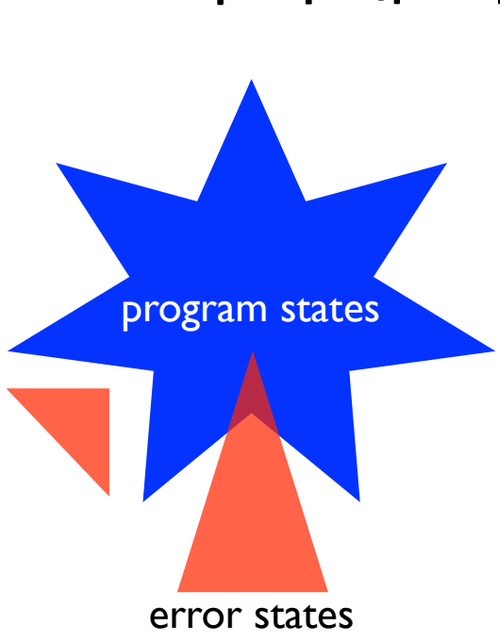
예비분석과 기계학습을 이용하여 선별적으로 정확하게 정적 분석

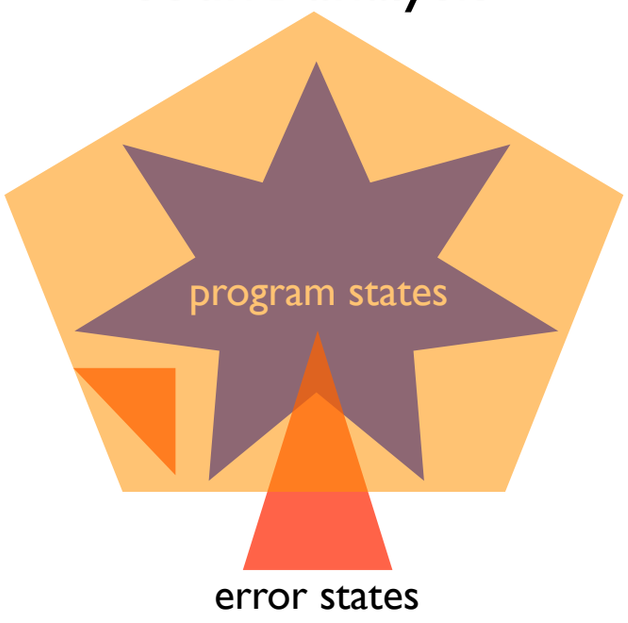
Selectively Sensitive Static Analysis by Impact Pre-analysis and Machine Learning

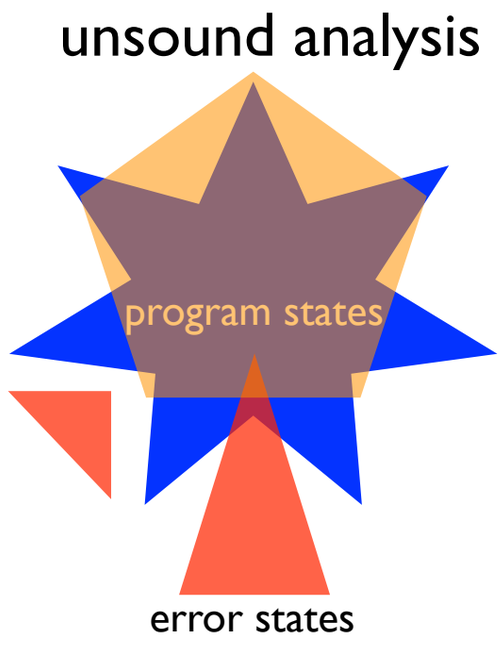
허 기 홍
서울대학교 박사학위심사
지도교수: 이광근
2017. 5. 2

정적 분석

- 자동으로 SW 의 동작을 미리 어림잡는 일반적인 방법
- 목적에 따라 다양하게 요약

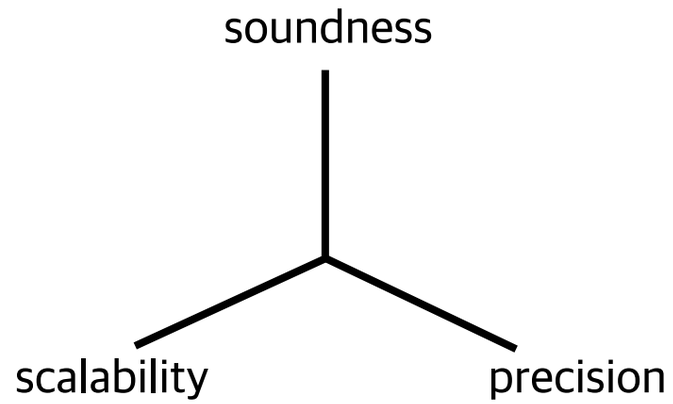




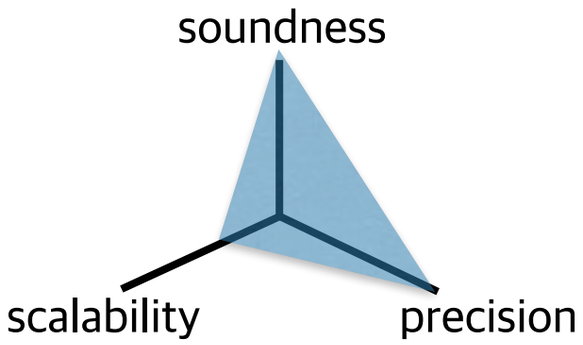


도전 과제

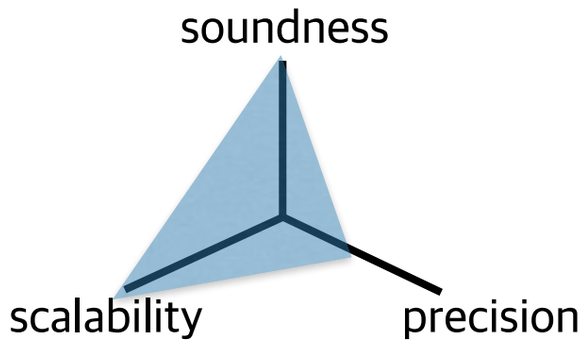
- 성능의 세가지 축: 모두 달성하는 것은 이론적으로 불가능



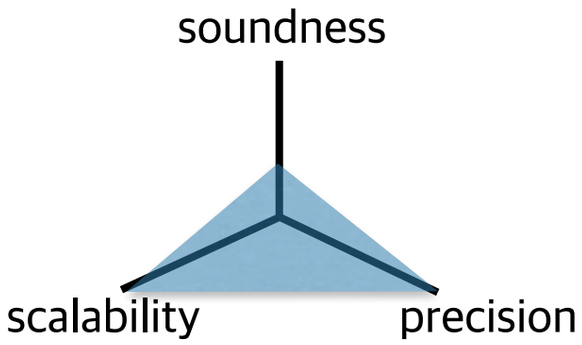
- 전통적인 분류,



무결성 검증용



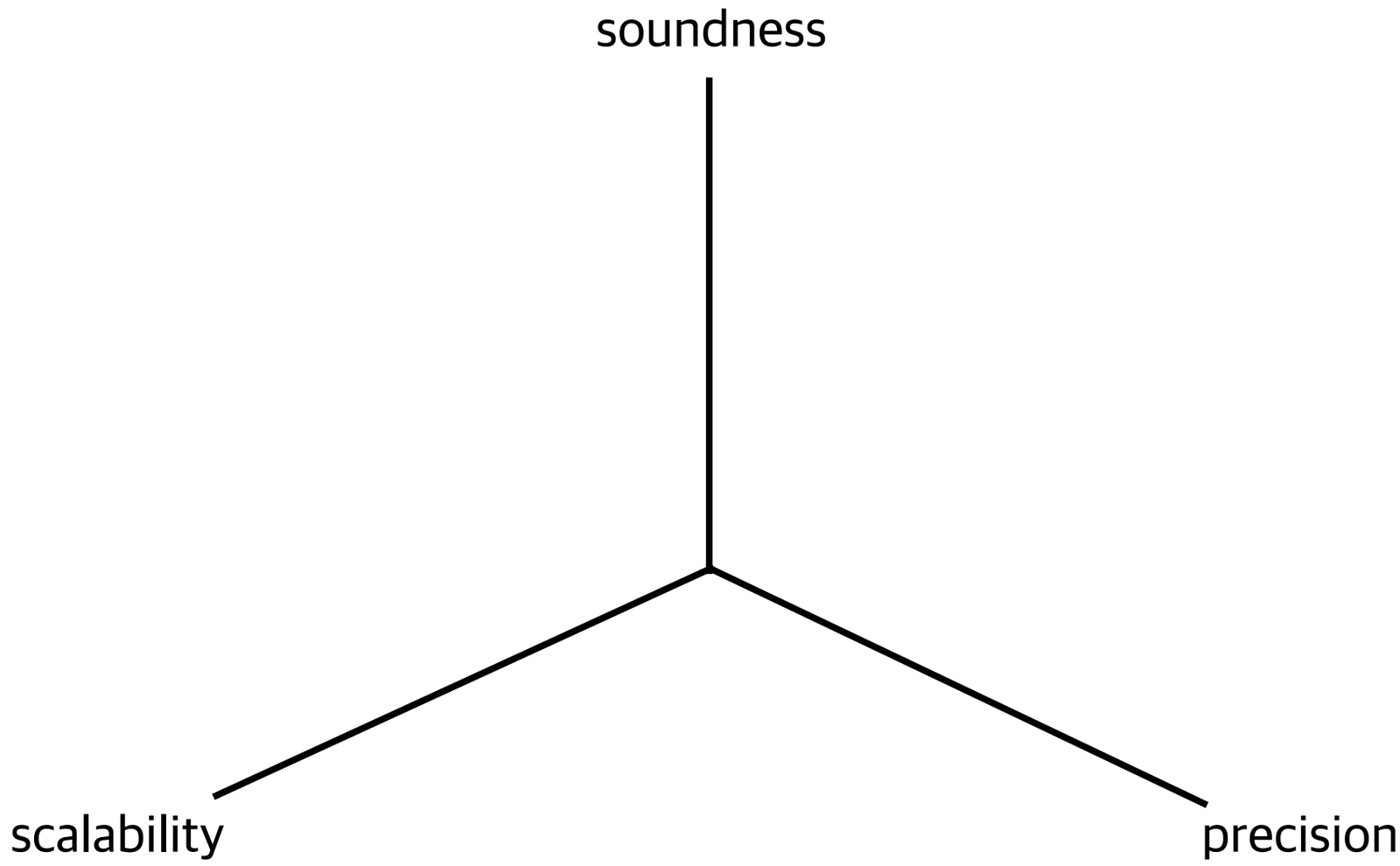
코드 최적화용



오류 검출용

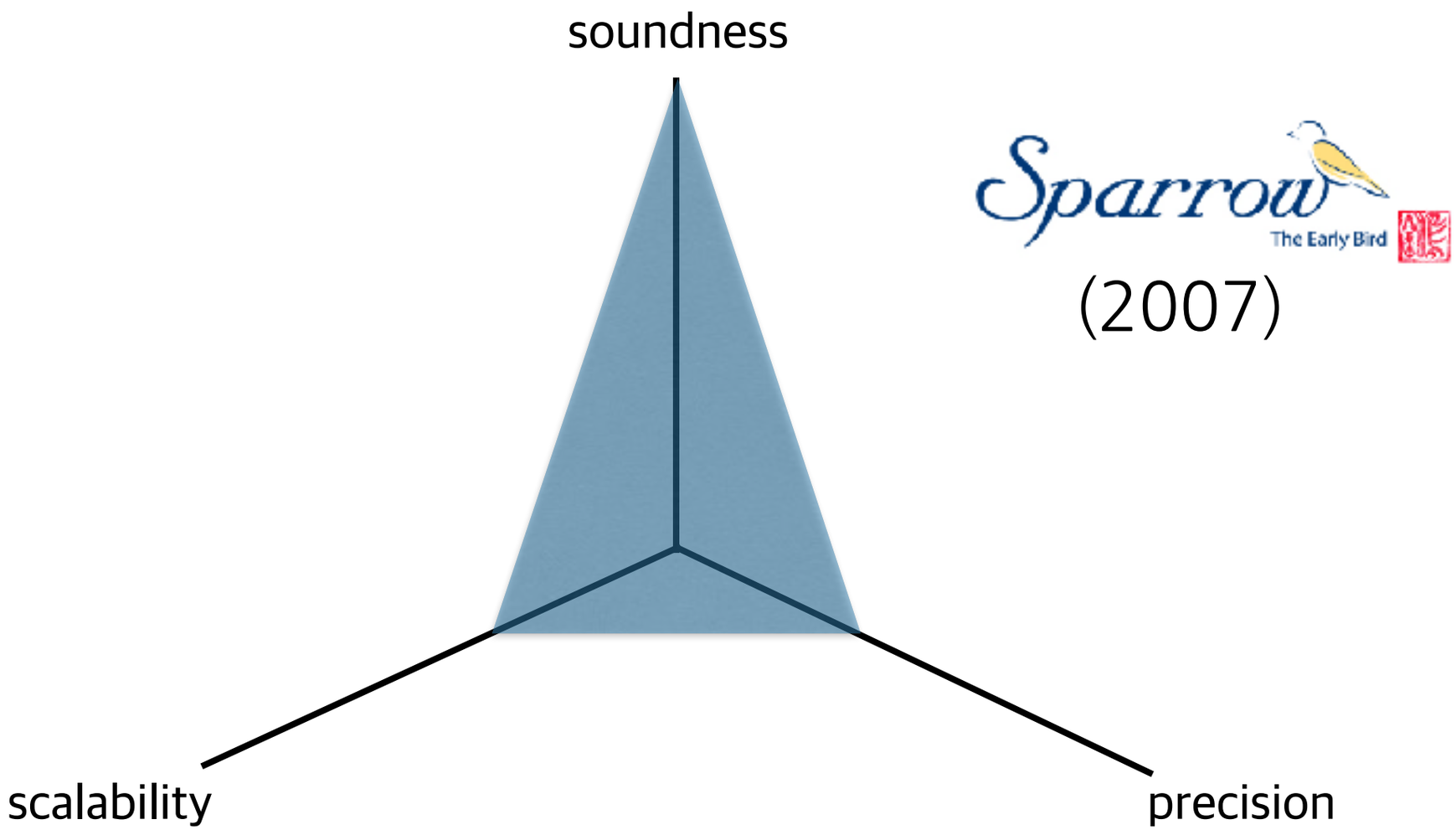
목표

- Sound, precise yet scalable static analysis



목표

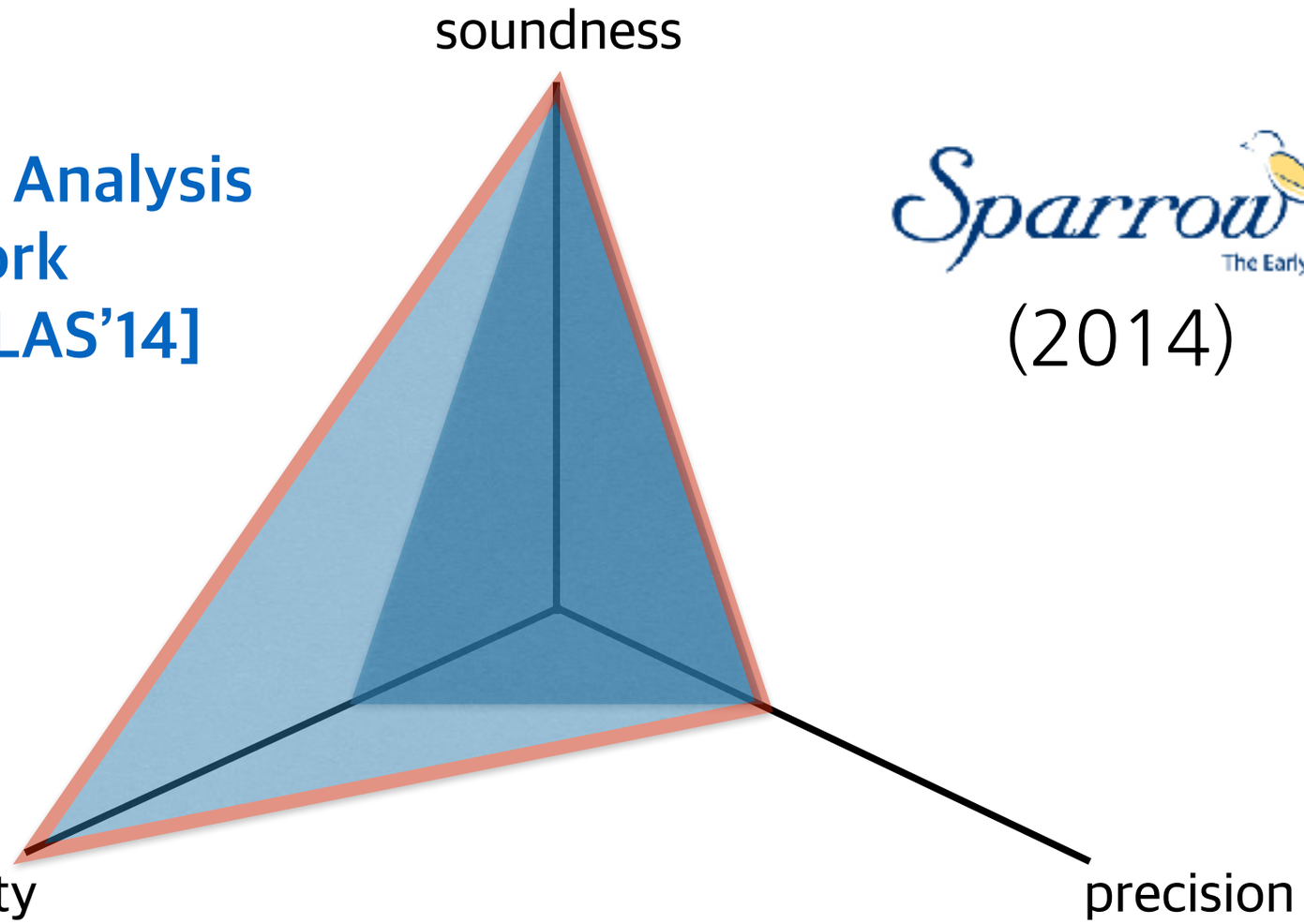
- Sound, precise yet scalable static analysis



