기계학습을 이용하여
선별적으로 안전하게 정적 분석 하기

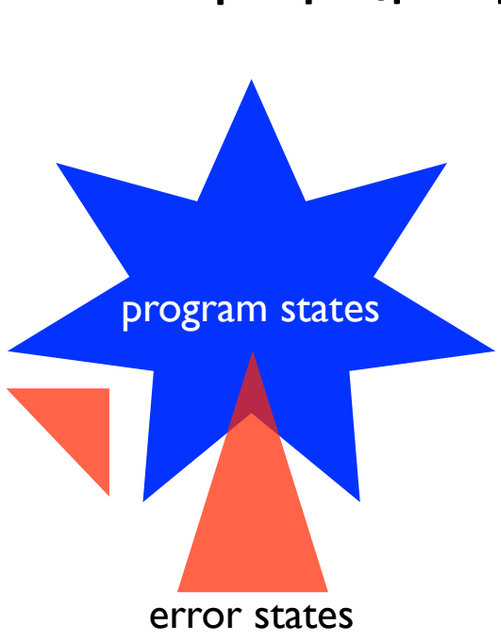
허기홍
서울대학교
(공동 연구: 오학주, 이광근)
2017.6.26 @ NAVER

꿈

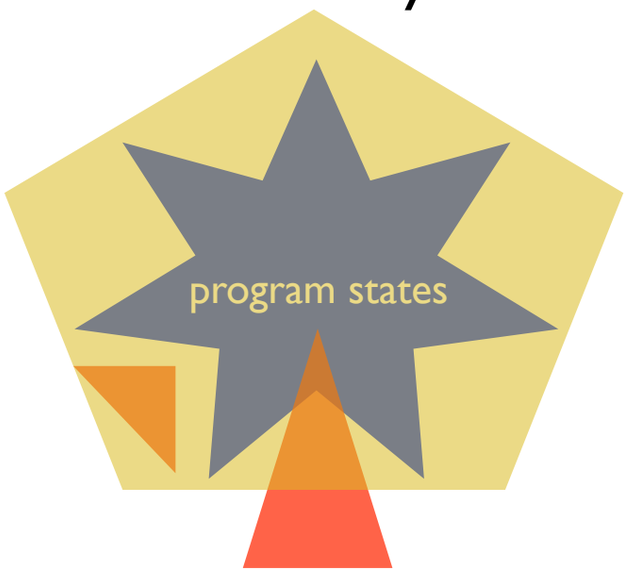
- 스스로 진화하는 정적 분석기
  - 여러 경험을 통해 점점 더 똑똑해지는 분석
  - 경험 : 비슷한 프로그램, 사용자 피드백, 과거 버전, 버그, 테스트 등
- 다른 분야는 이미 :  …

