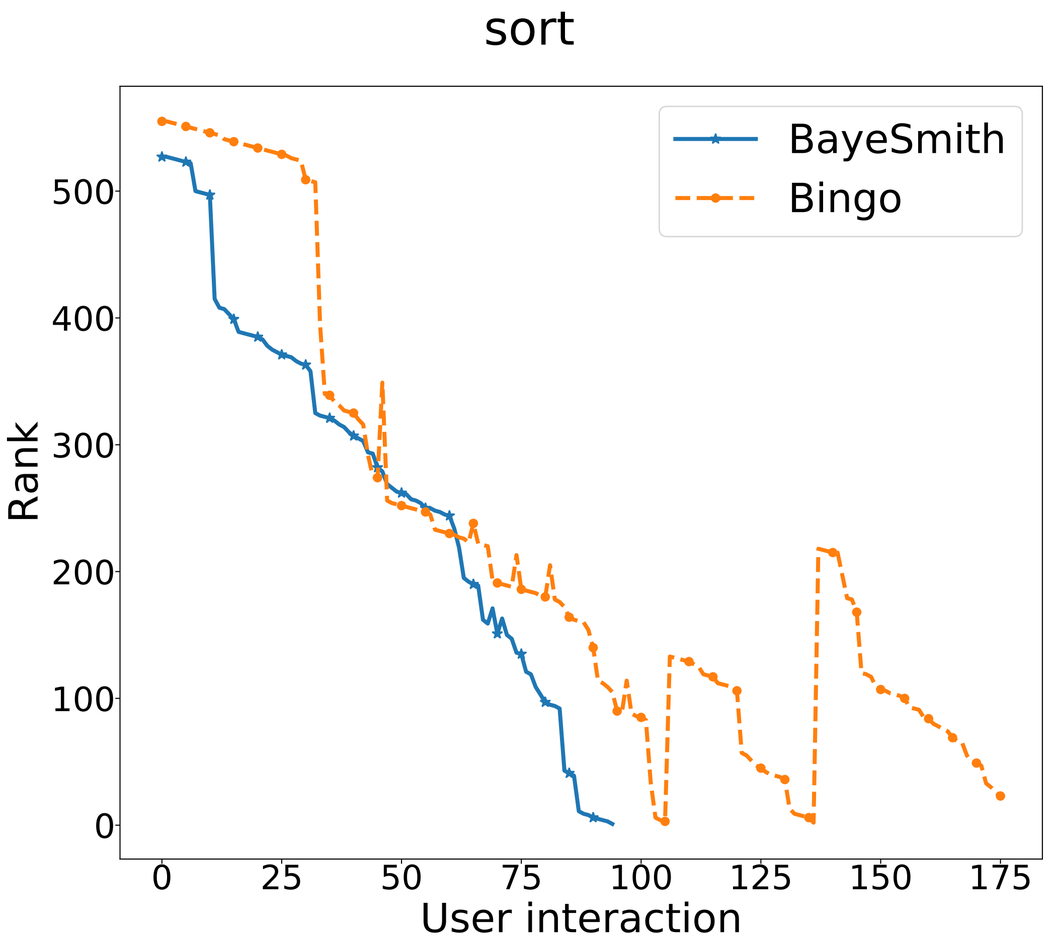
확률 모델 학습

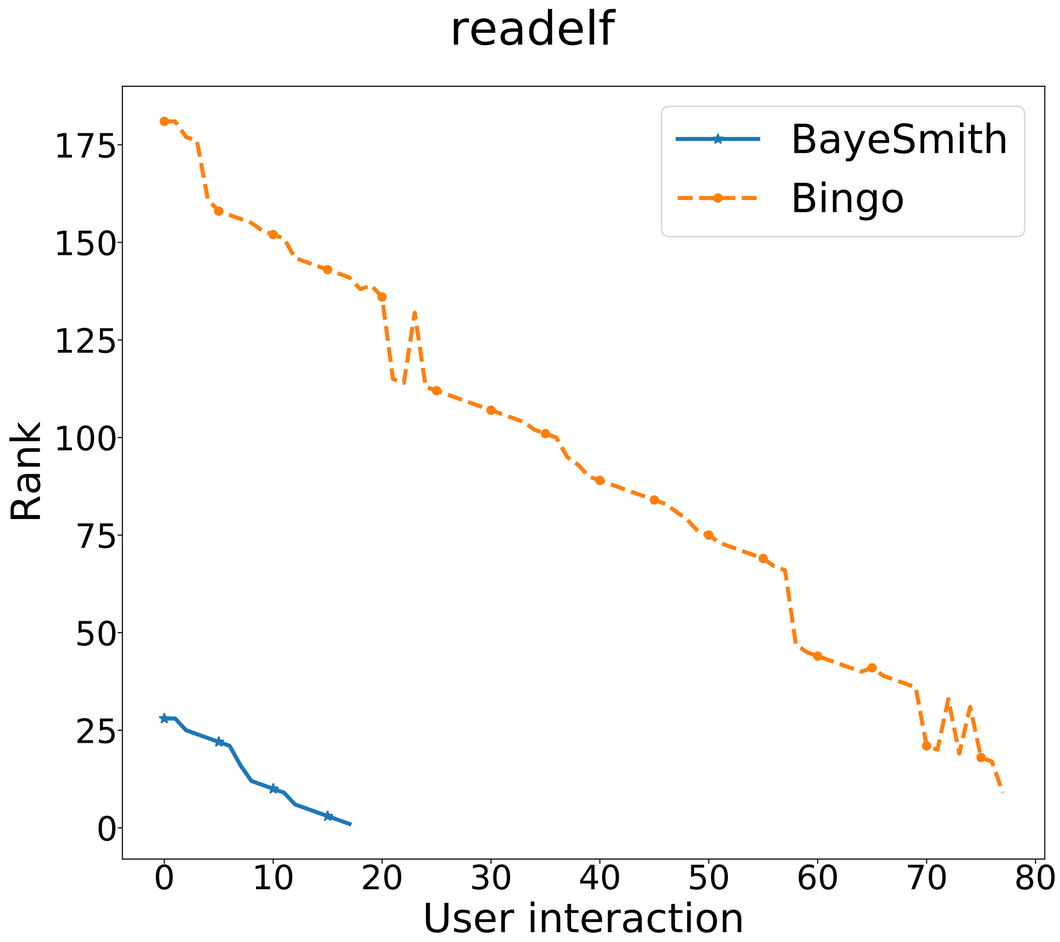
Ranking	Alarm	Confidence
1	LIne 6	0.93
2	Line 7	0.92
3	Line 8	0.91
...		



Ranking	Alarm	Confidence
...		
132	Line 7	0.41
133	Line 8	0.40
...		