목표

- Sound, precise yet scalable static analysis

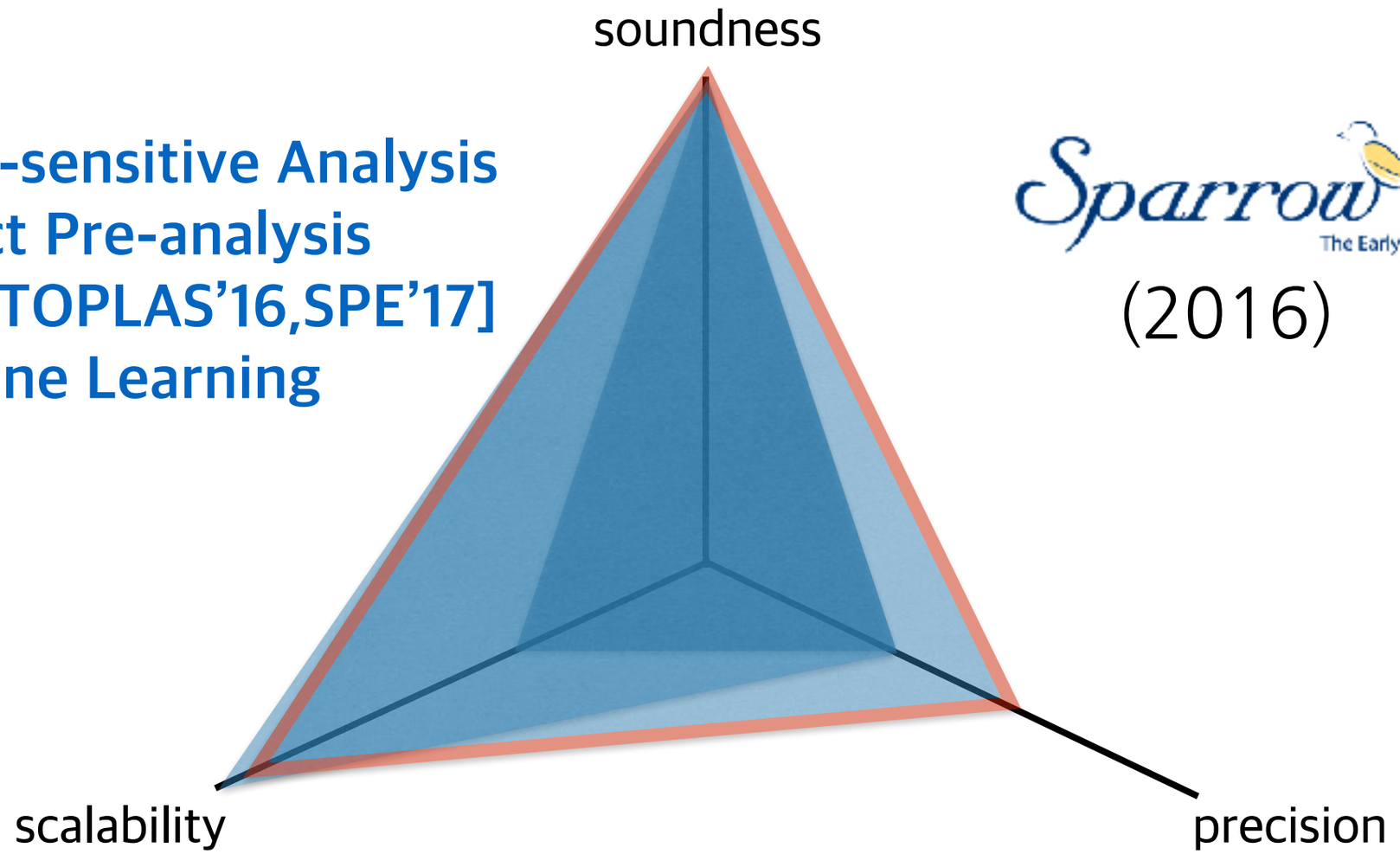
General Sparse Analysis Framework [PLDI'12,TOPLAS'14]



Sparrow The Early Bird (2014)

목표

- Sound, precise yet scalable static analysis



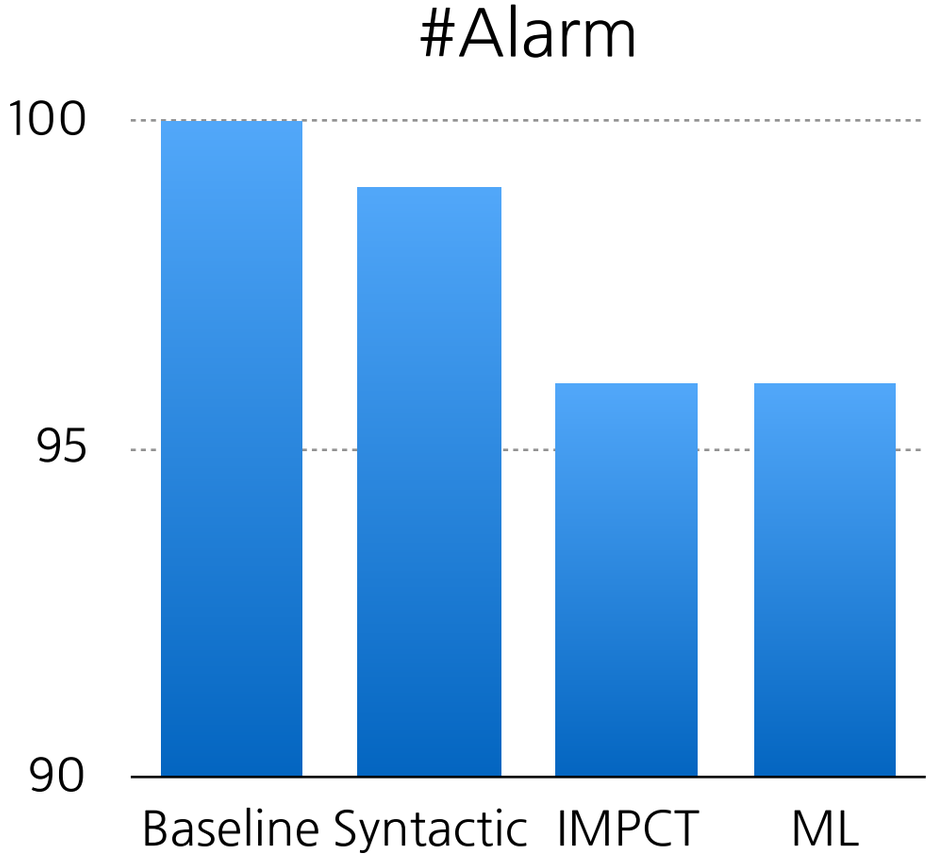
Selective X-sensitive Analysis
- **by Impact Pre-analysis [PLDI'14,TOPLAS'16,SPE'17]**
- **by Machine Learning [SAS'16]**

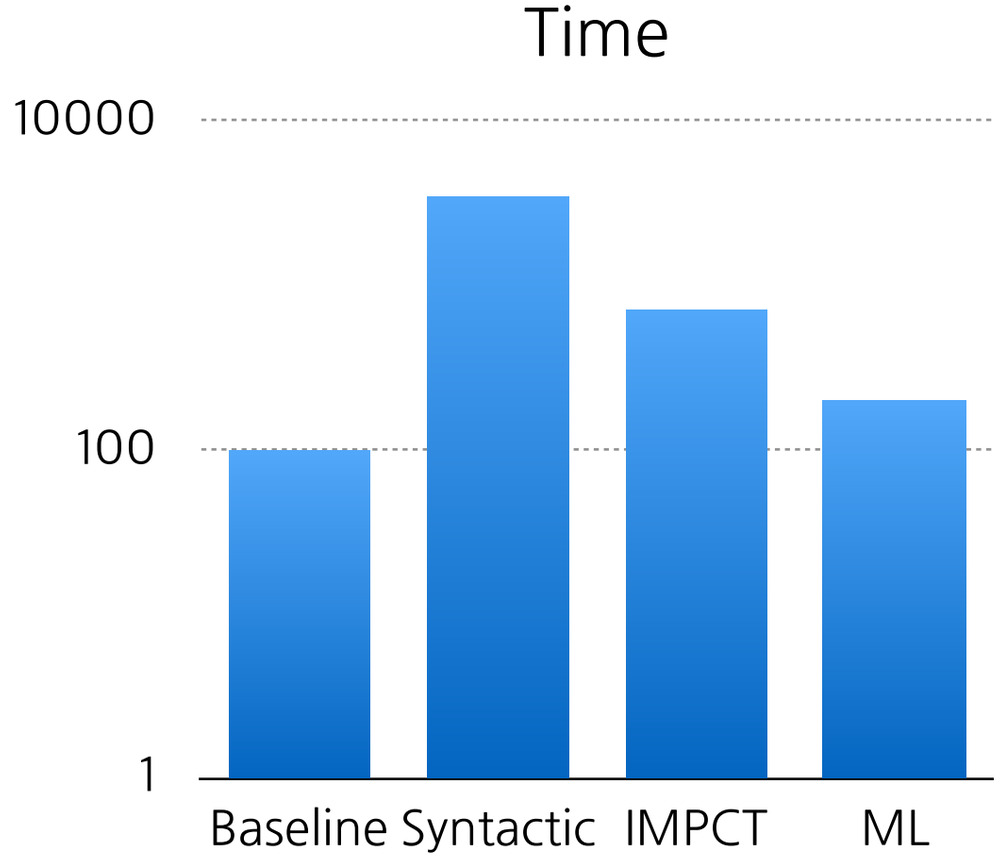
Sparrow
The Early Bird

(2016)

실험 결과

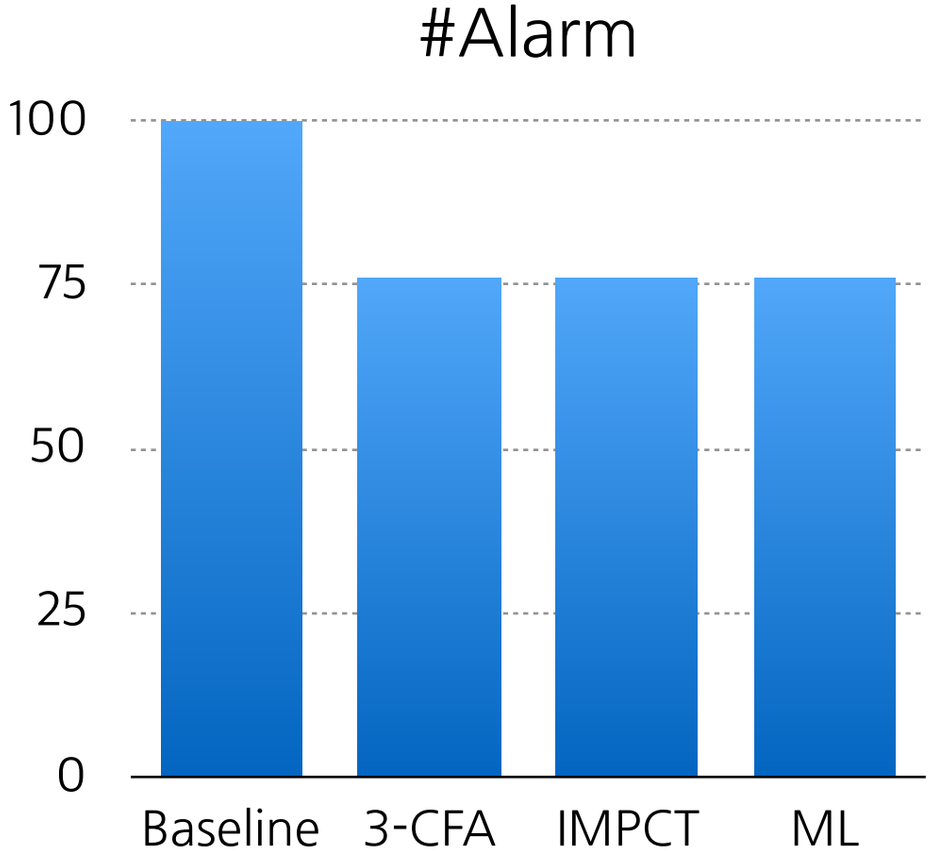
- 선별적으로 변수 관계를 추적하는 분석 (Selective Relational Analysis)

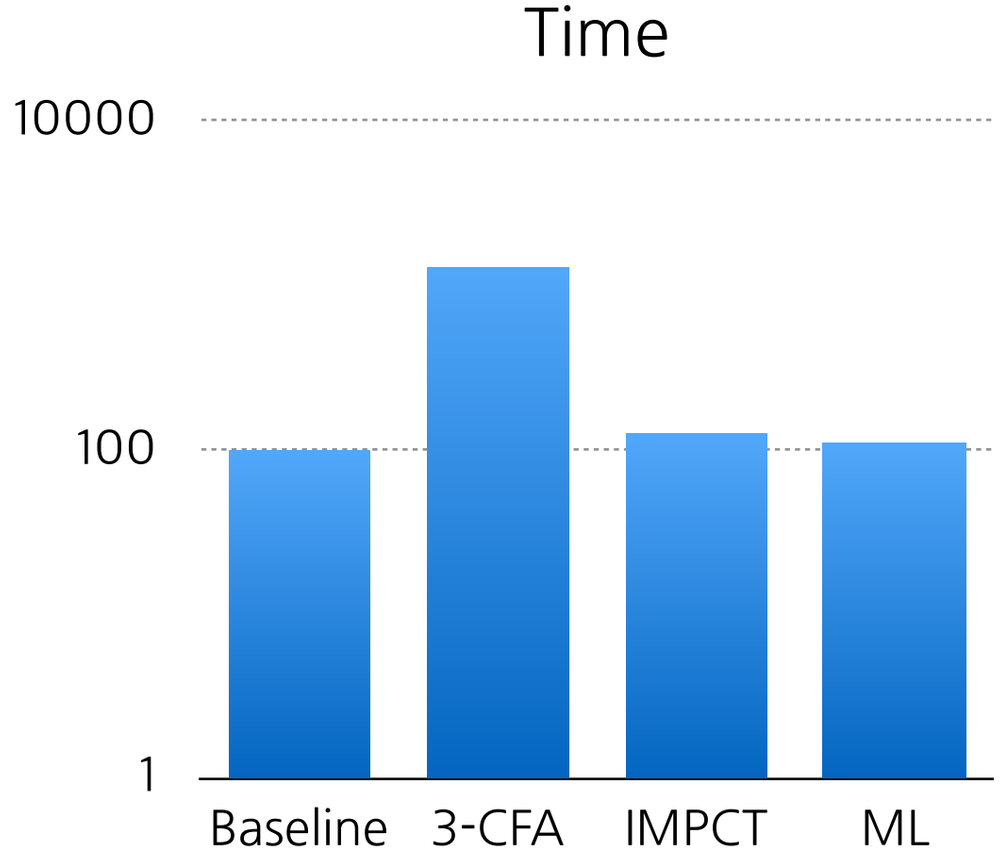




실험 결과

- 선별적으로 문맥을 구분하는 분석
  (Selective Context-sensitive Analysis)