정적 분석

- **자동**으로 SW 의 동작을 **미리 어림**잡는 **일반**적인 방법
- 목적에 따라 다양하게 요약

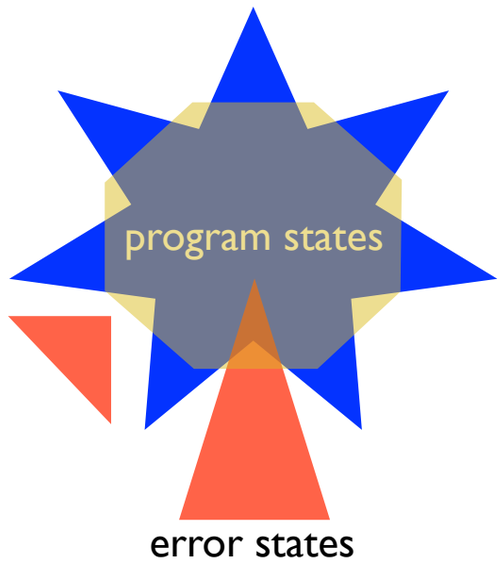


sound analysis



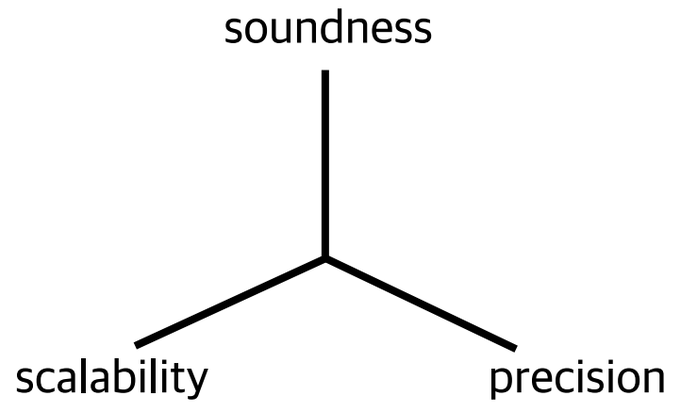
sound & precise analysis

unsound analysis

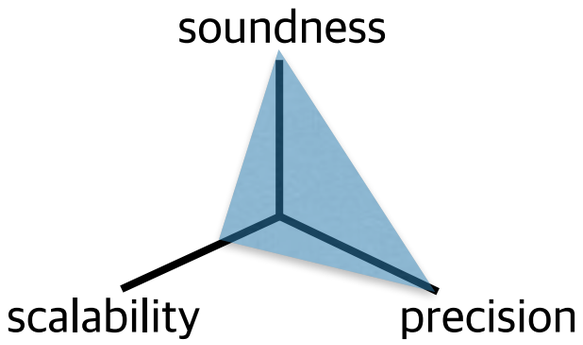


도전 과제

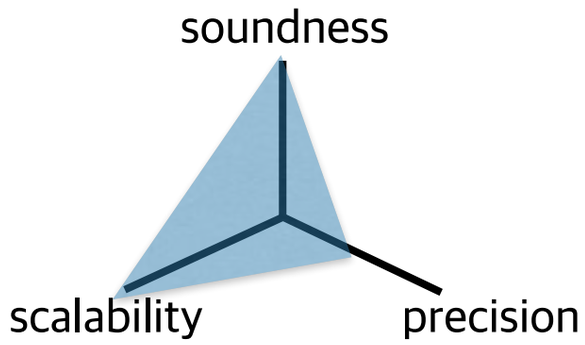
- 성능의 세가지 축: 모두 달성하는 것은 이론적으로 불가능



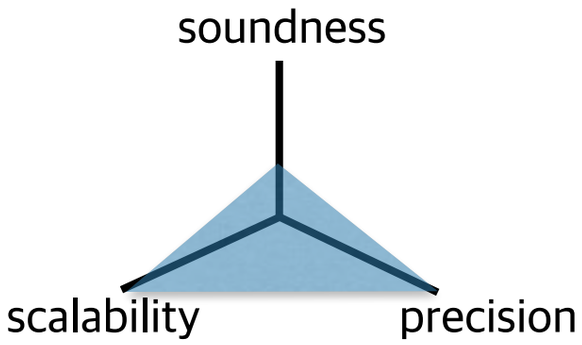
- 전통적인 분류,



무결성 검증용



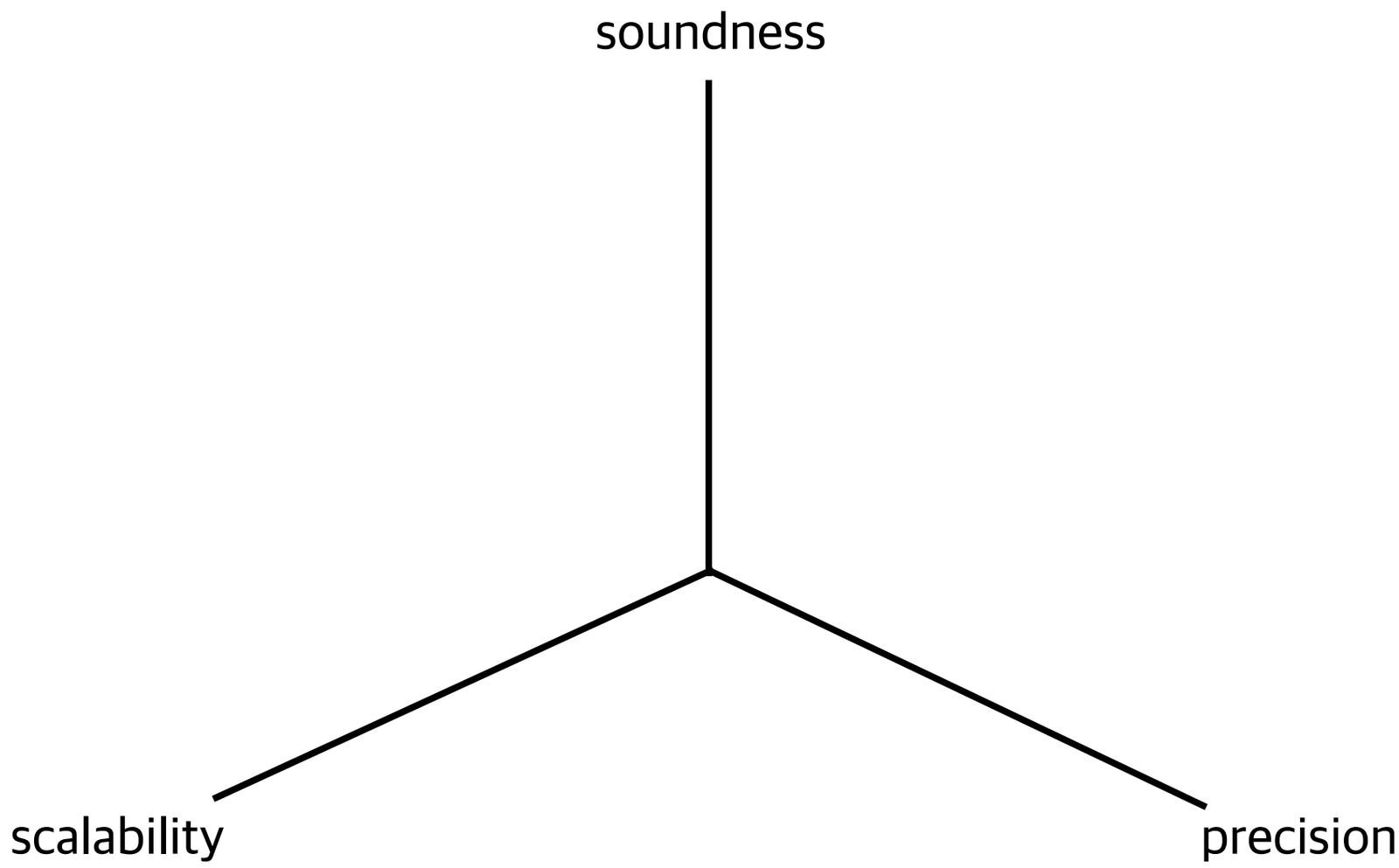
코드 최적화용



오류 검출용

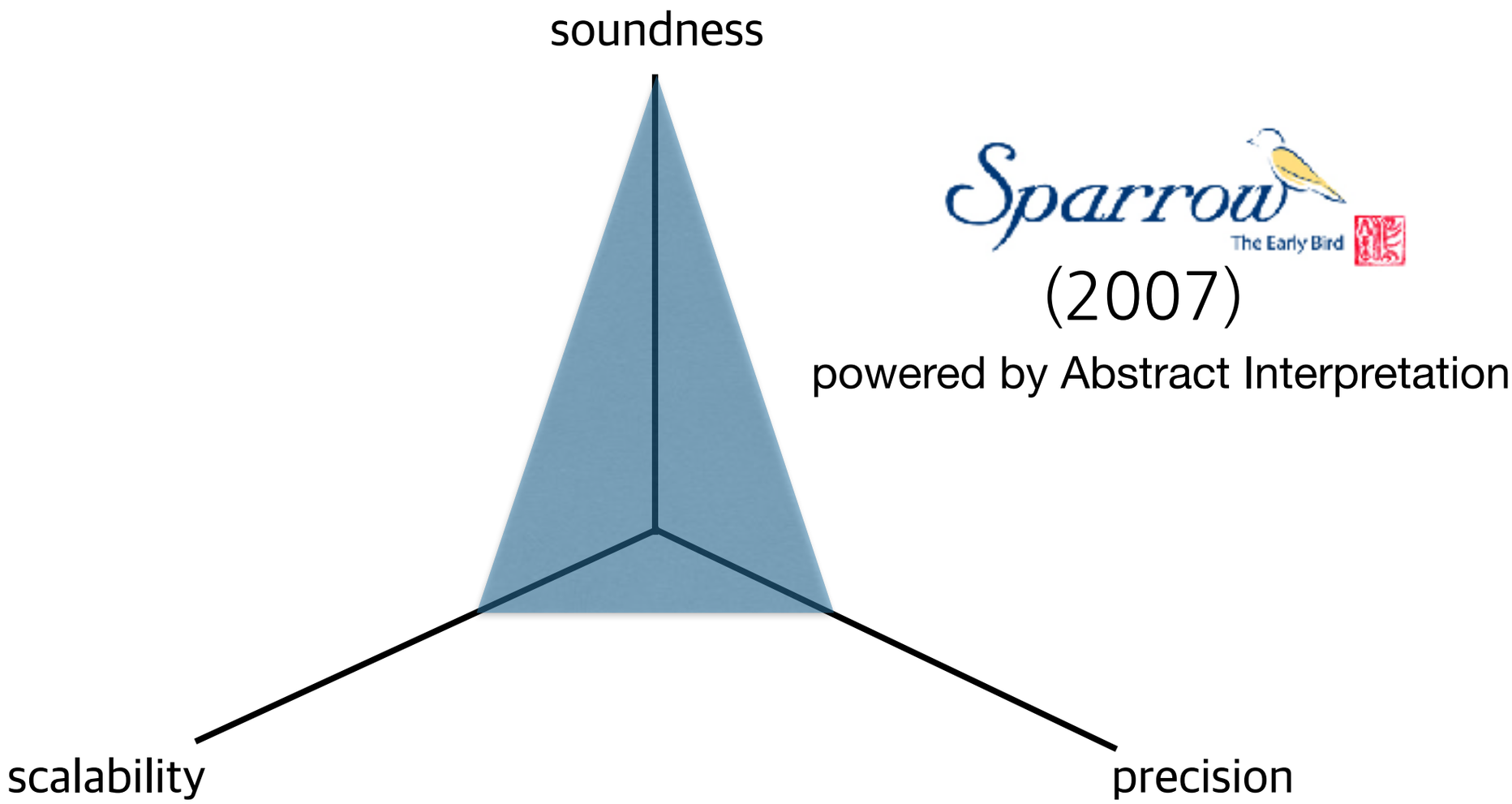
긴 여정

- 안전하고, 정확하고, 빠른 정적 분석기



긴 여정

- 안전하고, 정확하고, 빠른 정적 분석기



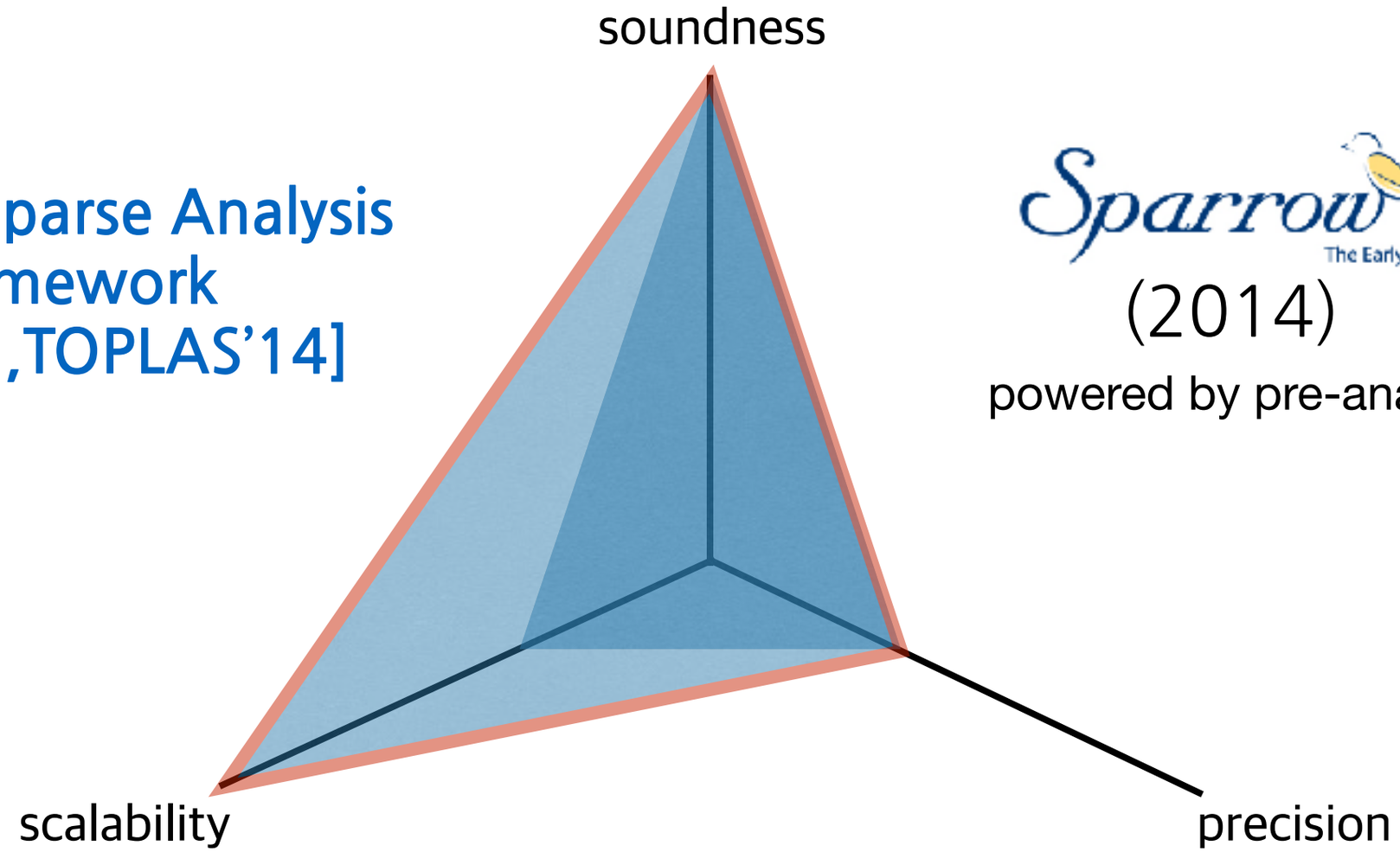
Sparrow
The Early Bird

(2007)