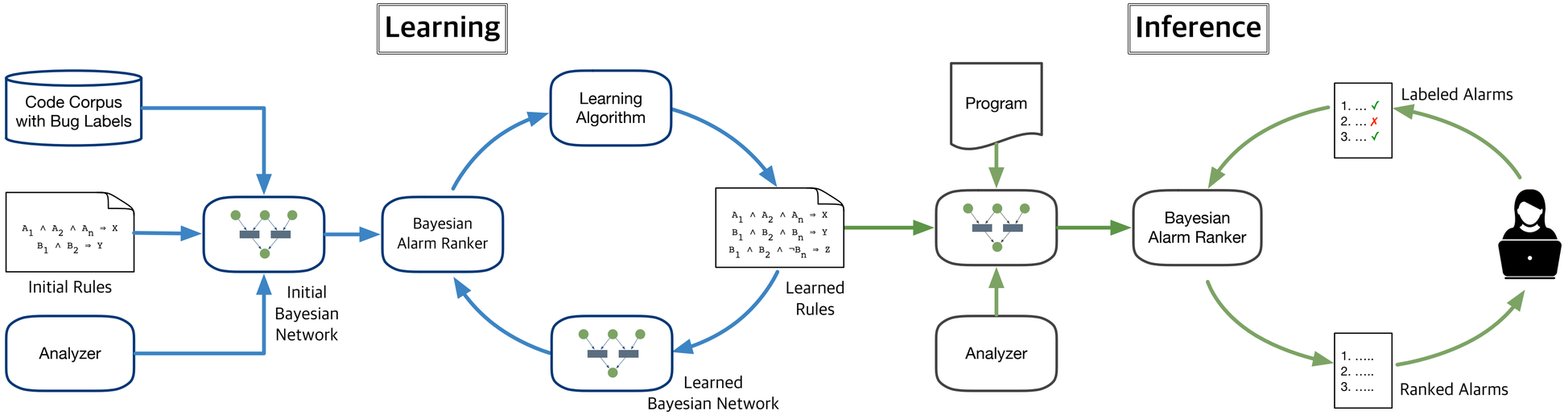
잘못된 일반화 문제


sort
BayeSmith
Bingo
Rank
User interaction
0
100
200
300
400
500
0
25
50
75
100
125
150
175


readelf
BayeSmith
Bingo
Rank
User interaction
0
25
50
75
100
125
150
175
0
10
20
30
40
50
60
70
80

핵심 기술

- 프로그램 합성 + 베이지안 추론

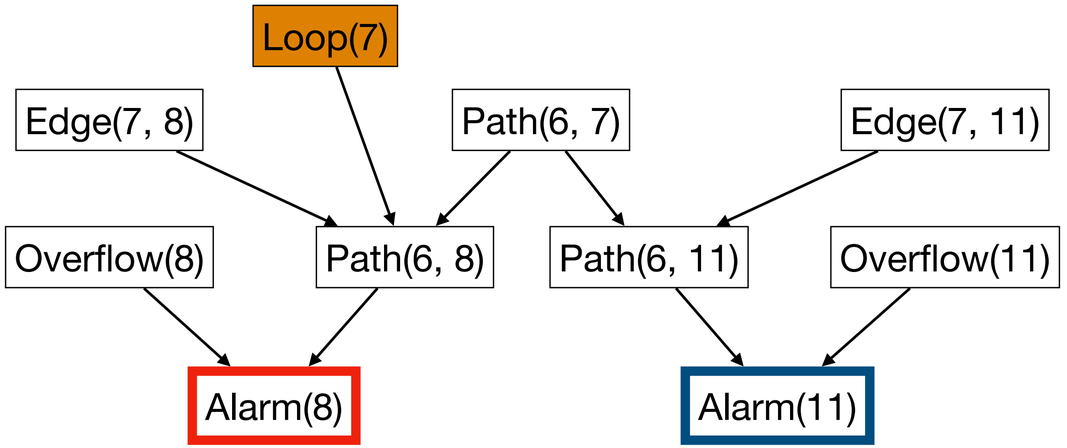


확률 모델 학습

- 잘못된 일반화가 일어나는 지점을 찾아 그 지점의 구문적 특징을 포착

Cmd ::= Lv := e | Assume(e) | Call(e, e) | Loop(e)
Lv ::= x | *E
E ::= n | Lv | E + E | …

Pr(Path(x, y) | Edge(x, y)) = 0.99
Pr(Path(x, y) | Path(x, z), Edge(x, y), **Loop(y)**) = 0.99 **x p**
Pr(Path(x, y) | Path(x, z), Edge(x, y), **!Loop(y)**) = 0.99
Pr(Alarm(y) | Path(x, y), Overflow(y)) = 0.99



정리

- PL 와 ML 의 결합 사례 1: ML 을 이용한 정적 분석
- 흥미로운 요소: 논리만으로 해결이 어려운 문제를 확률로 해결
  - 예: 요약 해석 (abstract interpretation) 이 다루지 않는 안전성 조절, 자원량 조절, 상호작용

"프로그램 분석 개론 수업을 들었을 때, 프로그램 분석을 ML, AI로 확장시키는 아이디어가 재미있다고 느꼈습니다. 더 일반적으로는 논리를 확률과 학습으로 확장시키는 과정이 흥미롭다고 생각합니다. {0, 1}의 세계에서 [0, 1]의 세계로 나아간다면 무언가 멋진 결과가 나오지 않을까 생각해서요. 또, 프로그램 분석을 AI로 확장시킬 때 베이지안 추론을 사용한다는 점이 재밌었습니다. 블랙박스인 딥러닝 보단 더 매력적으로 느껴지기 때문입니다."

- 한 수강생으로부터