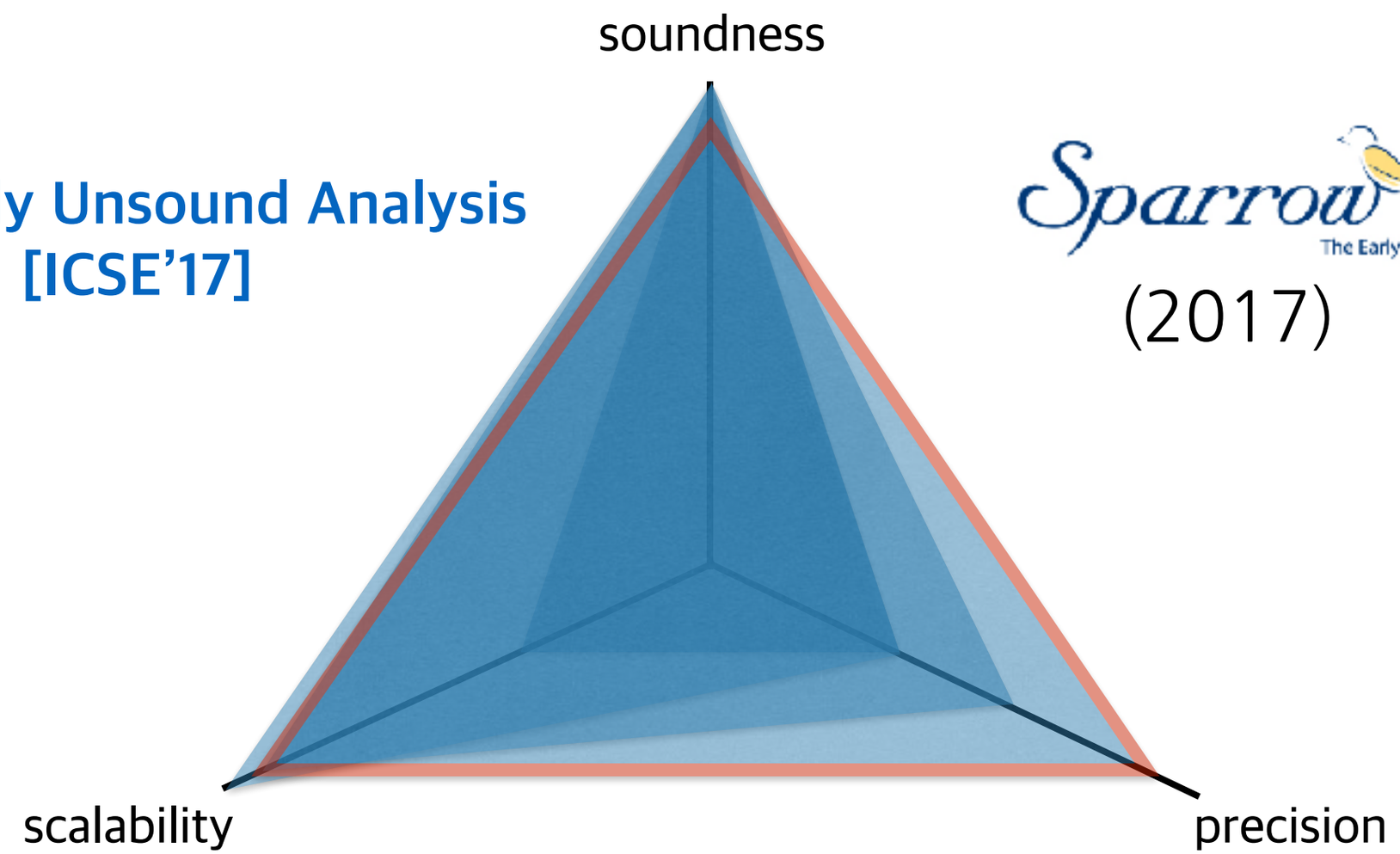




목표

- Sound, precise yet scalable static analysis

Selectively Unsound Analysis [ICSE'17]



Sparrow The Early Bird

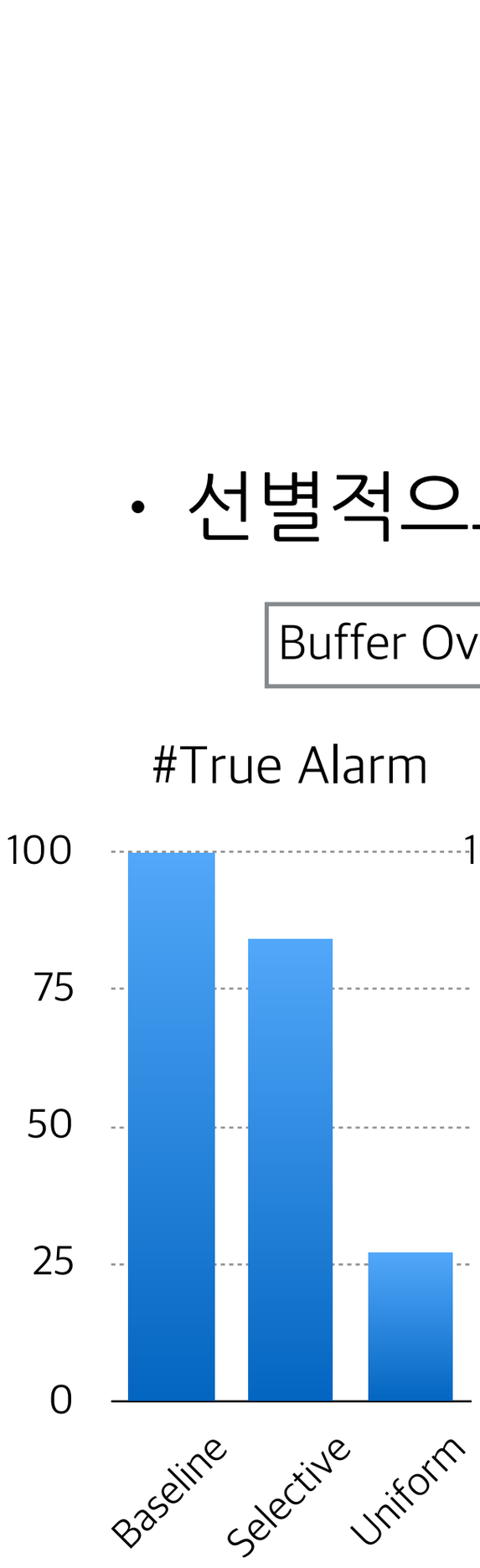
(2017)

실험 결과

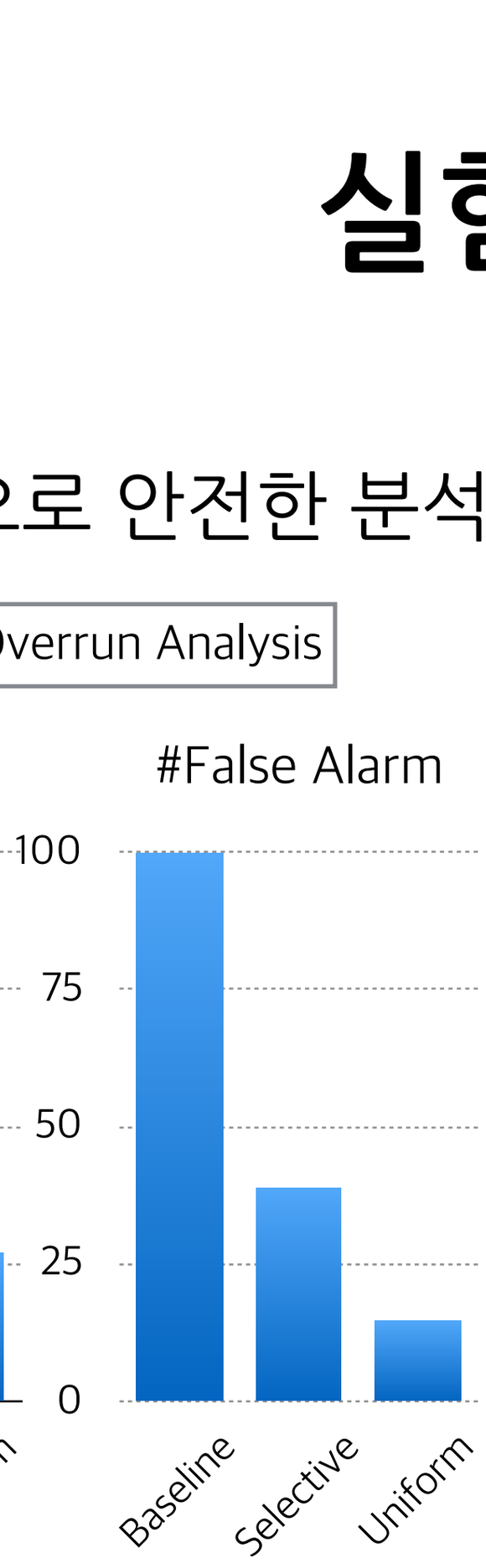
- 선별적으로 안전한 분석 (loop, lib call 안전성 조절)