powered by Abstract Interpretation

긴 여정

- 안전하고, 정확하고, 빠른 정적 분석기



General Sparse Analysis Framework [PLDI'12,TOPLAS'14]

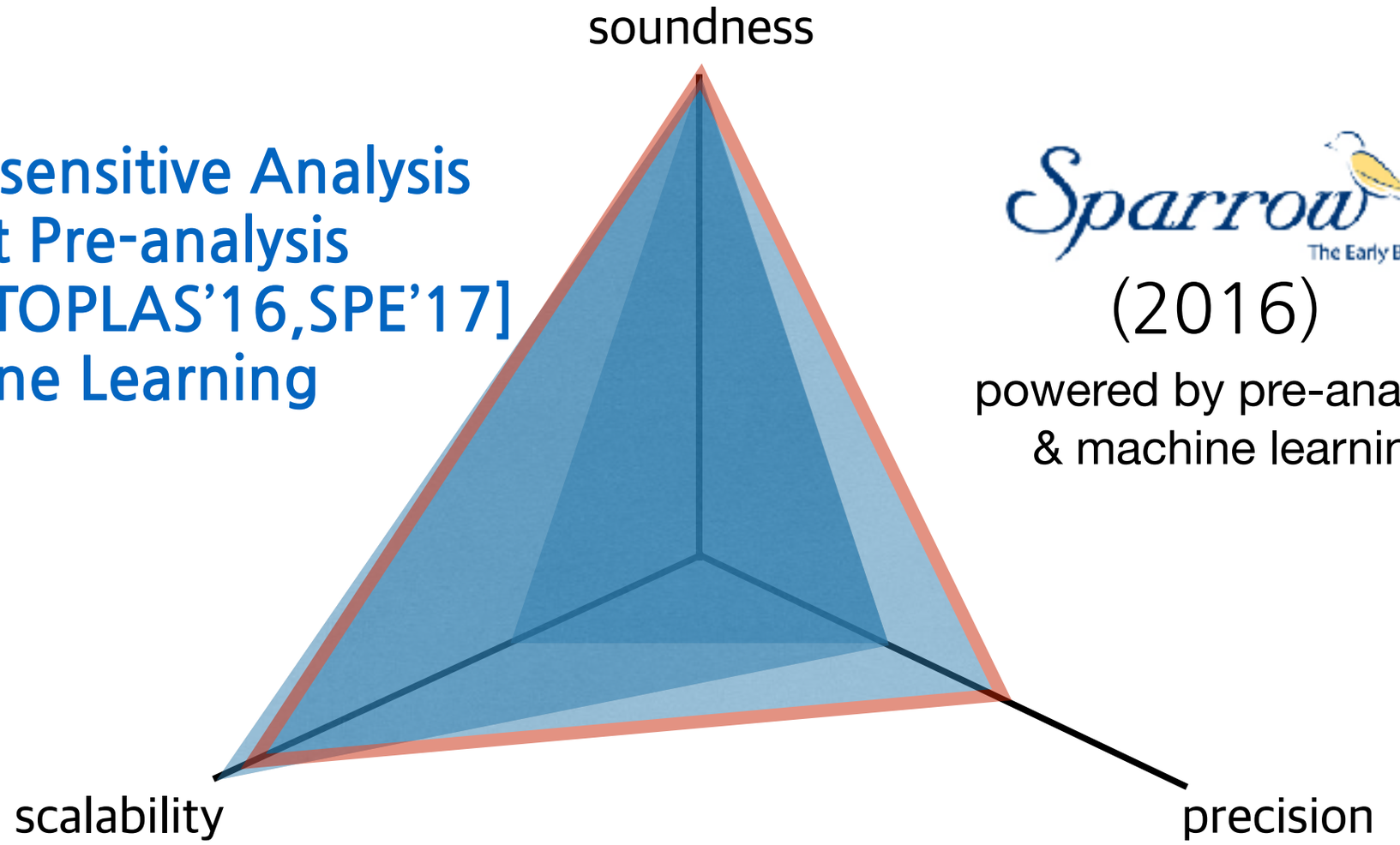
Sparrow The Early Bird

(2014)

powered by pre-analysis

긴 여정

- 안전하고, 정확하고, 빠른 정적 분석기



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

Sparrow
The Early Bird

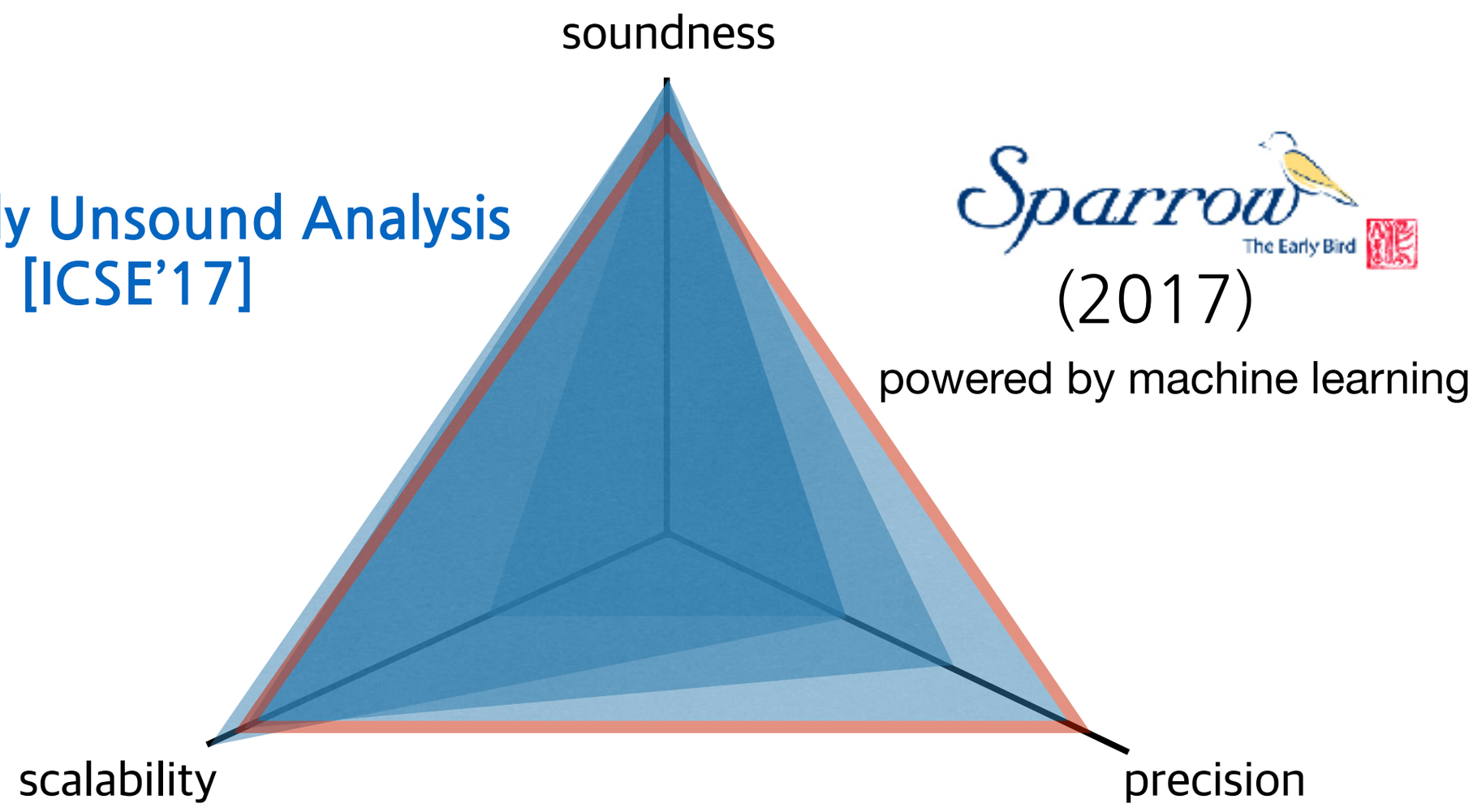
(2016)

powered by pre-analysis & machine learning

긴 여정

- 안전하고, 정확하고, 빠른 정적 분석기

Selectively Unsound Analysis
[ICSE'17]

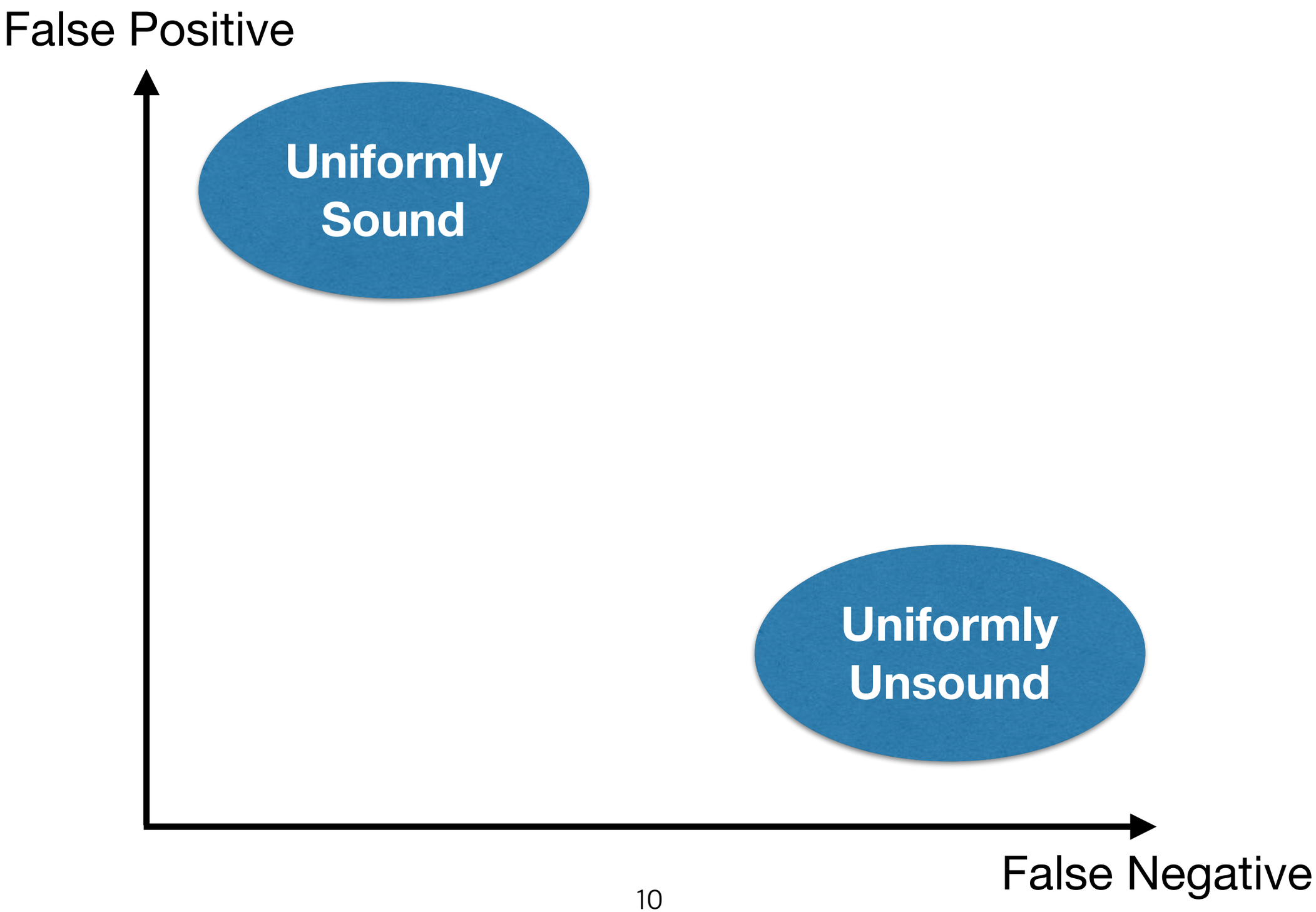


Sparrow
The Early Bird

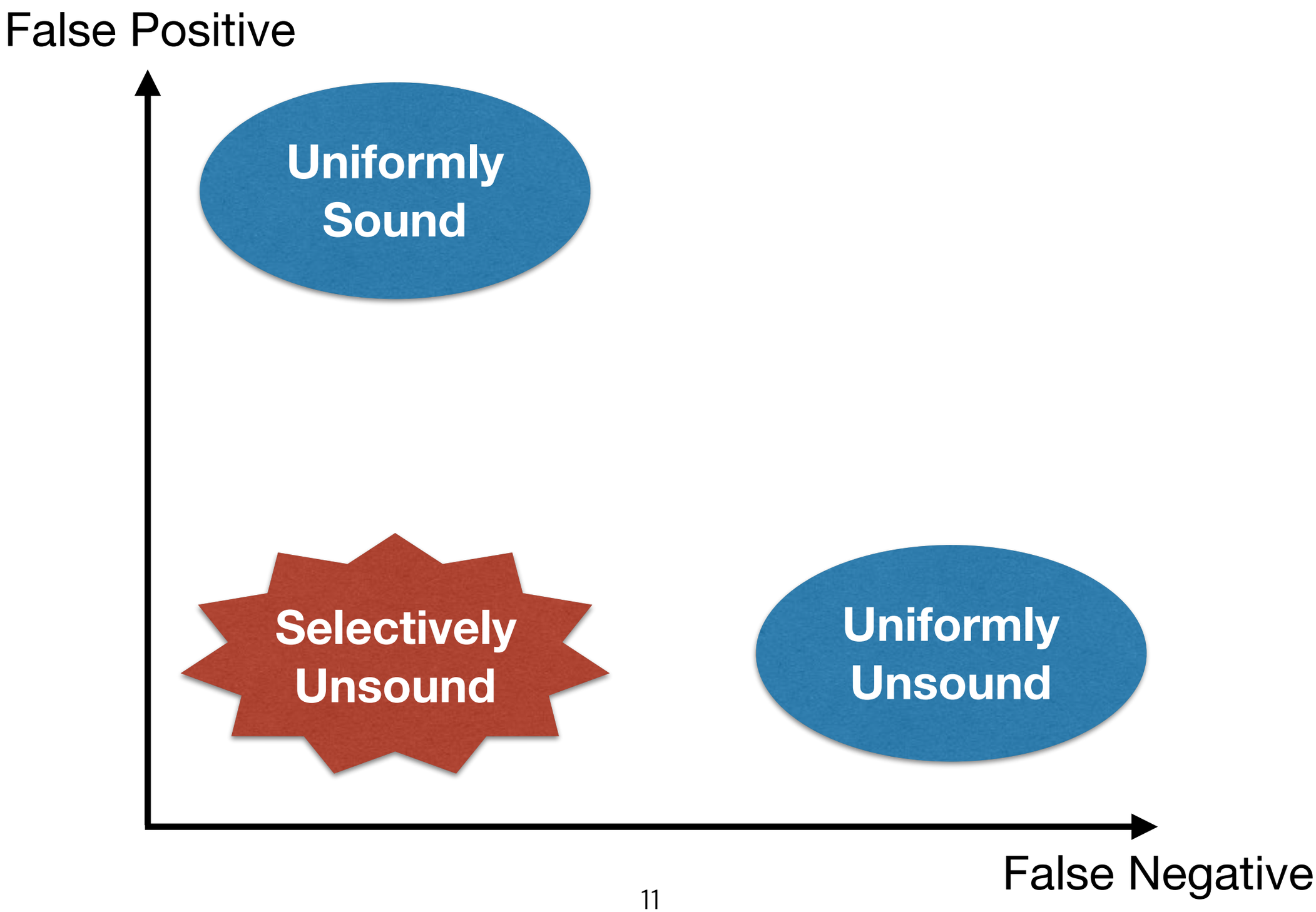
(2017)