Buffer Overrun Analysis

#True Alarm

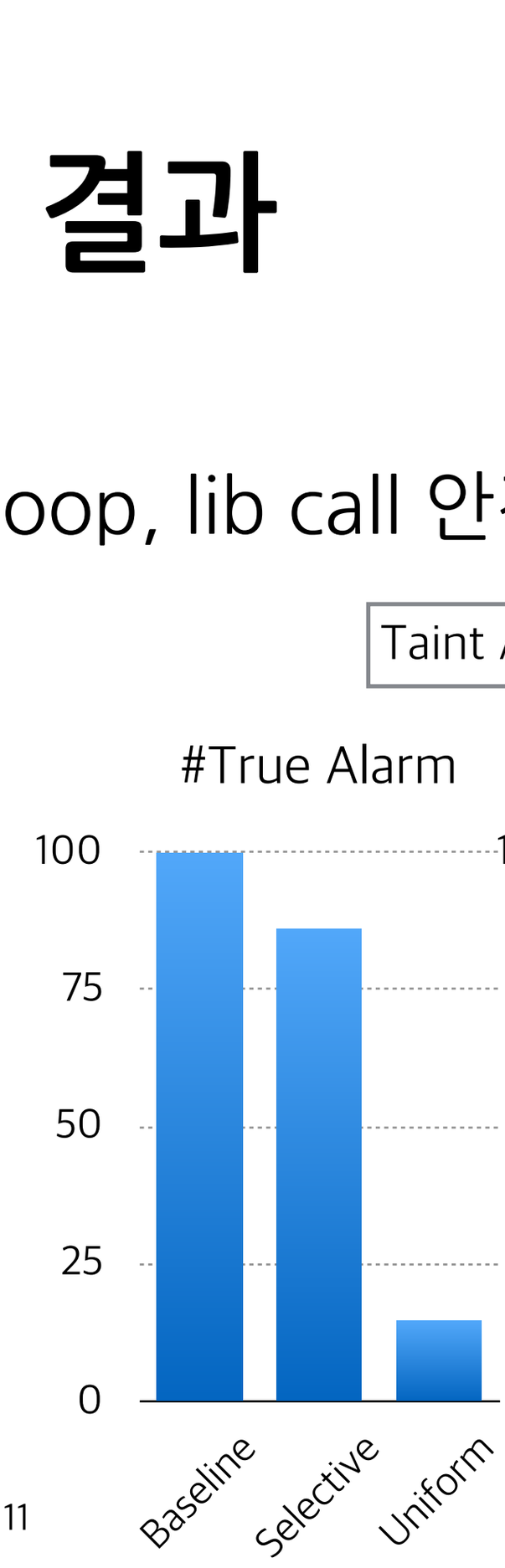


#False Alarm

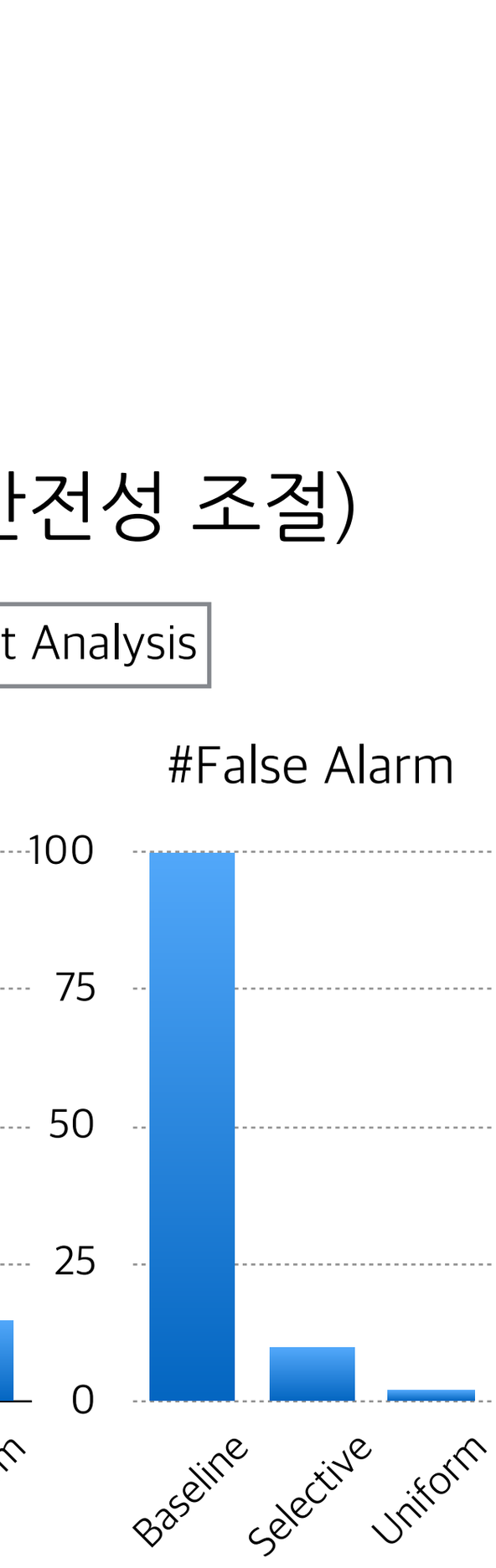


Taint Analysis

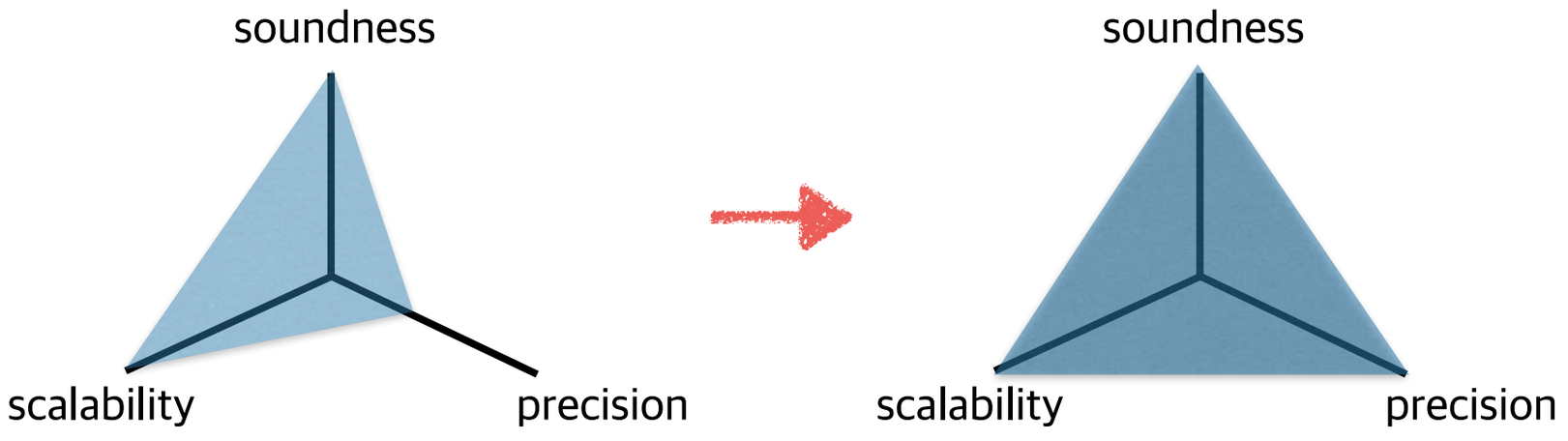
#True Alarm



#False Alarm



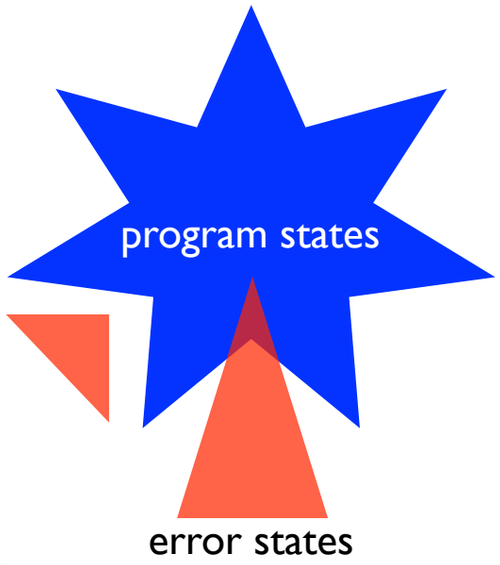
핵심 기술



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

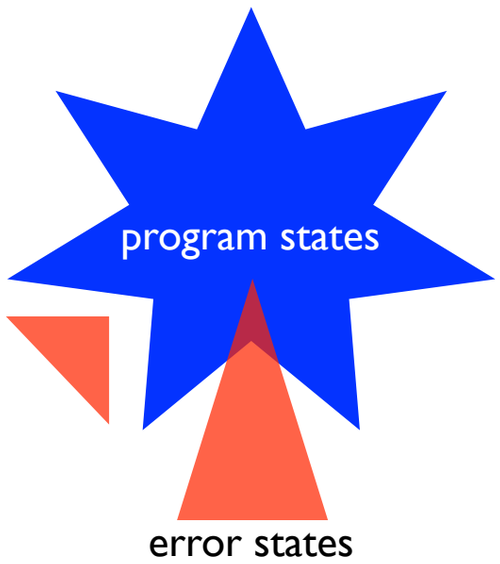
- 효율적인 정적 분석을 위해 적절한 요약을 찾기
  - 정적 분석 이론 기반 (예비 분석) / 통계 기반 (기계 학습)
  - 대상: 변수 관계, 문맥 (context), 흐름 (flow),
    안전성 (soundness), 등

핵심 기술



- 효율적인 정적 분석을 위해 적절한 요약을 찾기

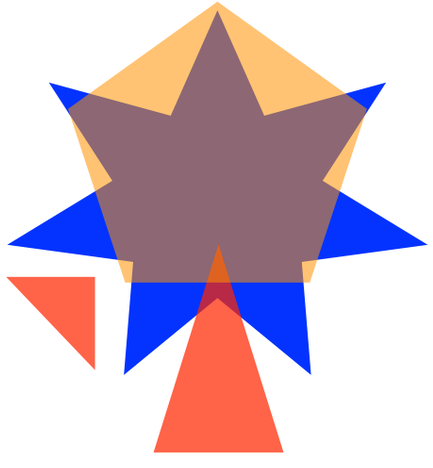
핵심 기술



- 효율적인 정적 분석을 위해 적절한 요약을 찾기

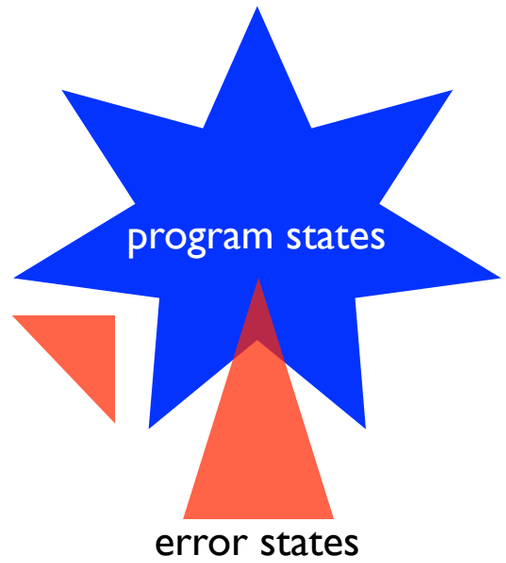
sound analysis

sound & precise analysis



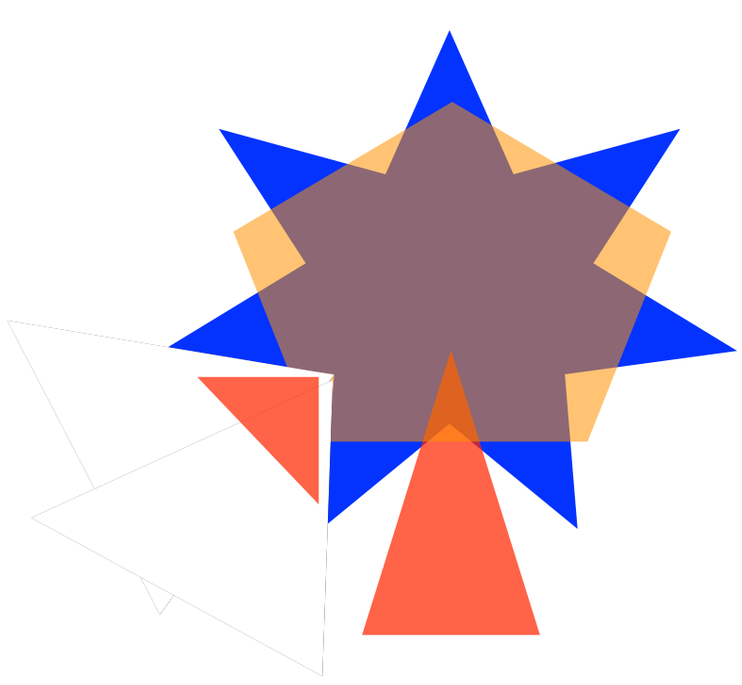
unsound analysis

핵심 기술



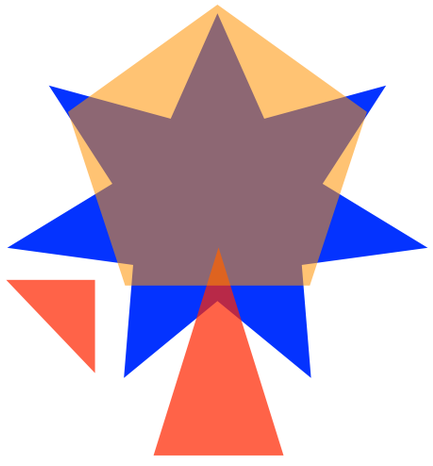
- 효율적인 정적 분석을 위해 적절한 요약을 찾기

sound analysis



selective analysis

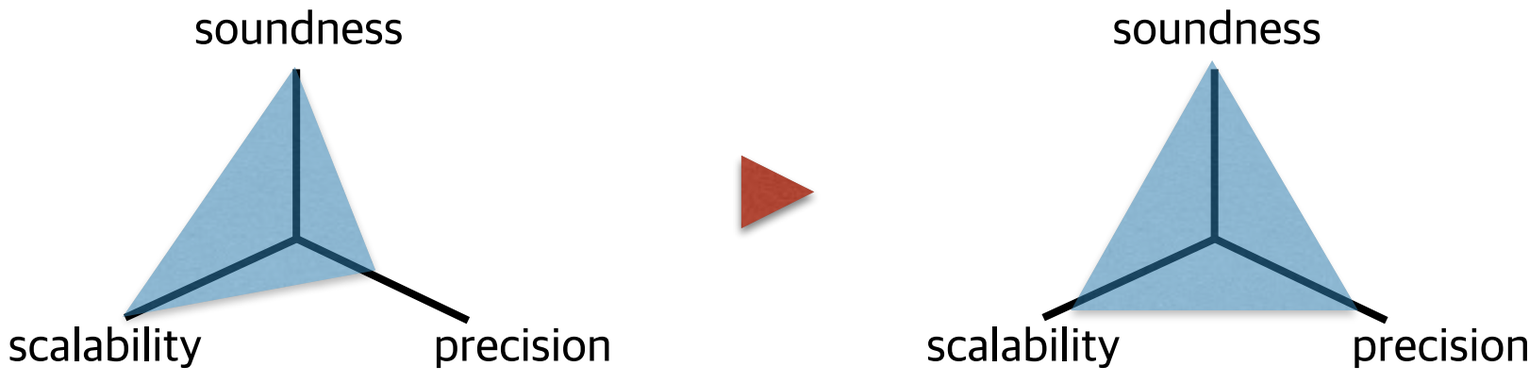
sound & precise analysis



unsound analysis

예비 분석을 이용하여 선별적으로 정확하게 정적 분석

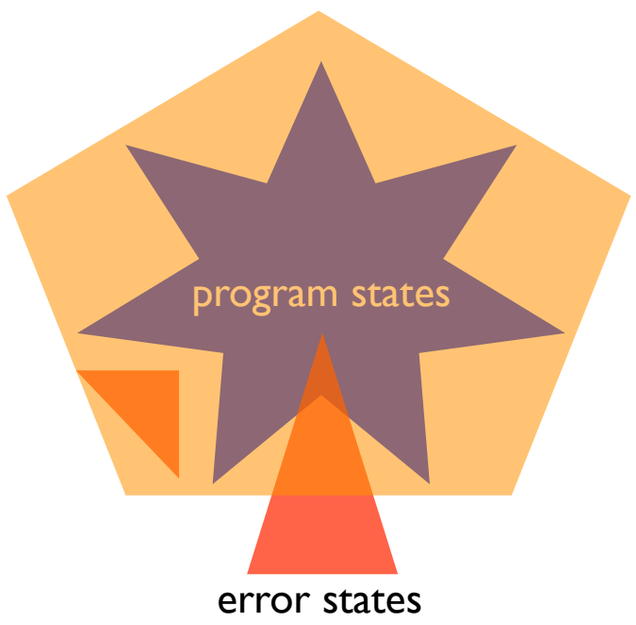
(Selective X-sensitive Analysis by Impact Pre-analysis)



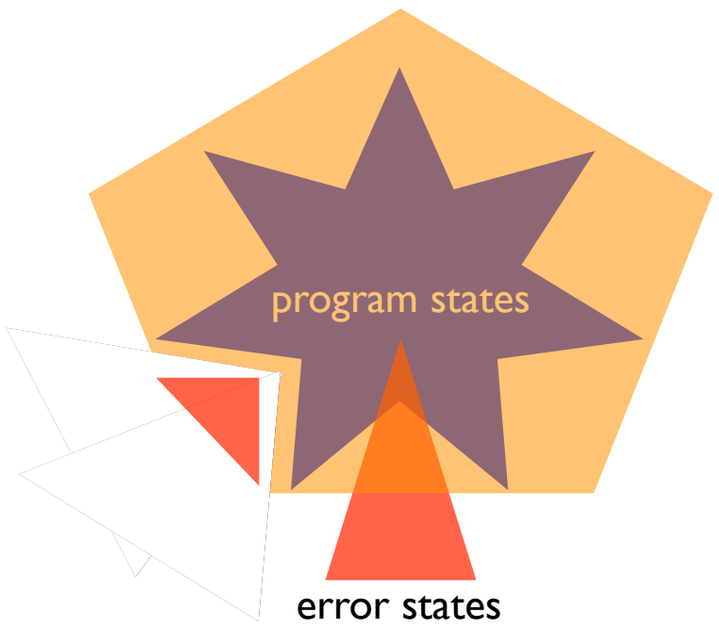
- Selective Context-Sensitivity Guided by Impact Pre-Analysis, PLDI'14
- Selective X-Sensitive Analysis Guided by Impact Pre-Analysis, TOPLAS'16
- Selective Conjunction of Context-sensitivity and Octagon Domain toward Scalable and Precise Global Static Analysis, SPE'17

선별적으로 정확한 분석

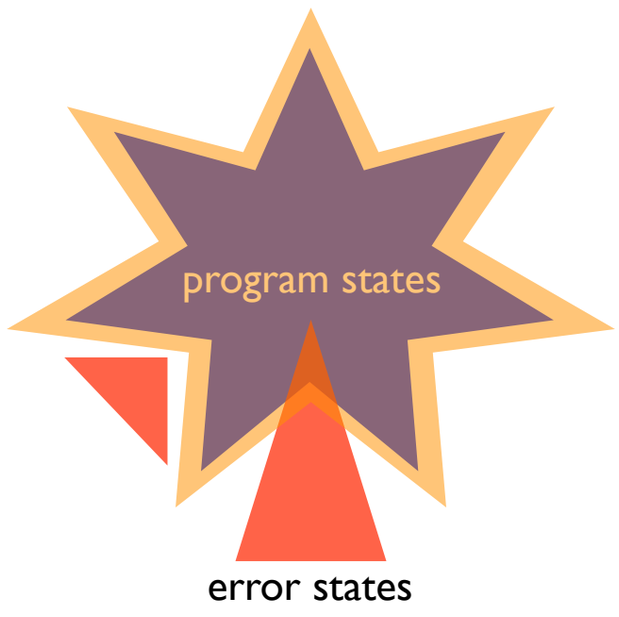
- 특정 X 를 필요한 곳에서만 정확하게 분석하는 방법
  - X : 문맥, 관계 등 정확성을 높이지만 비싼 기술



부정확한 분석



선별적으로 정확한 분석



정확한 분석

관계 분석

- 변수 사이의 관계를 특정한 형태로 분석
  - e.g.) octagon analysis : $(\pm x) - (\pm y) \leq c$

```
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  }
```

	a	b	c	i
a	0	∞	∞	∞
b	∞	0	∞	∞
c	∞	∞	0	∞
i	∞	∞	∞	0

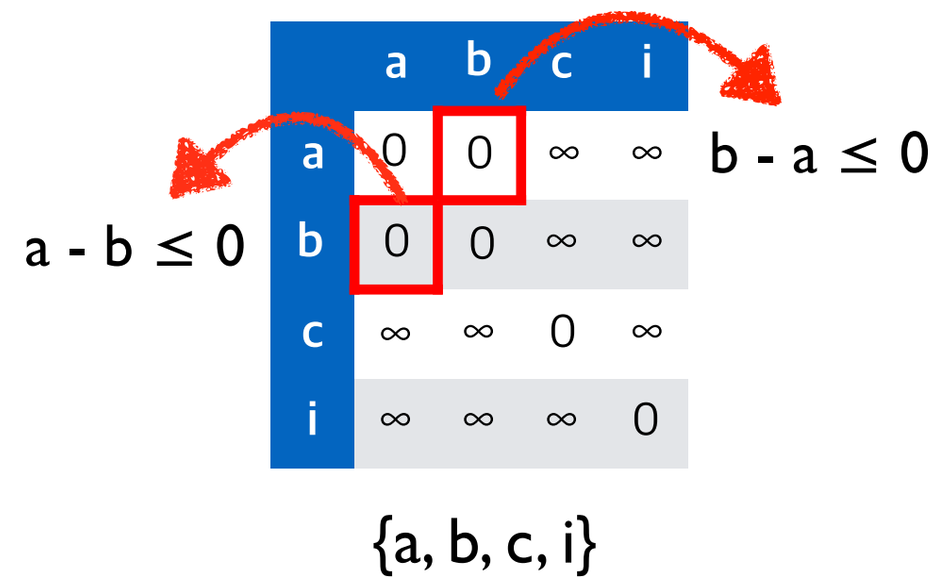
{a, b, c, i}

*Consider x-y ≤ c only, for simplicity

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  }
```



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  }
```

	a	b	c	i
a	0	0	∞	∞
b	0	0	∞	∞
c	∞	∞	0	∞
i	∞	∞	∞	0

c - a ≤ ∞

c - b ≤ ∞

a - c ≤ ∞

b - c ≤ ∞

{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  }
```

	a	b	c	i
a	0	0	∞	∞
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

i - b ≤ -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  }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

i - a ≤ -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  }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

i - c ≤ ∞

{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  }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

{a, b, c, i}

선별적 관계 분석

- 선별적으로 변수사이의 관계를 분석
  - 같은 묶음 (cluster) 안에 있는 변수 사이 관계만