powered by machine learning

목표


False Positive
Uniformly Sound
Uniformly Unsound
False Negative

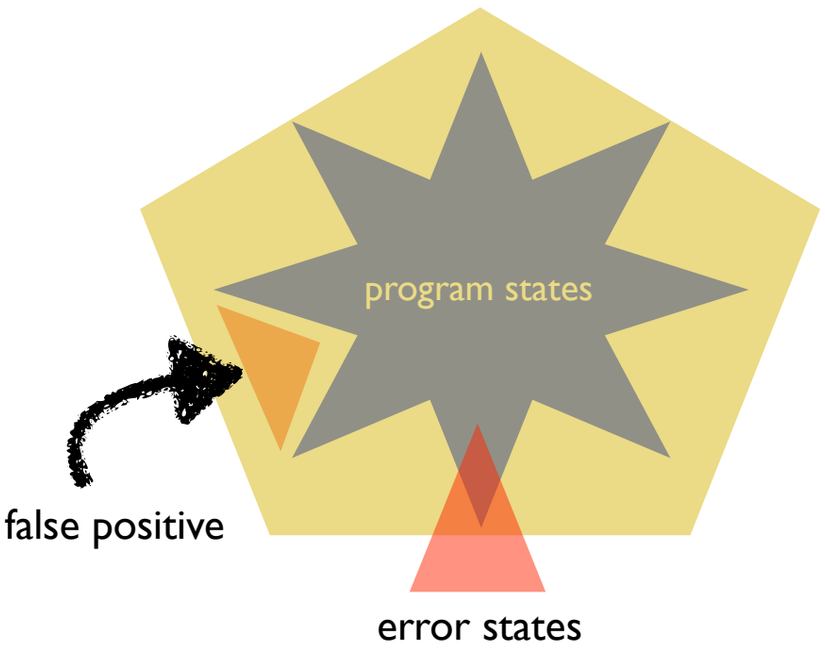
목표


False Positive
Uniformly Sound
Selectively Unsound
Uniformly Unsound
False Negative

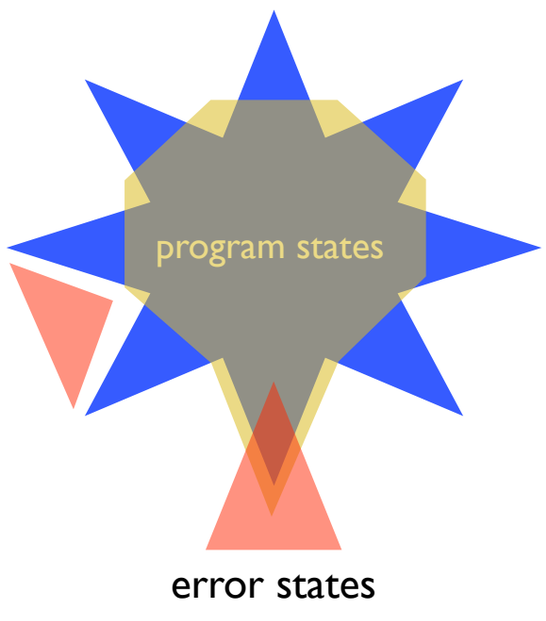
선별적으로 안전한 분석

- 불안전한 기법을 선별적으로 적용
  - 예) 순환문 해체, 라이브러리 호출 무시

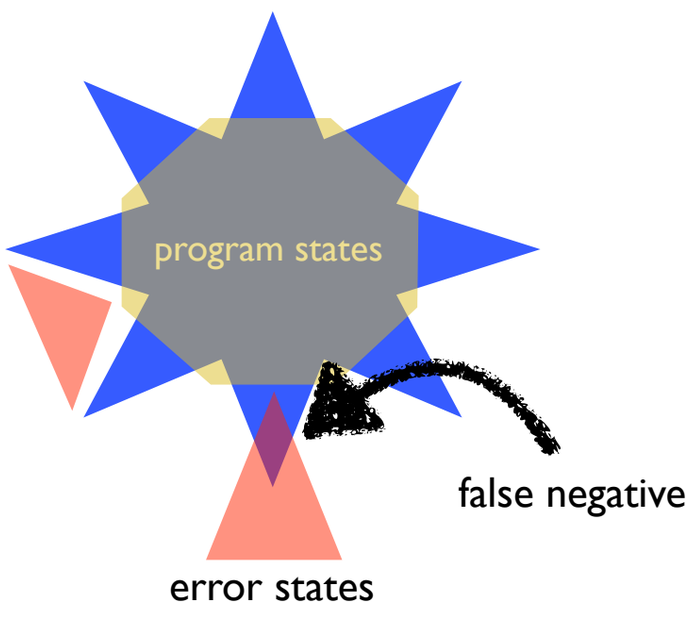
`while(e){ C }` ▶ `if(e){ C }`　　`A;lib();B;` ▶ `A;B;`



Uniformly Sound



Selectively Unsound



Uniformly Unsound

예제

- **안전한** 버퍼오버런 분석기 (숫자는 범위로 요약)
  - 모든 순환문을 안전하게

```
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

- **안전한** 버퍼오버런 분석기 (숫자는 범위로 요약)
  - 모든 순환문을 안전하게

```
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
```


str.size: [12, 12]

i: [0, +oo]

size: [0, +oo]

i: [0, +oo]

예제

- **획일적으로 불안전한** 버퍼오버런 분석기
  - 모든 순환문을 해체

```
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

- **획일적으로 불안전한** 버퍼오버런 분석기
  - 모든 순환문을 해체

```
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
```

i: [0, 0]

i: [0, 0]

예제

- **선별적으로 안전한** 버퍼오버런 분석기
  - "무해한" 순환문만 선별적으로 해체

```
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
```

예제

- **선별적으로 안전한** 버퍼오버런 분석기
  - "무해한" 순환문만 선별적으로 해체

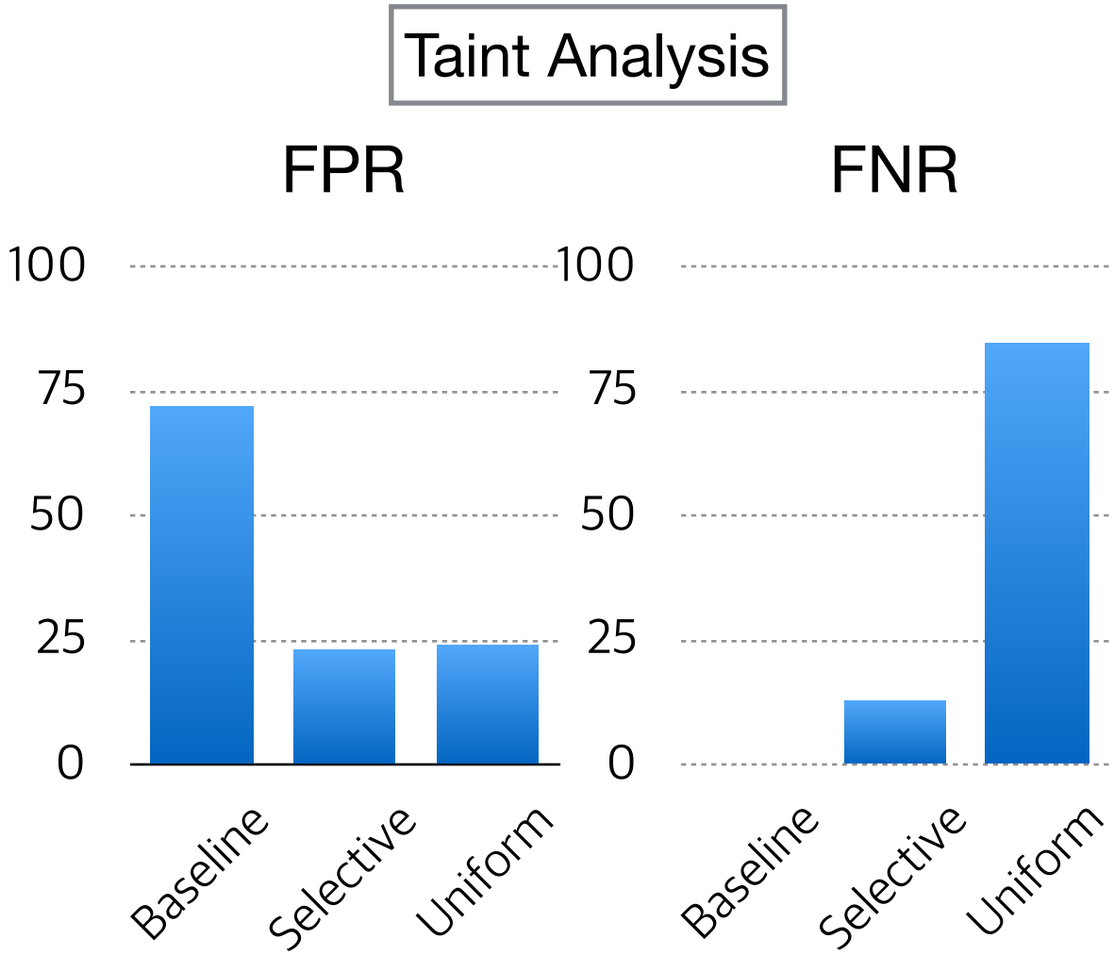
```
str = "hello world";
i = 0;
if( str[i])           // buffer access 1
  skip;

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

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

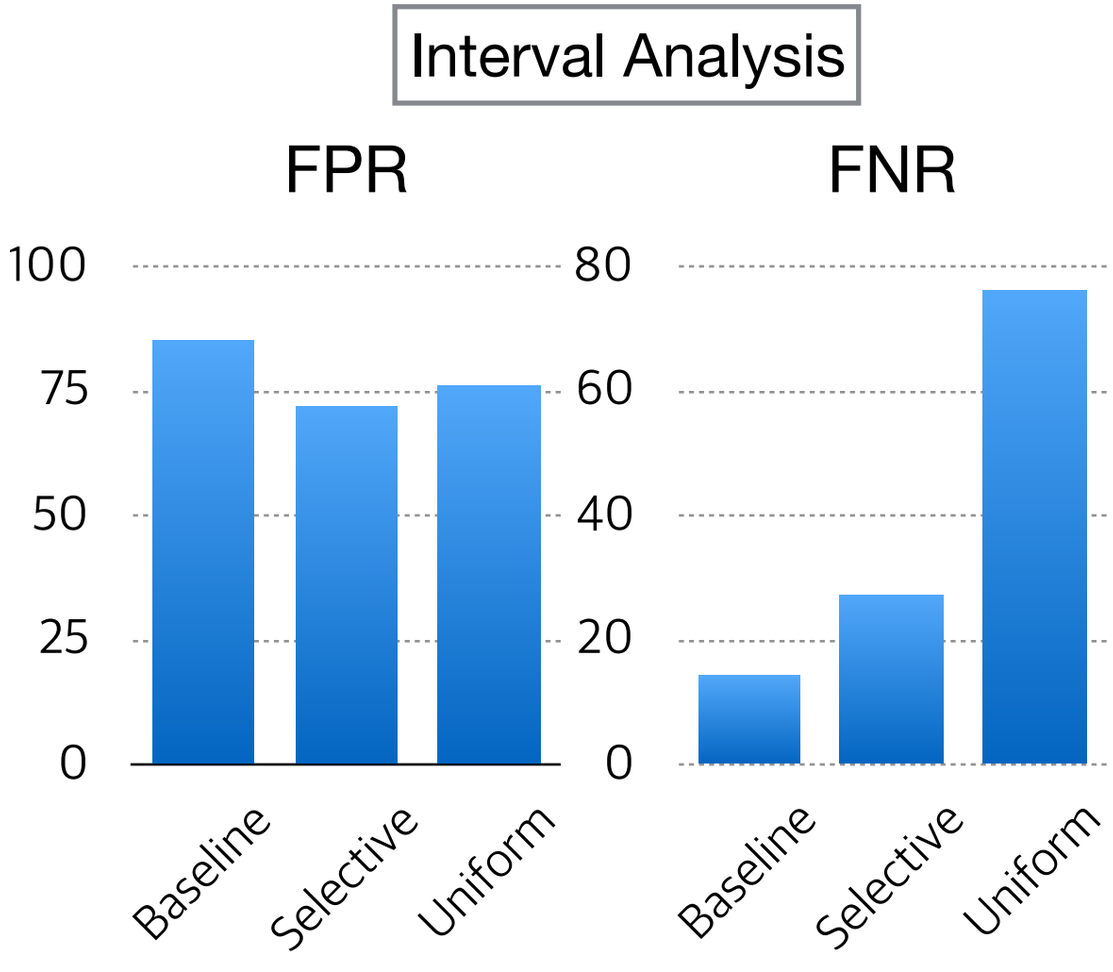
i: [0, 0]

i: [0, +oo]

성능

- 실험: 두 가지 정적분석기 + 오픈소스 SW
  - 오염 (taint) 분석: 106 포맷 스트링 오류 / 13 programs
  - 구간 (interval) 분석: 138 버퍼 오버런 오류 / 23 programs



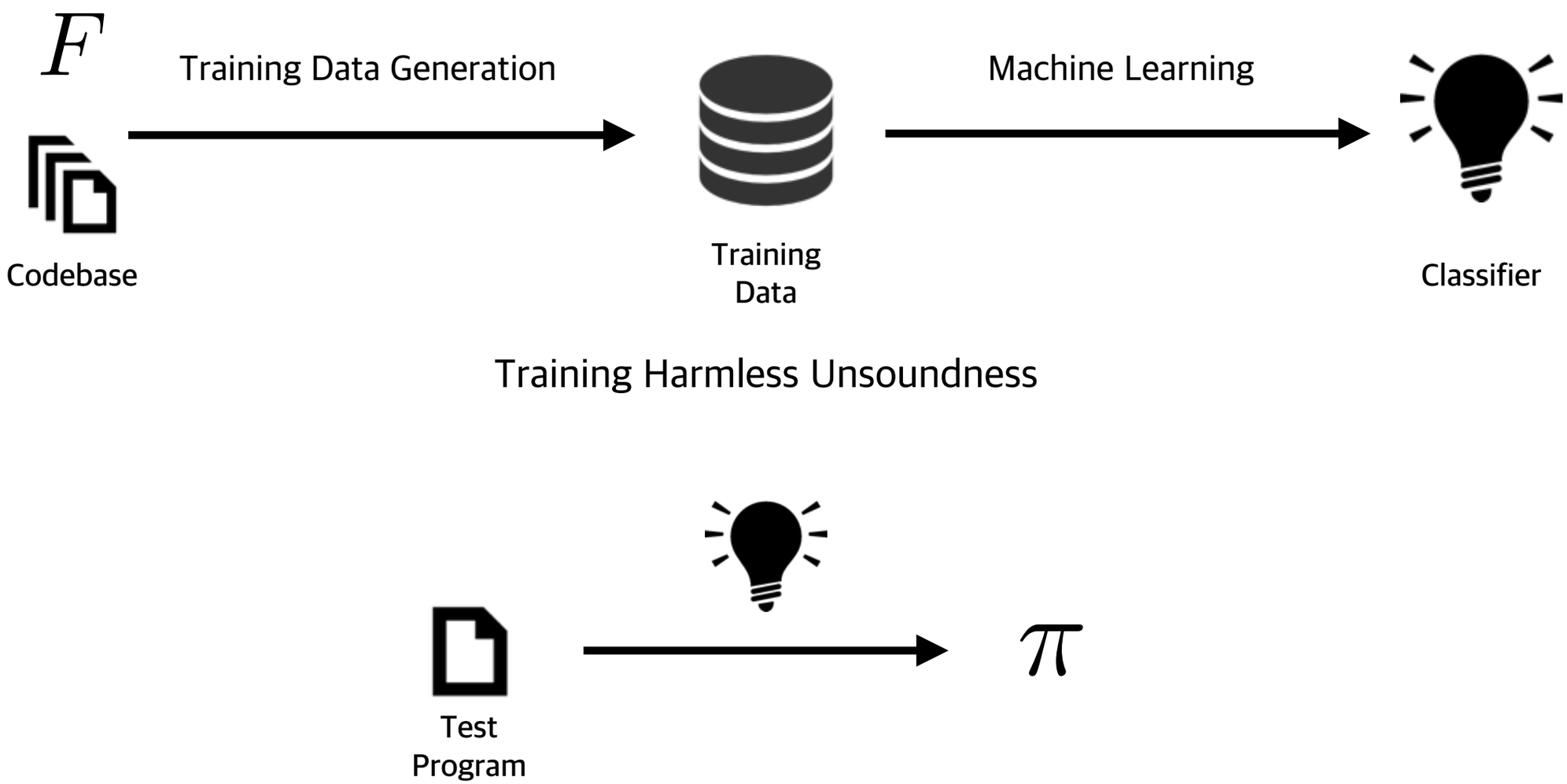


문제 정의

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

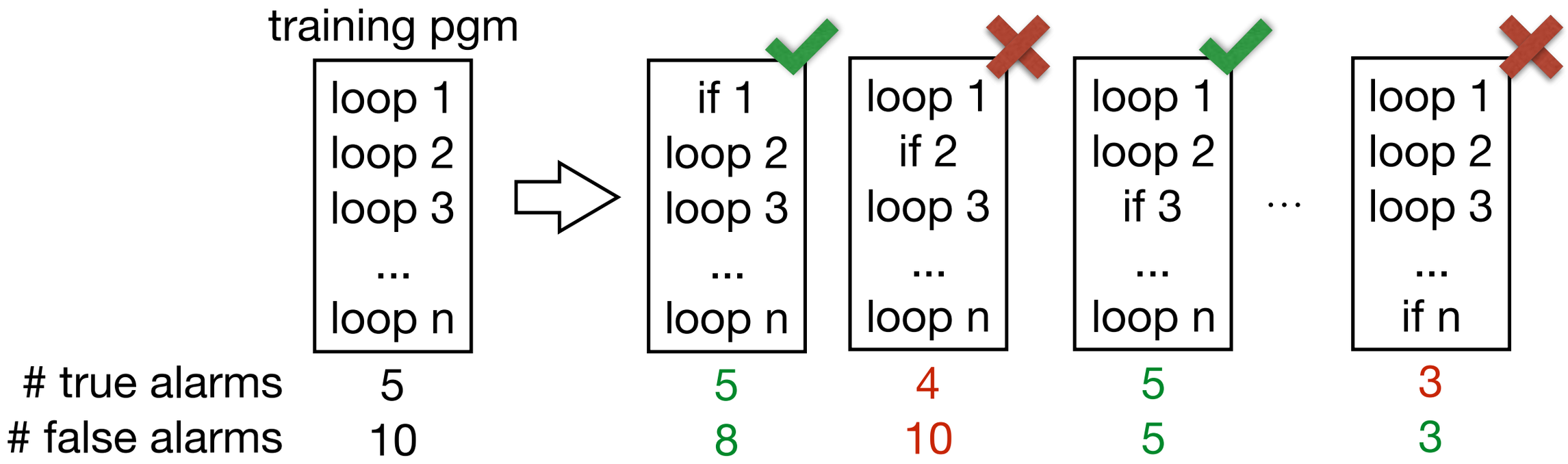
- 안전성을 포기할 대상 $\pi \in \Pi$ 찾기
  - 불안전하게 분석할 순환문 ($\Pi = 2^{Loop}$)
  - 불안전하게 분석할 라이브러리 호출 ($\Pi = 2^{Lib}$)
- $p \in \pi$ 만 불안전하게 분석

전체 구조


F
Training Data Generation
Machine Learning
Codebase
Training
Data
Classifier
Training Harmless Unsoundness
Test
Program
π
Inferring Harmless Unsoundness

학습 데이터 생성

- 준비물: 버그가 알려져 있는 프로그램 뭉치 + 정적 분석기
- 학습 데이터: 분석의 정확도를 떨어뜨리지만 무해한 부품
  - 예) "어떤 순환문 해체했더니 FP 감소, TP 유지"



특질 & 학습

- 각 프로그램 부품을 특질 벡터 (feature vector) 형태로 인코딩

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

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

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

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

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

- 기존 ML 알고리즘으로 분류기 (classifier) 학습
  - e.g.) OC-SVM

특질 (feature)

- 22 가지 순환문 관련 특질

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

특질 (feature)

- 15 가지 라이브러리 호출 관련 특질

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

무해한 부품

- 분석 정확도를 떨어트리지만 안전한 코드
  - 전형적: 코딩 스타일, 요약 도메인, 등등
  - 풍부: 거짓 경보 >> 버그, 올바른 코드 >> 틀린 코드
  - 예)