```
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  }
```

	a	b	i
a	0	0	-1
b	0	0	-1
i	∞	∞	0

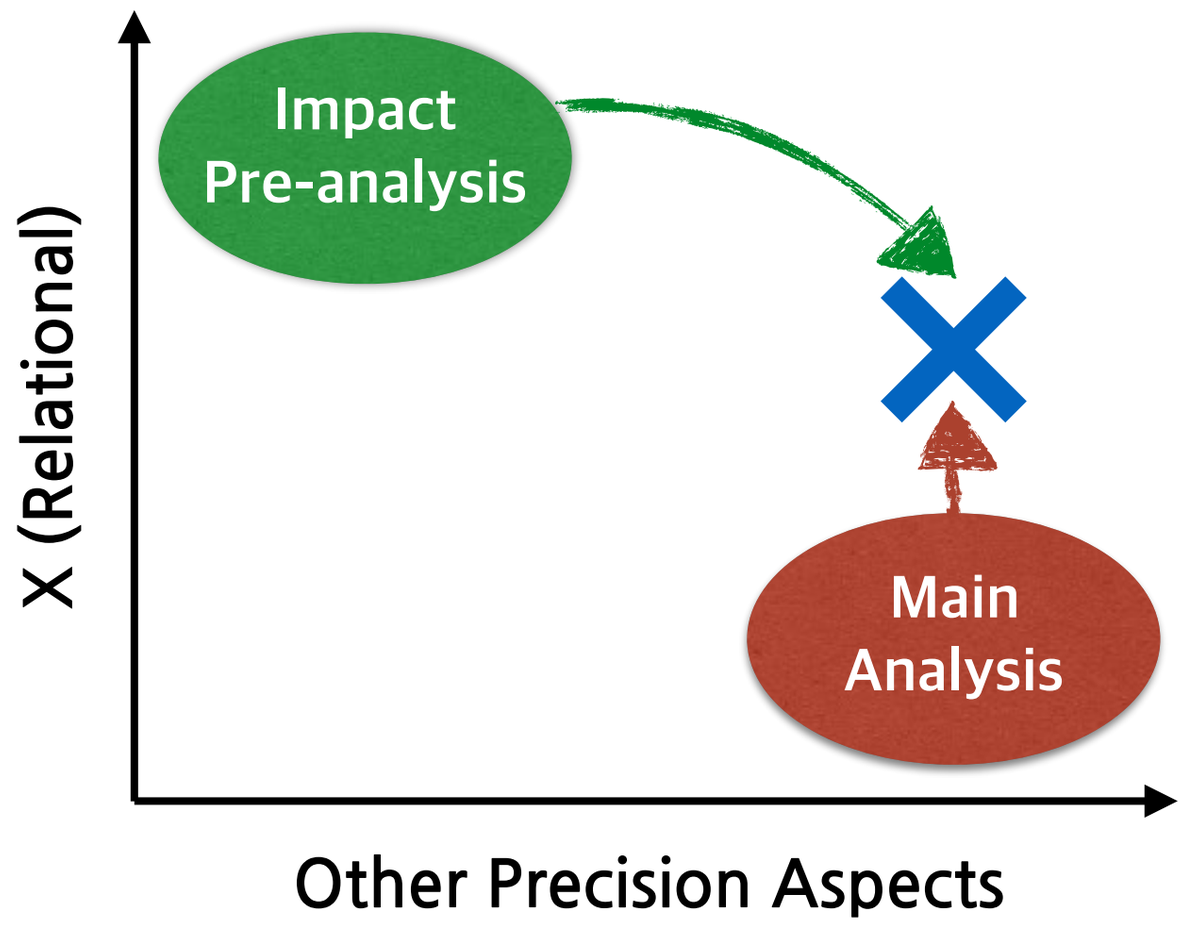
+

$-\infty \leq c \leq +\infty$

{a,b,i} {c}

예비 분석

- X 의 효과를 가늠하는 정적 분석
  - X는 가장 정확하게 + 나머지는 과감히 요약



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: }
```

i - a <=★

i - c <= T

	a	b	c	i
a	★	★	T	★
b	★	★	T	★
c	T	T	★	T
i	T	T	T	★

T 무한할 수도 ($\mathbb{Z} \cup \{+\infty\}$)

★ 반드시 유한 ($\mathbb{Z}$)

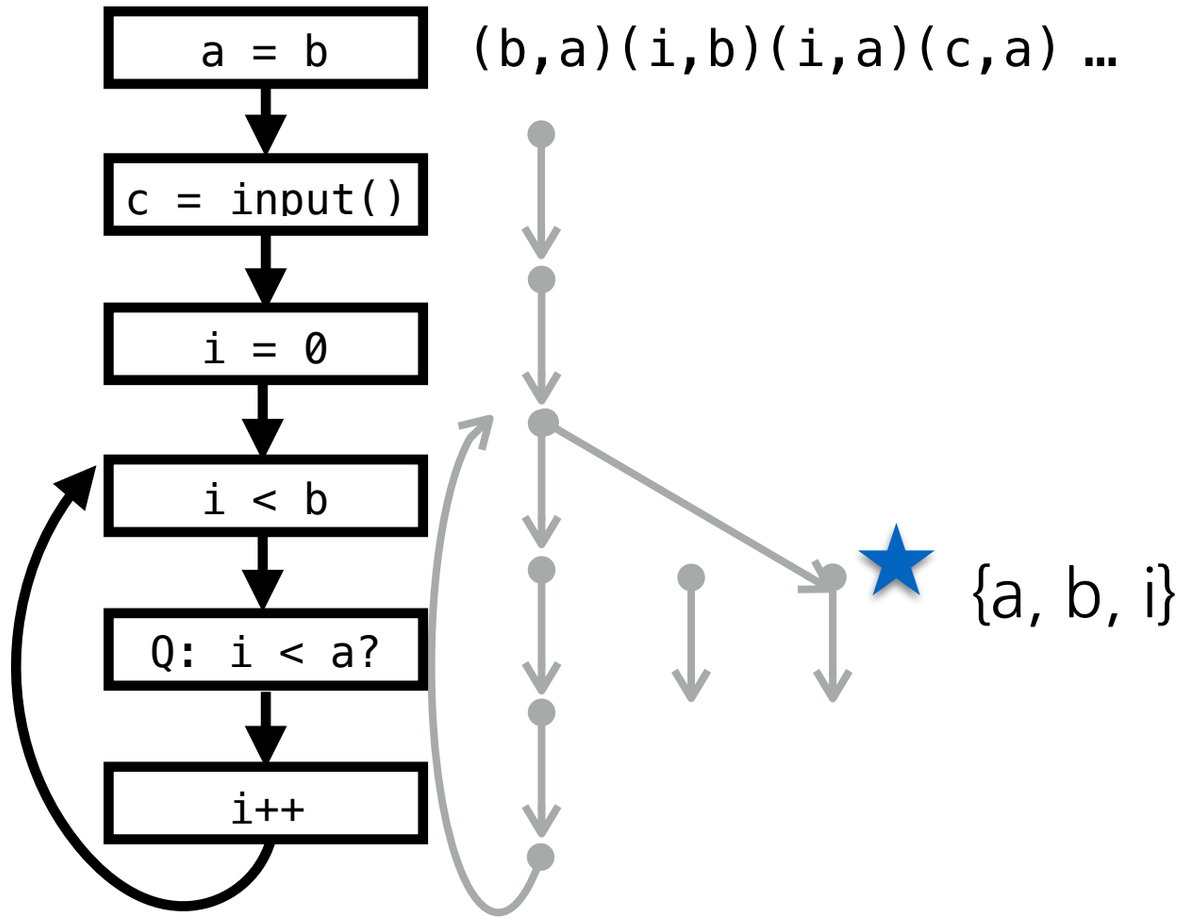
변수 묶음

- 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: }
```

i - a <=★

i - c <= T



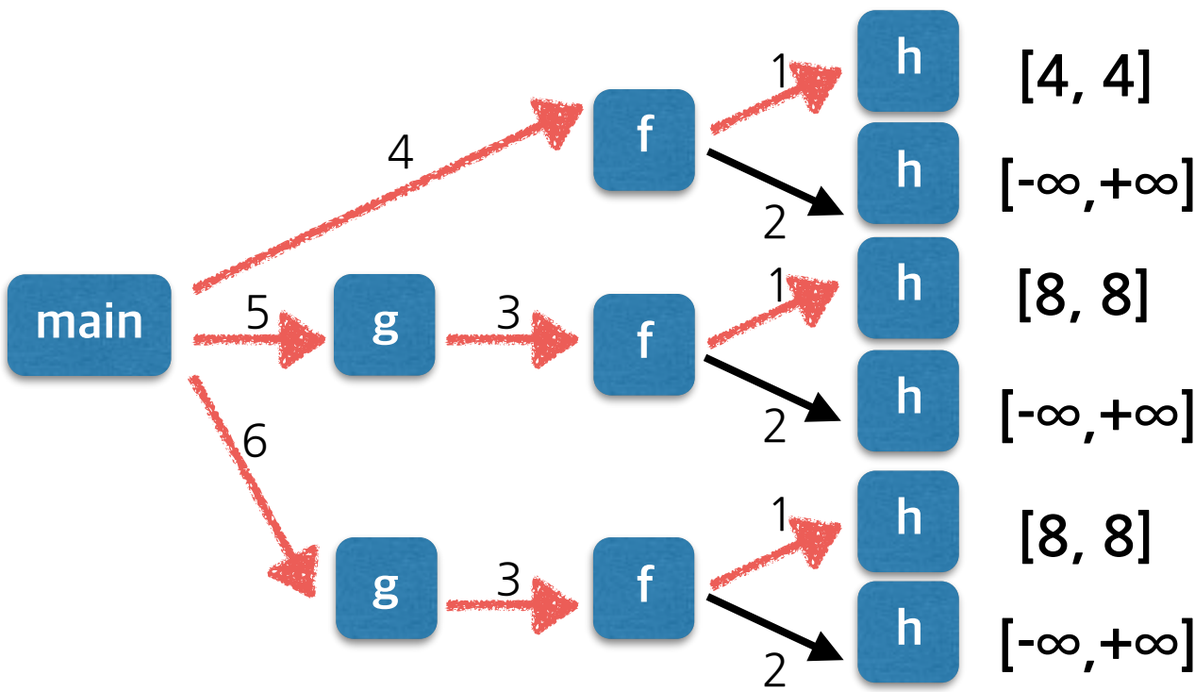
실험 결과

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

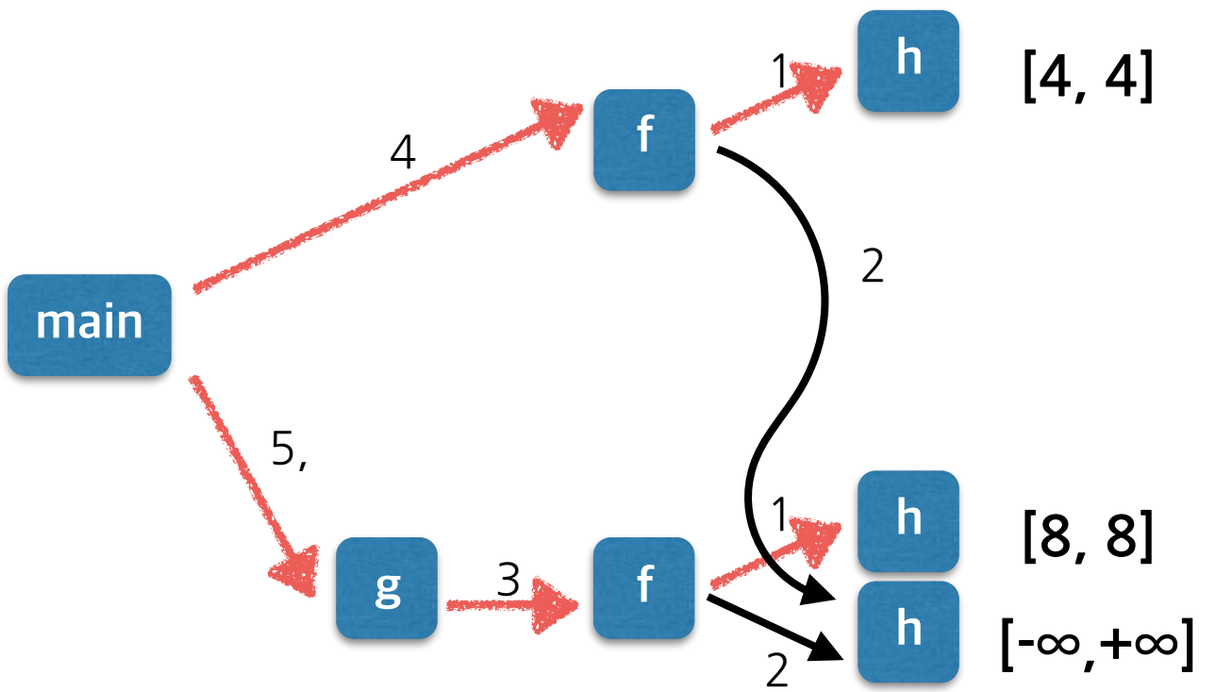
Program	LOC	#Variable	#Query	Syntactic Packing Approach				Our Selective Relational Analysis						Comparison	
				proven	time	mem	pack	proven	pre	main	total	mem	pack	Precision	Time
calculator-1.0	298	197	10	2	0.3	63	18 (7.3)	10	0.1	0.1	0.2	52	3 (3.6)	+8	-33.3%
spell-1.0	2,213	531	16	1	4.8	109	119 (7.7)	16	1.7	0.7	2.4	63	6 (11.0)	+15	-50.0%
barcode-0.96	4,460	2,002	37	16	11.8	221	276 (8.1)	37	12.2	18.3	30.5	100	12 (25.0)	+21	158.5%
httptunnel-3.3	6,174	1,908	28	16	26.0	220	454 (7.0)	26	10.8	4.5	15.3	105	8 (5.8)	+10	-41.2%
bc-1.06	13,093	2,194	10	2	247.1	945	606 (7.8)	9	82.3	35.0	117.3	212	4 (4.0)	+7	-52.5%
tar-1.17	20,258	5,332	17	7	1,043.2	1,311	1,259 (7.5)	17	598.5	63.3	661.8	384	7 (3.9)	+10	-36.6%
less-382	23,822	4,482	13	0	3,031.5	1,439	1,017 (6.3)	13	2,253.2	596.2	2,849.4	955	8 (6.3)	+13	-6.0%
a2ps-4.14	64,590	16,531	11	0	29,479.3	2,304	2,608 (7.8)	11	2,223.5	518.2	2,741.7	909	6 (6.7)	+11	-90.7%
Total	135,008	33,177	142	44	33,840.3	6,611		139	5,182.3	1,236.3	6,418.6	2,780		+95	-81.0%

예) 문맥 구분 (Context-sensitivity)