```
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
```

무해한 부품

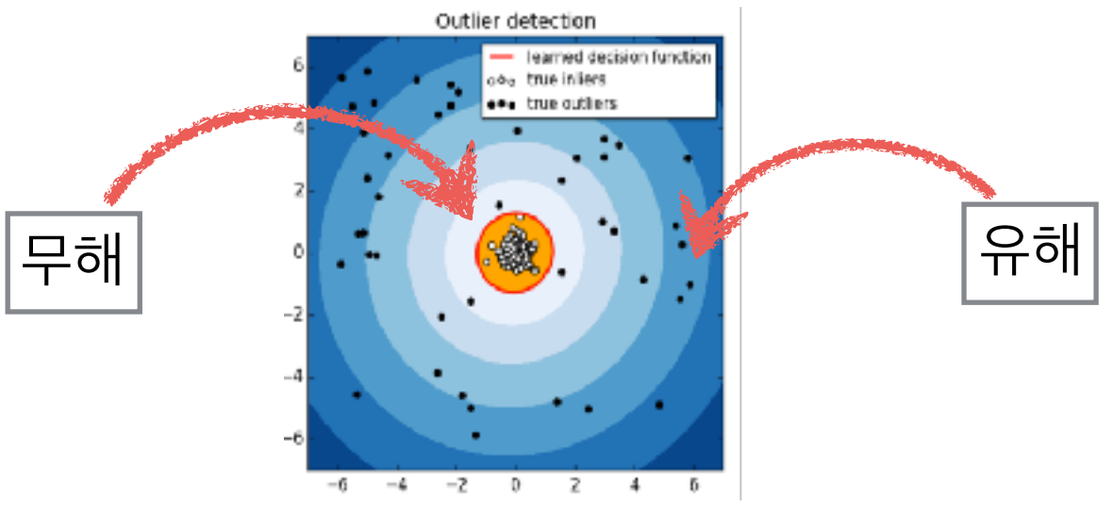
- 분석 정확도를 떨어트리지만 안전한 코드
  - 전형적: 코딩 스타일, 요약 도메인, 등등
- 풍부: 거짓 경보 >> 버그, 올바른 코드 >> 트릭 코드
- 예)

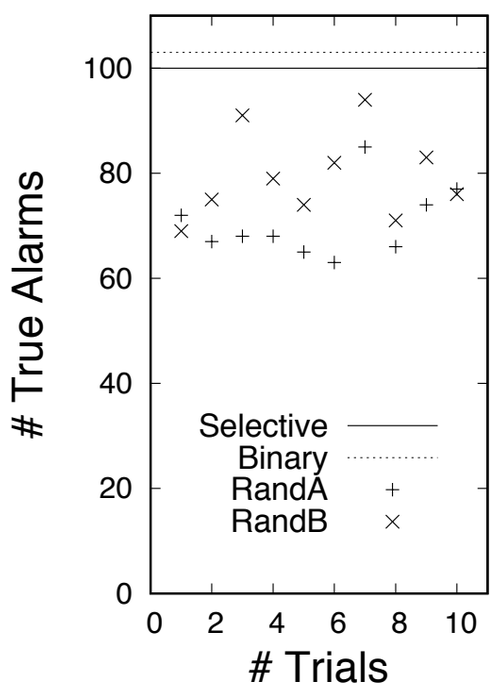
```
str = "hello world";
for(i=0;  str[i]; i++)// buf
  skip;

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

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

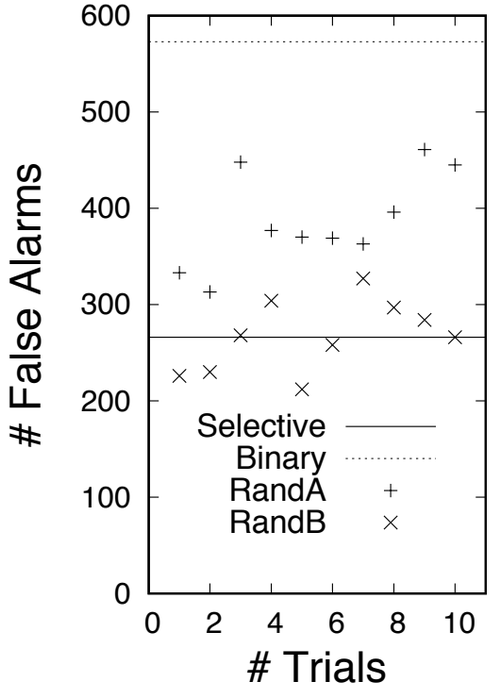
무해한 순환문
- termination with null check
- iteration on a finite string
- no global variables
- …

학습

- 전형성 검출에 특화된 학습 알고리즘 (OC-SVM) 사용
  - 전형적이면 무해, 비전형적이면 유해
  - 일반 SVM 에 비해 전형성 검출 성능 우수







중요한 특질

- 구간(interval) 분석
  - 유한 문자열을 훑는 순환문
  - 숫자를 리턴하거나 문자열을 받는 라이브러리 호출

```
str = “hello world”;
for (p = str; *p; p++)
 ...
```

```
int r = lib1();
lib2(str1, str2);
```