```
   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();
   }
```



모든 문맥 구분 분석

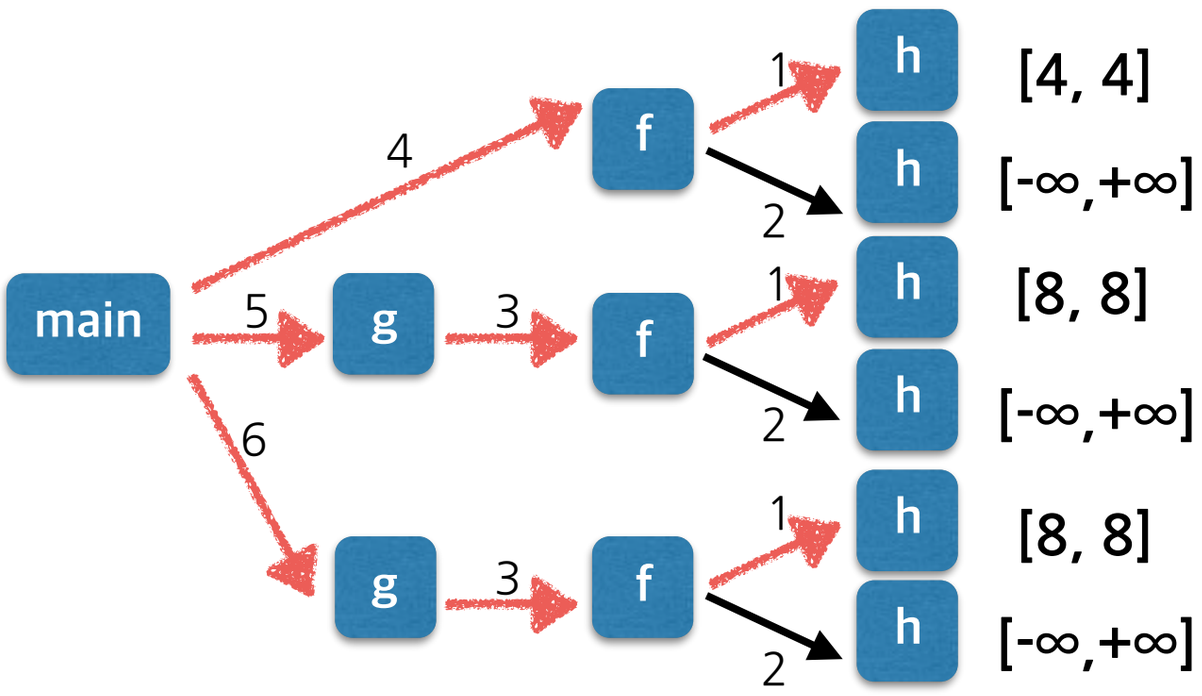


선별적 문맥 구분 분석

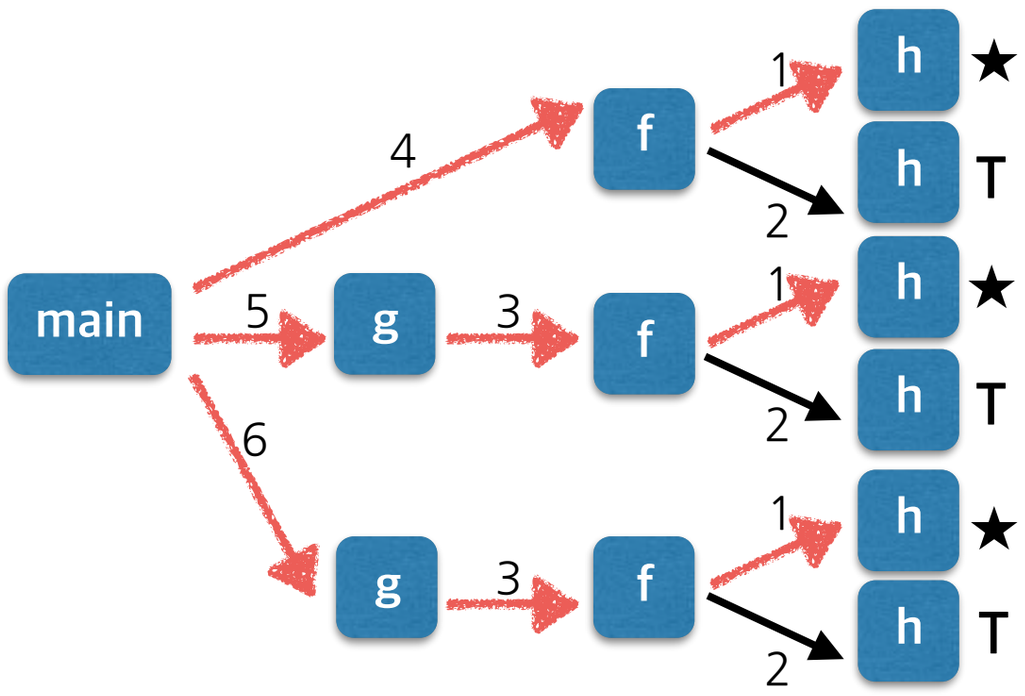
예) 문맥을 위한 예비 분석

T 모든 인터벌

★ 음이 아닌 인터벌 (e.g. [1,5], [0, ∞])



모든 문맥 구분 분석



예비 분석

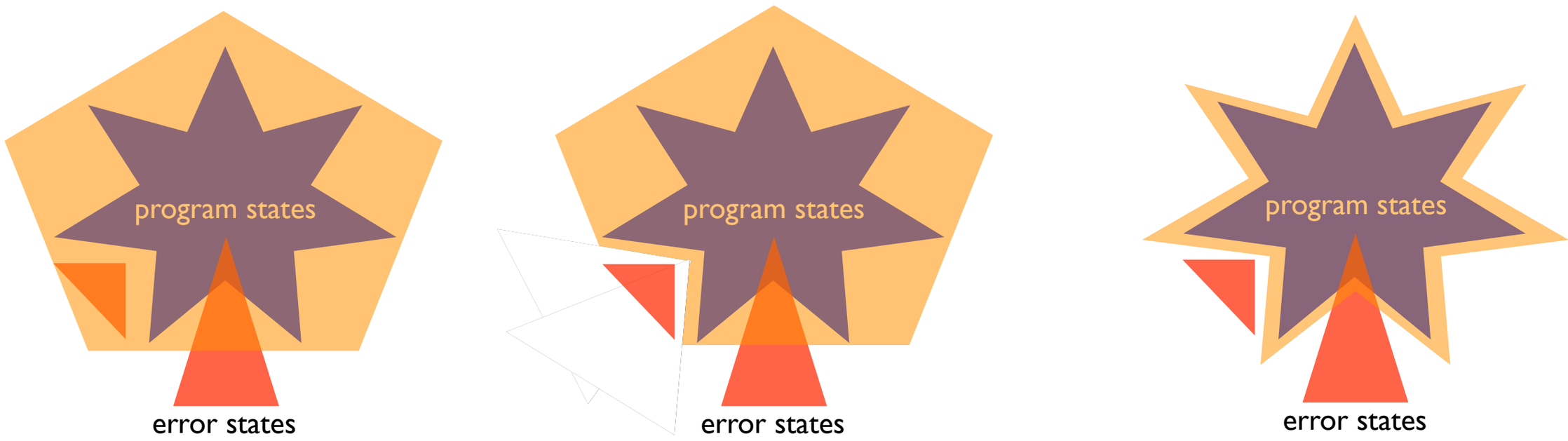
실험 결과

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

Program	LOC	Proc	Context-Insensitive		Our Selective Context-Sensitive Analysis						Alarm	Overhead	
			#alarm	time	#alarm	pre	main	total	#selected call-sites	⇝	reduction	pre	main
spell-1.0	2,213	31	58	0.6	30	0.1	0.8	0.9	25 / 124 (20.2 %)	3	48.3%	16.7%	33.3%
bc-1.06	13,093	134	606	14.0	483	1.9	14.3	16.2	29 / 777 (3.7 %)	2	20.3%	13.6%	2.1%
tar-1.17	20,258	222	940	42.1	799	5.4	41.8	47.2	51 / 1213 (4.2 %)	3	15.0%	12.8%	−0.7%
less-382	23,822	382	654	123.0	562	3.3	163.1	166.4	51 / 1,522 (3.4 %)	4	14.1%	2.7%	32.6%
sed-4.0.8	26,807	294	1,325	107.5	1,238	7.4	110.2	117.6	25 / 868 (2.9 %)	3	6.6%	6.9%	2.5%
make-3.76	27,304	191	1,500	84.4	1,028	7.1	99.1	106.2	67 / 1,050 (6.4 %)	3	31.5%	8.4%	17.4%
grep-2.5	31,495	153	735	12.1	653	2.4	13.5	15.9	33 / 530 (6.2 %)	3	11.2%	19.8%	11.6%
wget-1.9	35,018	434	1,307	69.0	942	12.5	69.6	82.1	79 / 1,973 (4.0 %)	5	27.9%	18.1%	0.9%
a2ps-4.14	64,590	980	3,682	118.1	2,121	29.5	148.2	177.7	237 / 2,450 (9.7 %)	9	42.4%	25.0%	25.5%
bison-2.5	101,807	1,427	1,894	136.3	1,742	34.6	138.8	173.4	173 / 2,038 (8.5 %)	4	8.0%	25.4%	1.8%
Total	346,407	4,248	12,701	707.1	9,598	104.2	799.4	903.6	770 / 12,545 (6.1 %)		**24.4%**	**14.7%**	**13.1%**

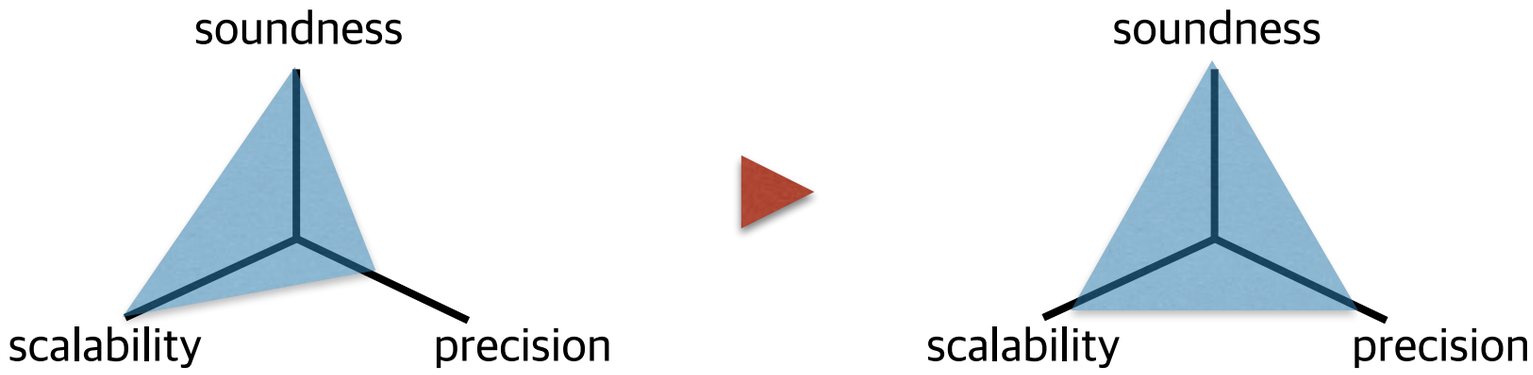
정리

- 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



선별적 분석

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 :   amazon …

Big Data

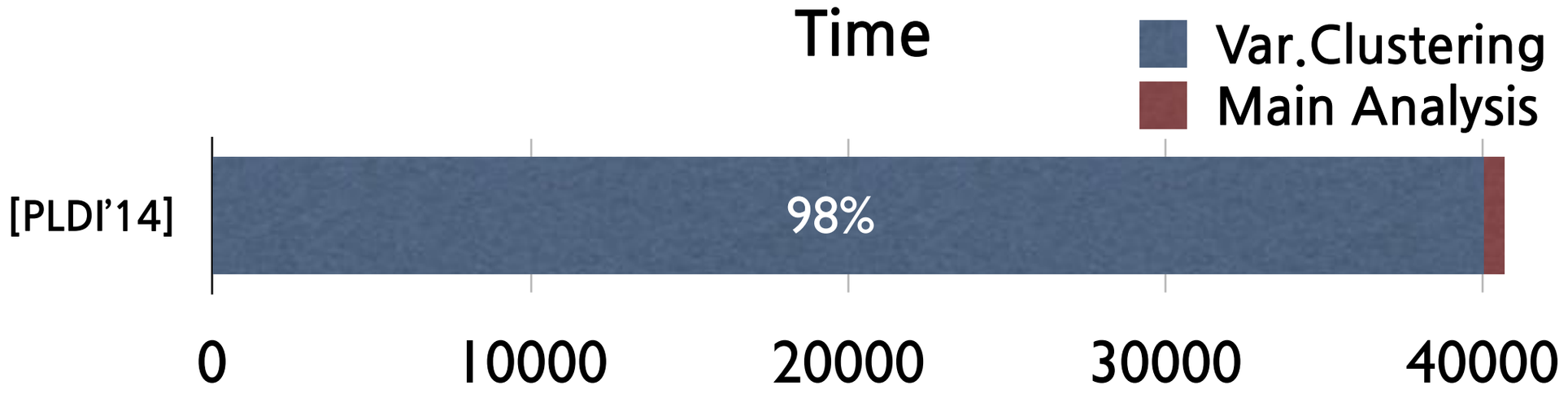
+

Static Analyzer

PLDI'14

문제

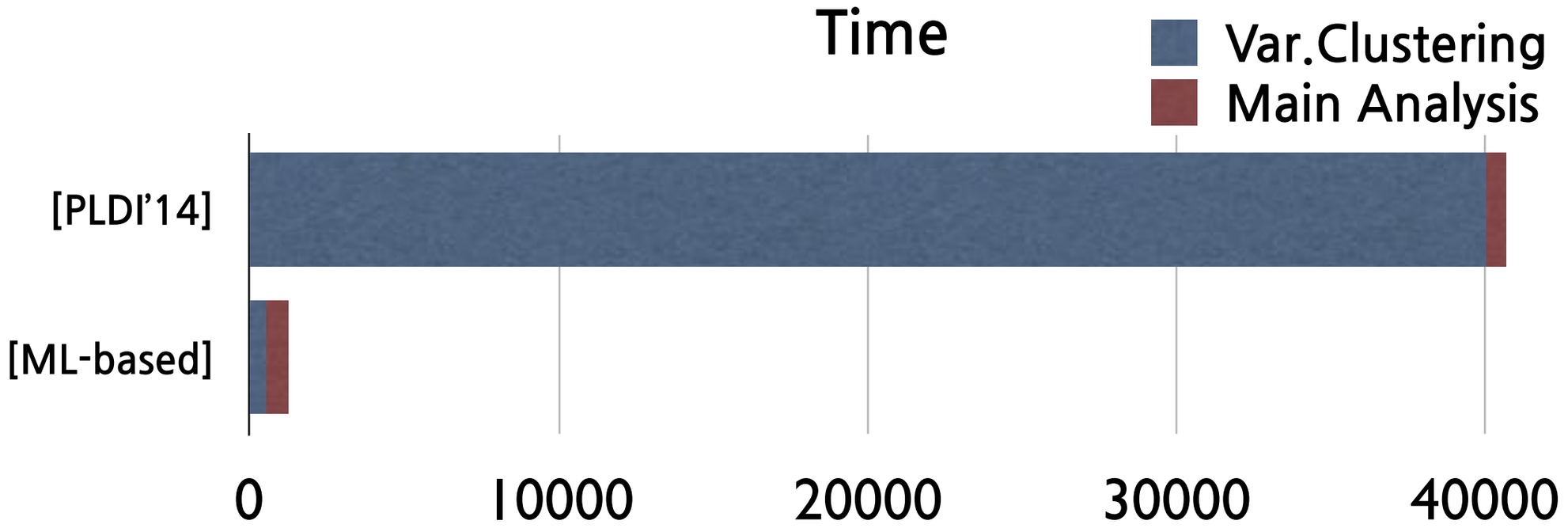
- 예비 분석도 여전히 버거운 상황 (예. 관계 분석)
  - 모든 관계를 추적하는 예비 분석: online estimator
  - e.g.) 17 open source benchmarks (~100KLOC)



SAS'16

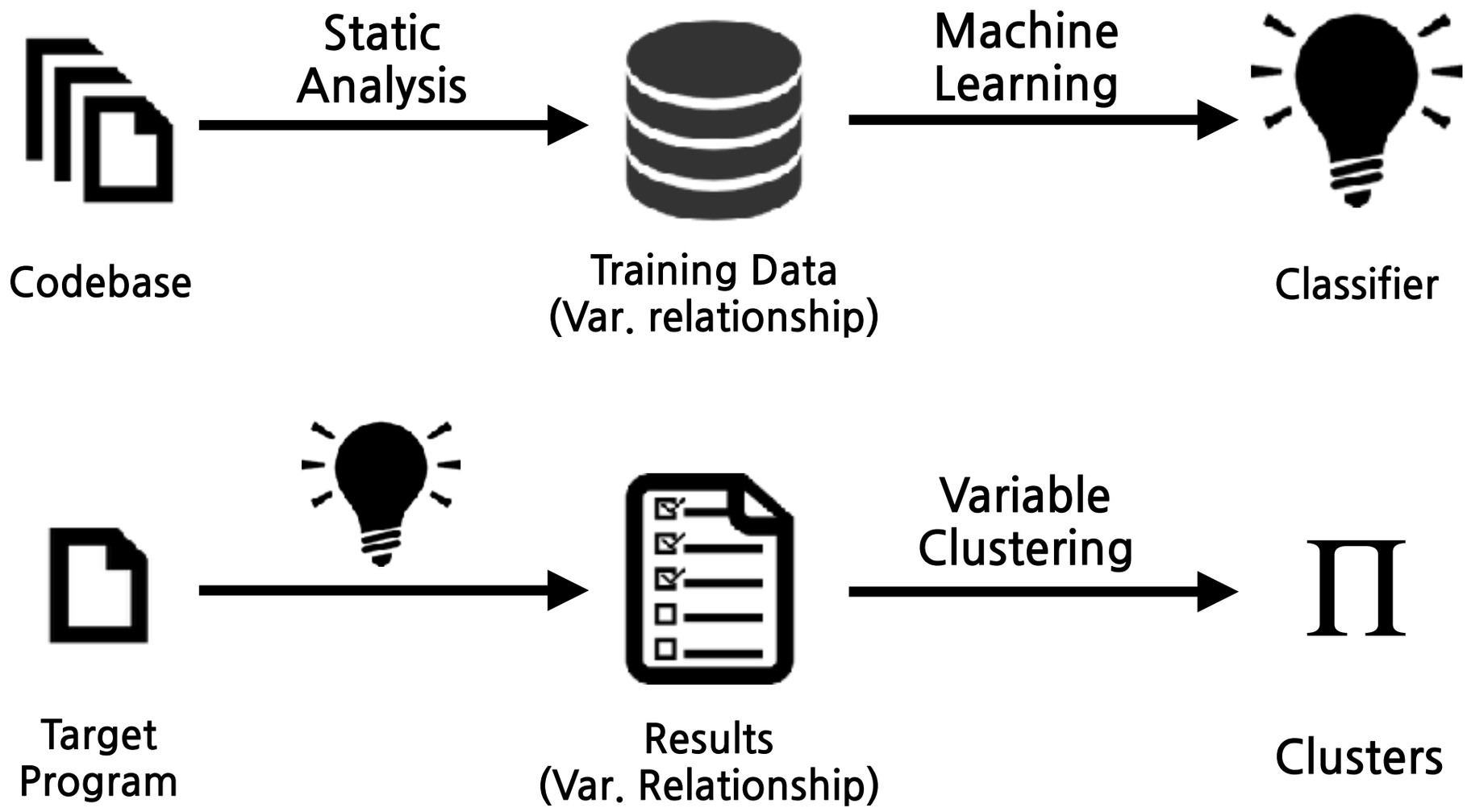
해결책

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습
  - 모든 관계를 추적하는 예비 분석: offline teacher
- 33 배 빠르고 정확도는 비슷



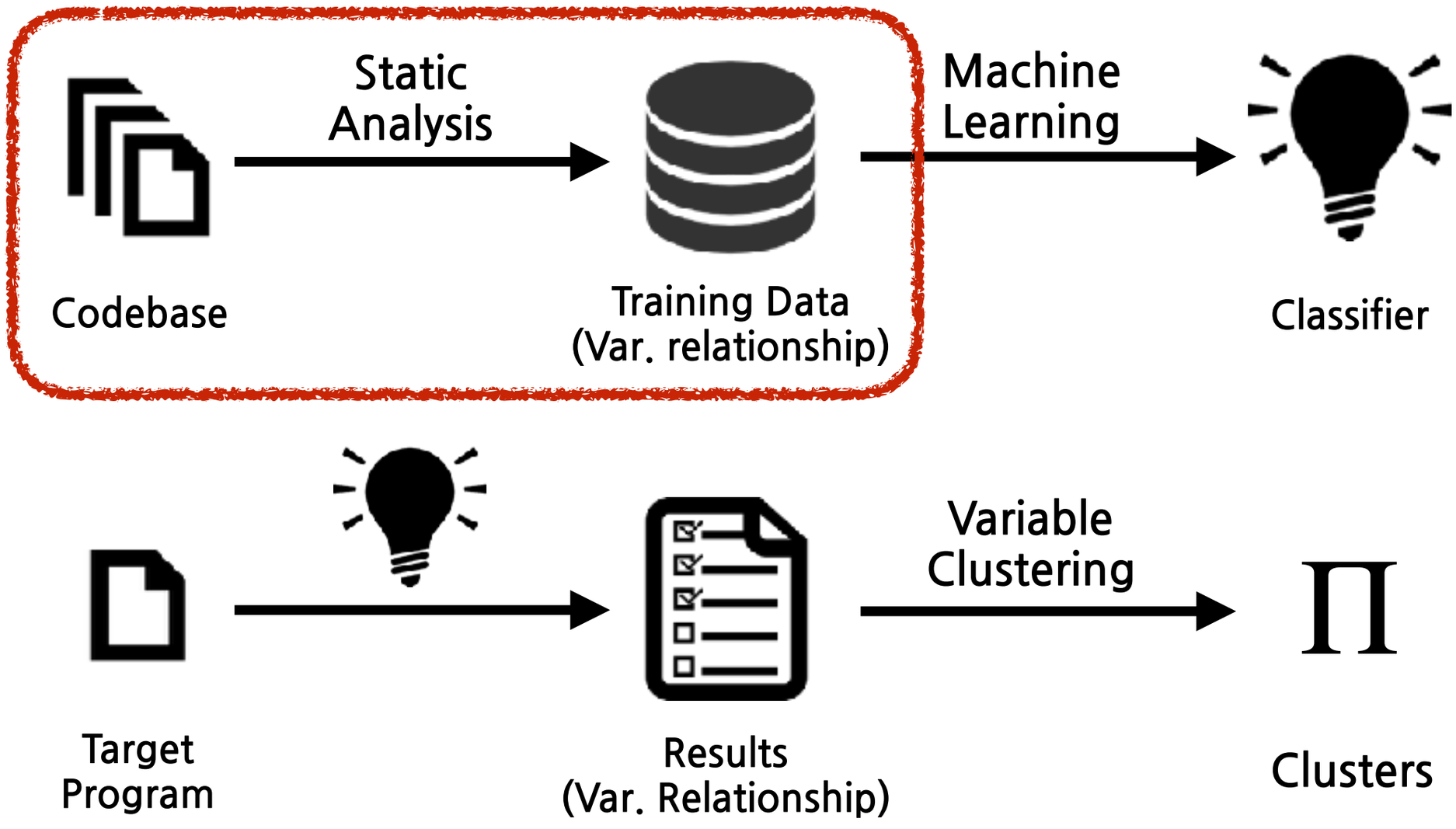
큰 그림

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습



큰 그림

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습



학습 데이터

- 꼬리표 {⊕, ⊖} 가 붙은 변수 쌍
  - ⊕: 정확 (< +∞), ⊖: 부정확 (= +∞)

```
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  }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

Octagon Analysis

⊕ : {(a,b), (a,i), (b,a) …}

⊖ : {(a,c), (b,c), (c,a) …}

학습 데이터

- 예비 분석 결과로부터 자동 생성 [PLDI'14]
  - 모든 관계 분석, 본 분석보다는 적은 비용

```
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  }
```

	a	b	c	i
a	0	0	∞	-1
b	0	0	∞	-1
c	∞	∞	0	∞
i	∞	∞	∞	0

Octagon Analysis

$$\gamma(\bigstar) = \mathbb{Z}$$
$$\gamma(\top) = \mathbb{Z} \cup \{+\infty\}$$

$\oplus$: {(a,b), (a,i), (b,a) …}

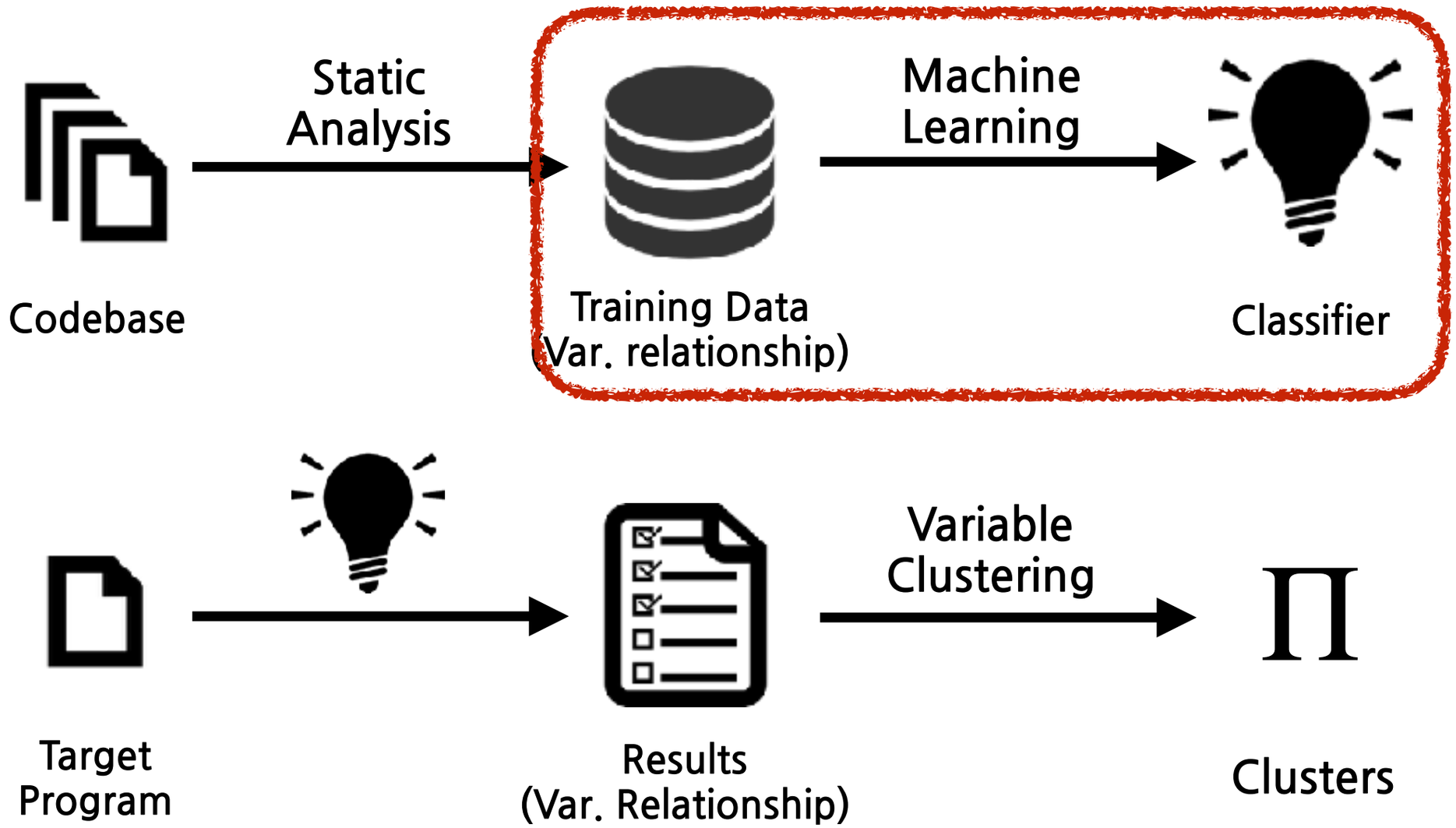
$\ominus$: {(a,c), (b,c), (c,a) …}

	a	b	c	i
a	★	★	T	★
b	★	★	T	★
c	T	T	★	T
i	T	T	T	★

Impact Pre-analysis

큰 그림

- 빅데이터로부터 필요한 변수 관계를 선별하는 전략 학습



특질 (feature)

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

(Positive situations for Octagon)
- `x=y+k` or `y=x+k`
- `x<=y+k` or `y<=x+k`
- `x=malloc(y)` or `y=malloc(x)`
- `x[y]` or `y[x]`
- …

(Negative situations for Octagon)
- `x=cy` or `y=cx (c != 1)`
- `x=yz` or `y=xz`
- `x=y/z` or `y=x/z`
- …

(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
- …

(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
- …

특질 (feature)

- 특질의 중요도 (Gini Index 로 측정)
  - 부정 & 보편 > 긍정 & 특수

(Positive situations for Octagon)
- `x=y+k` or `y=x+k`
- `x<=y+k` or `y<=x+k`
- `x=malloc(y)` or `y=malloc(x)`
- `x[y]` or `y[x]`
- …

(Negative situations for Octagon)
- `x=cy` or `y=cx (c != 1)`
- `x=yz` or `y=xz`
- `x=y/z` or `y=x/z`
- …

(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
- …

(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
- …

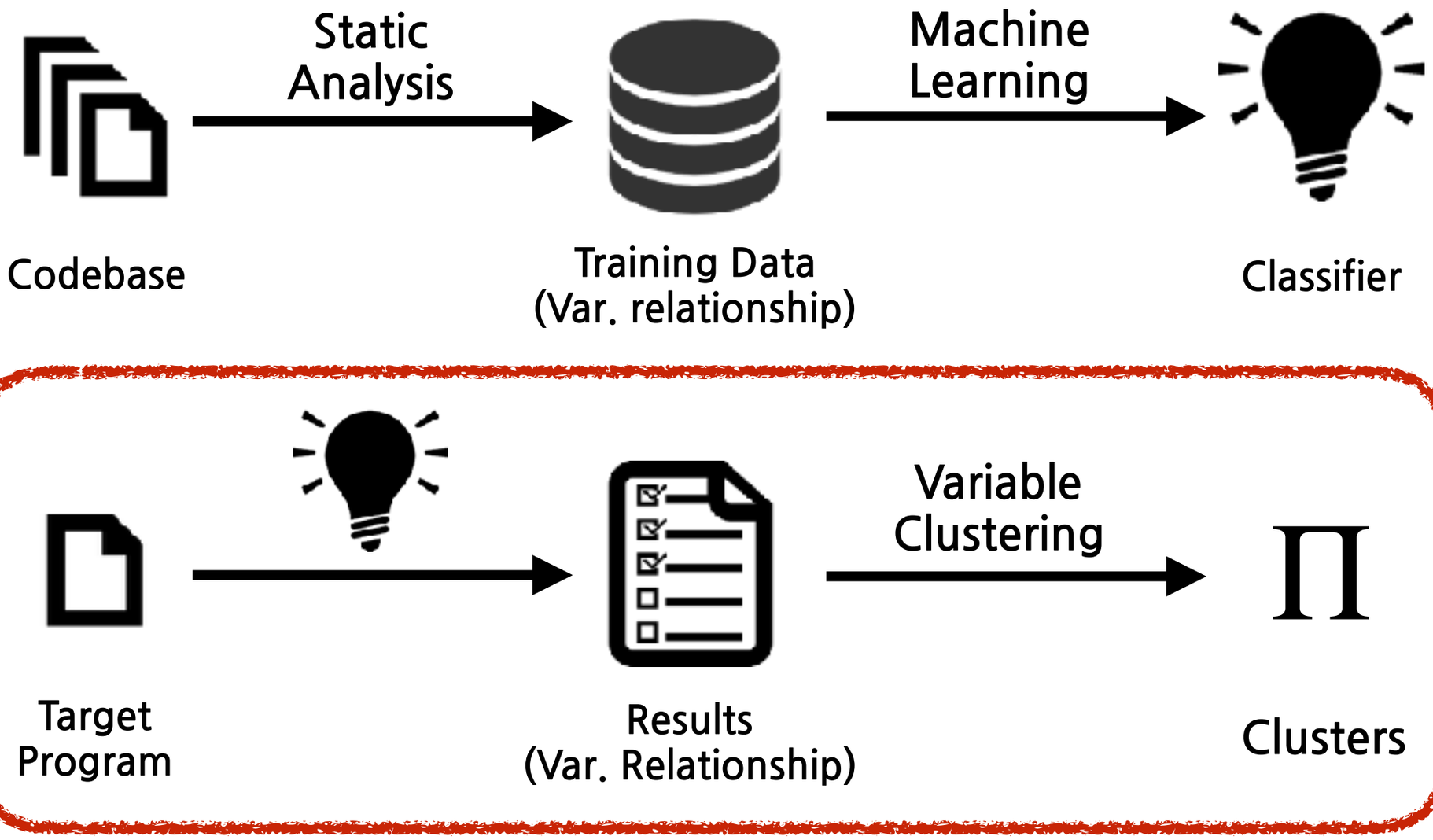
*Top 5 most important features

분류기 (Classifier)