중요한 특질

- 구간(interval) 분석
  - 유한 문자열을 훑는 순환문
  - 숫자를 리턴하거나 문자열을 받는 라이브러리 호출

```
str = “hello world”;
for (p = str; *p; p++)
 ...
```

finite string

array access

ptr increment

return integer

```
int r = lib1();
lib2(str1, str2);
```

str manipulation

중요한 특질

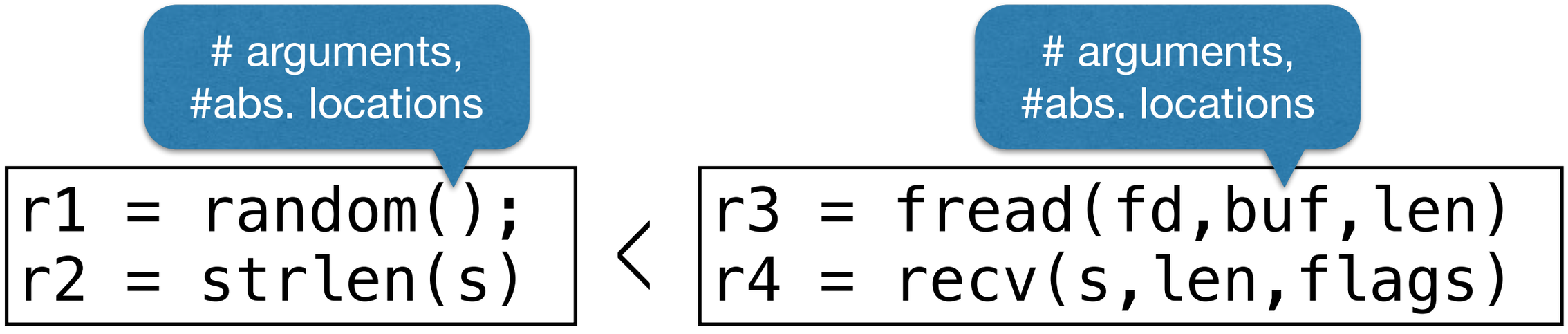
- 오염(taint) 분석
  - 사용자 입력을 전달하지 않는 라이브러리 호출

```
r1 = random();
r2 = strlen(s)
```

```
r3 = fread(fd,buf,len)
r4 = recv(s,len,flags)
```

중요한 특질

- 오염(taint) 분석
  - 사용자 입력을 전달하지 않는 라이브러리 호출



후속 연구

- 기계 학습을 위한 특질 자동 생성
  - 특질: 해당 분석의 핵심을 드러내는 (작은) 프로그램

```
/* real code */
pos = 0;
path = malloc(tmp);
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++;
  }
}
```

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

후속 연구

- 기계 학습을 위한 특질 자동 생성
  - 특질: 해당 분석의 핵심을 드러내는 (작은) 프로그램

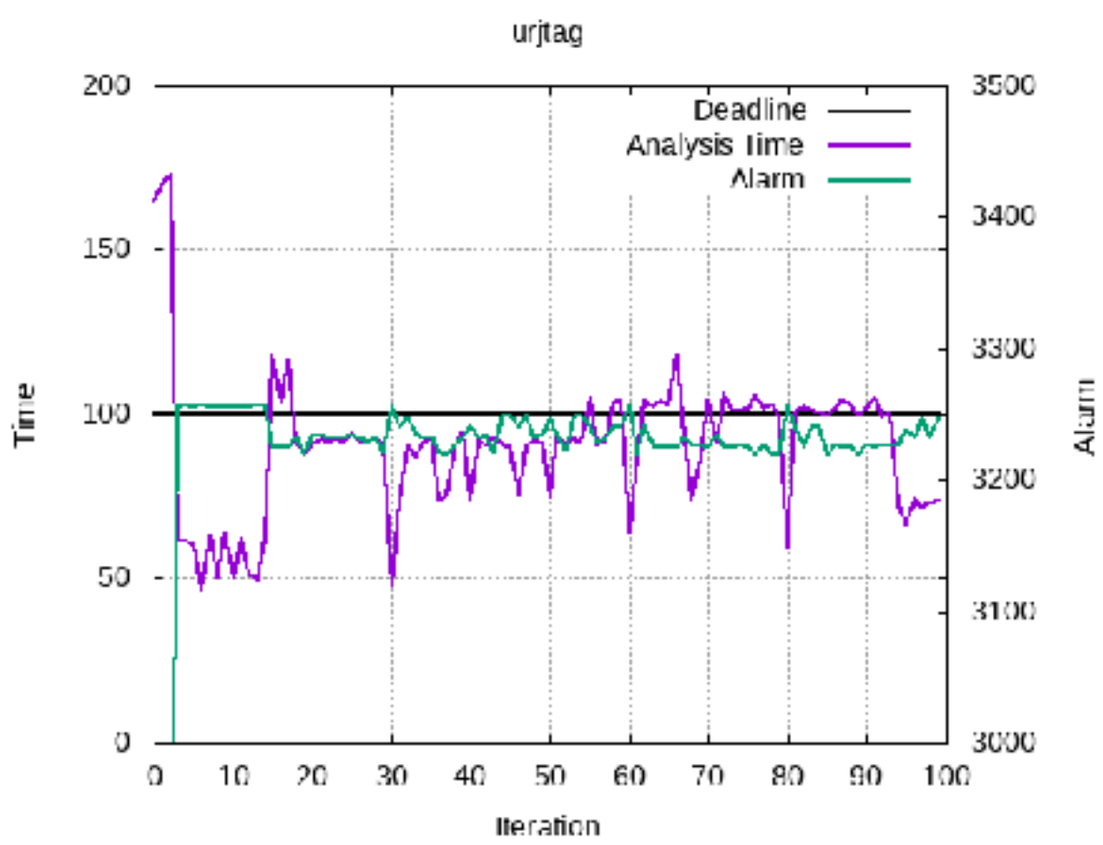
```
/* real code */
pos = 0;
path = malloc(tmp);
path_len = tmp;
file = str_make(0);
if (*){
  while(pos < path_len){
    path[pos] = 1;
    str_add_char(path,pos);
    str_buf = nstr_to_str(path);
    pos++;
  }
}
```

≾

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

후속 연구

- 분석기에 타이머 (제한시간) 달기
  - 실시간 스케줄링: 남은 시간에 따라 정확도를 조절



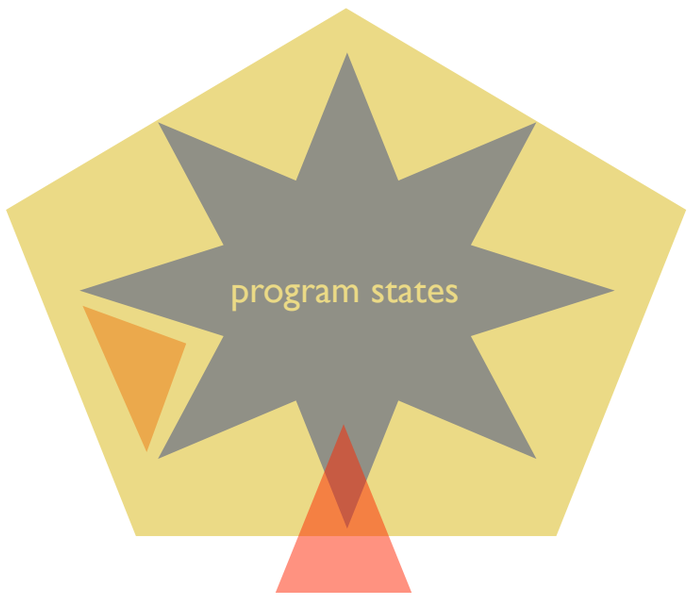
정리

- **선별적**으로 안전성을 조절하는 정적 분석기
  - 획일적으로 안전 / 불안전한 분석기보다 효과적
  - 기계학습을 이용하여 자동, 체계적으로 안전성을 조절

Sound **Selectively Unsound** Uniformly Unsound

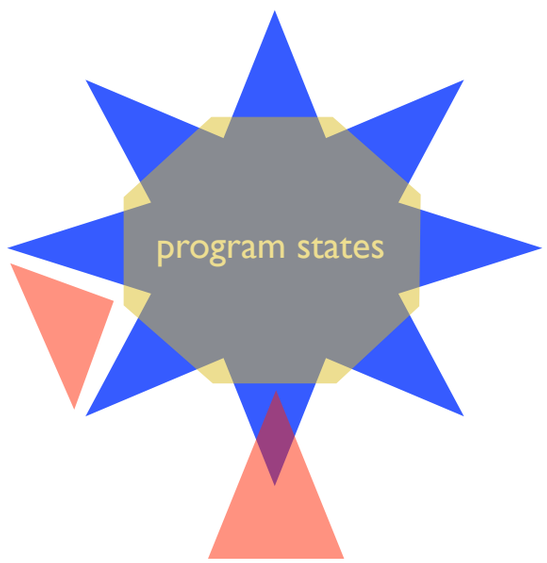
정리

- **선별적**으로 안전성을 조절하는 정적 분석기
  - 획일적으로 안전 / 불안전한 분석기보다 효과적
  - 기계학습을 이용하여 자동, 체계적으로 안전성을 조절



Sound

Selectively Unsound



Uniformly Unsound

고맙습니다