- 변수 쌍 분류기 $\mathcal{C} : Var \times Var \to \{\oplus, \ominus\}$ 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

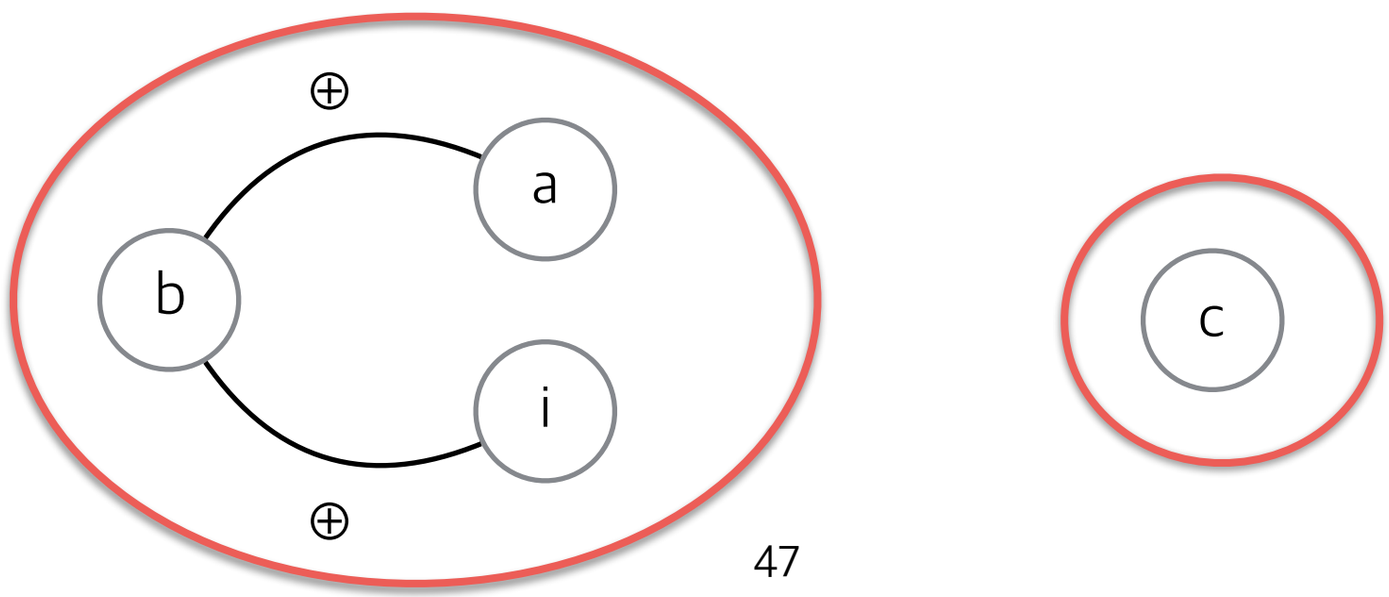


변수 묶음 전략

- ⊕-표 변수쌍을 같은 묶음에
  - transitive 관계도 자연스럽게 포함

```
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   }
```

	C(x,y)
(a,b)	⊕
(a,i)	⊖
(b,i)	⊕
(a,c)	⊖
…	…



실험 결과

- Effectiveness (leave-one-out cross validation)

Program	LOC	#Abs.Loc.	# Alarms			Time(s)		
			Itv	Impt	ML	Itv	Impt	ML
brutefir	103	54	4	0	0	0	0	0
consol	298	165	20	10	10	0	0	0
id3	512	527	15	6	6	0	0	1
spell	2,213	450	20	8	17	0	1	1
mp3rename	2,466	332	33	3	3	0	1	1
irmp3	3,797	523	2	0	0	1	2	3
barcode	4,460	1,738	235	215	215	2	9	6
httptunnel	6,174	1,622	52	29	27	3	35	5
e2ps	6,222	1,437	119	58	58	3	6	3
bc	13,093	1,891	371	364	364	14	252	16
less	23,822	3,682	625	620	625	83	2,354	87
bison	56,361	14,610	1,988	1,955	1,955	137	4,827	237
pies	66,196	9,472	795	785	785	49	14,942	95
icecast-raptor	68,564	6,183	239	232	232	51	109	107
dico	76,378	8,889	2,156	2,148	2,148	242	17,844	345
lsh	84,333	4,349	402	396	396	38	156	51
Total	110,898	18,880	330	325	325	33	139	251
			7,406	7,154	7,166	656	40,677	1,207

실험 결과

- Effectiveness (leave-one-out cross validation)

Program	LOC	#Abs.Loc.	# Alarms			Time(s)		
			Itv	Impt	ML	Itv	Impt	ML
brutefir	103	54	4	0	0	0	0	0
consol	298	165	20	10	10	0	0	0
id3	512	527	15	6	6	0	0	1
spell	2,213	450	20	8	17	0	1	1
mp3rename	2,466	332	33	3	3	0	1	1
irmp3	3,797	523	2	0	0	1	2	3
barcode	4,460	1,738	235	215	215	2	9	6
httptunnel	6,174	1,622	52	29	27	3	35	5
e2ps	6,222	1,437	119	58	58	3	6	3
bc	13,093	1,891	371	364	364	14	252	16
less	23,822	3,682	625	620	625	83	2,354	87
bison	56,361	14,610	1,988	1,955	1,955	137	4,827	237
pies	66,196	9,472	795	785	785	49	14,942	95
icecast-	68,564	6,183	239	232	232	51	109	107
raptor	76,378	8,889	2,156	2,148	2,148	242	17,844	345
dico	84,333	4,349	402	396	396	38	156	51
lsh	110,898	18,880	330	325	325	33	139	251
Total			7,406	7,154	7,166	656	40,677	1,207

-252

-240

실험 결과

- Effectiveness (leave-one-out cross validation)

Program	LOC	#Abs.Loc.	# Alarms			Time(s)		
			Itv	Impt	ML	Itv	Impt	ML
brutefir	103	54	4	0	0	0	0	0
consol	298	165	20	10	10	0	0	0
id3	512	527	15	6	6	0	0	1
spell	2,213	450	20	8	17	0	1	1
mp3rename	2,466	332	33	3	3	0	1	1
irmp3	3,797	523	2	0	0	1	2	3
barcode	4,460	1,738	235	215	215	2	9	6
httptunnel	6,174	1,622	52	29	27	3	35	5
e2ps	6,222	1,437	119	58	58	3	6	3
bc	13,093	1,891	371	364	364	14	252	16
less	23,822	3,682	625	620	625	83	2,354	87
bison	56,361	14,610	1,988	1,955	1,955	137	4,827	237
pies	66,196	9,472	795	785	785	49	14,942	95
icecast-	68,564	6,183	239	232	232	51	109	107
raptor	76,378	8,889	2,156	2,148	2,148	242	17,844	345
dico	84,333	4,349	402	396	396	38	156	51
lsh	110,898	18,880	330	325	325	33	139	251
Total			7,406	7,154	7,166	656	40,677	1,207

-252 -240

실험 결과

- Effectiveness (leave-one-out cross validation)

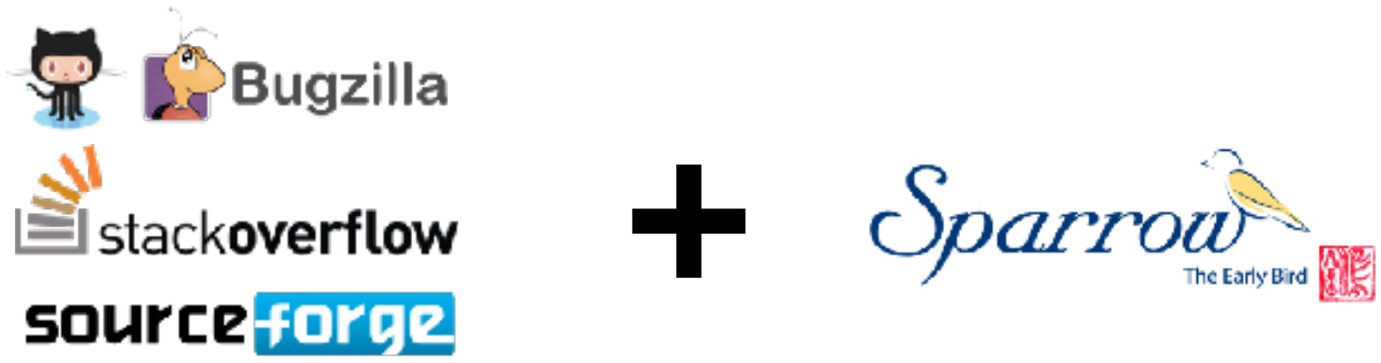
Program	LOC	#Abs.Loc.	# Alarms			Time(s)		
			Itv	Impt	ML	Itv	Impt	ML
brutefir	103	54	4	0	0	0	0	0
consol	298	165	20	10	10	0	0	0
id3	512	527	15	6	6	0	0	1
spell	2,213	450	20	8	17	0	1	1
mp3rename	2,466	332	33	3	3	0	1	1
irmp3	3,797	523	2	0	0	1	2	3
barcode	4,460	1,738	235	215	215	2	9	6
httptunnel	6,174	1,622	52	29	27	3	35	5
e2ps	6,222	1,437	119	58	58	3	6	3
bc	13,093	1,891	371	364	364	14	252	16
less	23,822	3,682	625	620	625	83	2,354	87
bison	56,361	14,610	1,988	1,955	1,955	137	4,827	237
pies	66,196	9,472	795	785	785	49	14,942	95
icecast-	68,564	6,183	239	232	232	51	109	107
raptor	76,378	8,889	2,156	2,148	2,148	242	17,844	345
dico	84,333	4,349	402	396	396	38	156	51
lsh	110,898	18,880	330	325	325	33	139	251
Total			7,406	7,154	7,166	656	40,677	1,207
							x62	x2

실험 결과

- Generalization : 작은 프로그램으로만 (<60KLOC) 학습

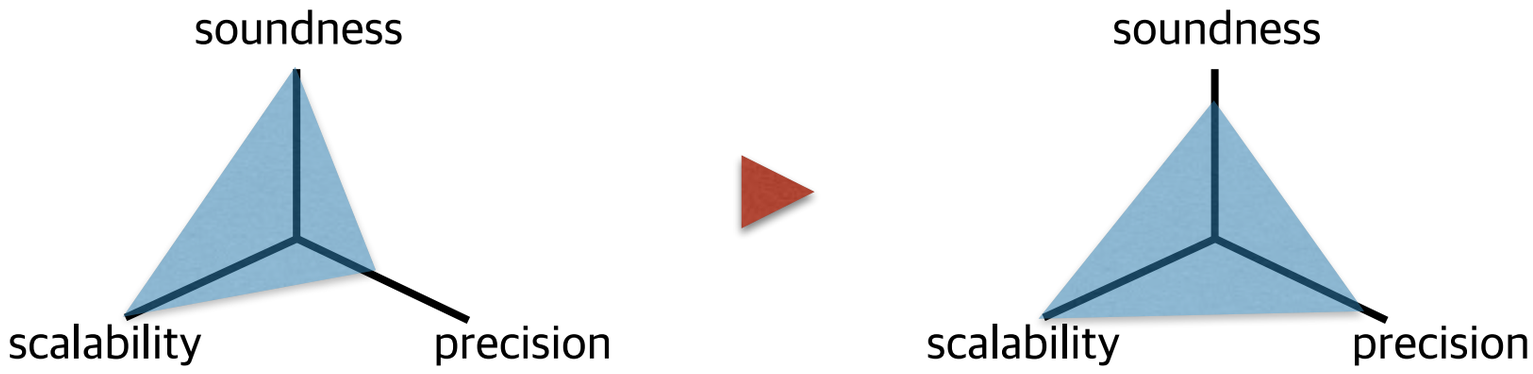
Program	LOC	Abs. Loc.	# Alarms			Time(s)		
			Itv	All	Small	Itv	All	Small
pies	66,196	9,472	795	785	785	49	95	98
icecast-server	68,564	6,183	239	232	232	51	113	99
raptor	76,378	8,889	2,156	2,148	2,148	242	345	388
dico	84,333	4,349	402	396	396	38	61	62
lsh	110,898	18,880	330	325	325	33	251	251
Total			7,406	3,886	3,886	413	865	898

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)

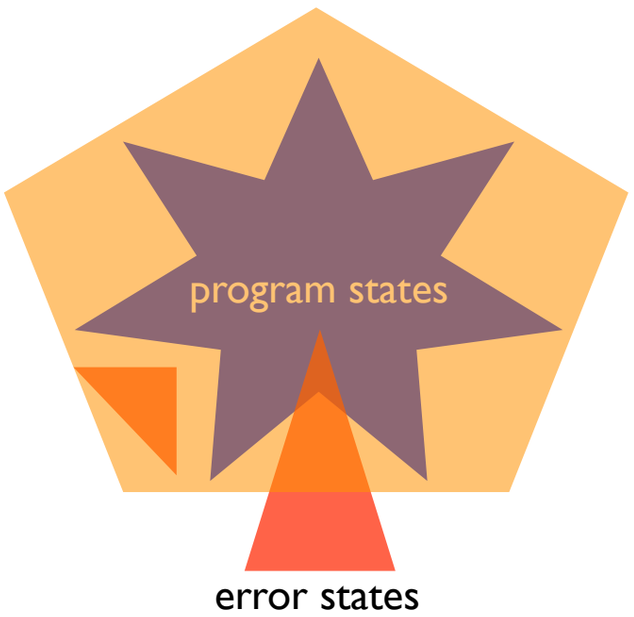


- Machine-Learning-Guided Selectively Unsound Static Analysis, ICSE'17

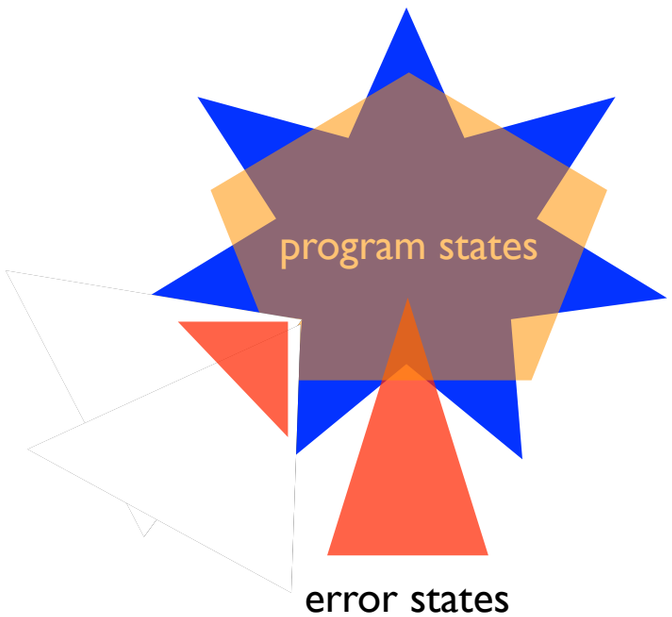
선별적으로 안전한 분석

- 안전성을 포기하는 전략을 조심스럽게 선별적으로
  - 예) 순환문 해체, 명령어 무시

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



sound



selectively unsound

unsound

예제

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

```
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

```
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 안전성을 포기하는 버퍼 오버런 분석기
  - 모든 순환문을 획일적으로 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 안전성을 포기하는 버퍼 오버런 분석기
  - 모든 순환문을 획일적으로 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 불필요한 안전성을 포기하는 버퍼 오버런 분석기

```
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]

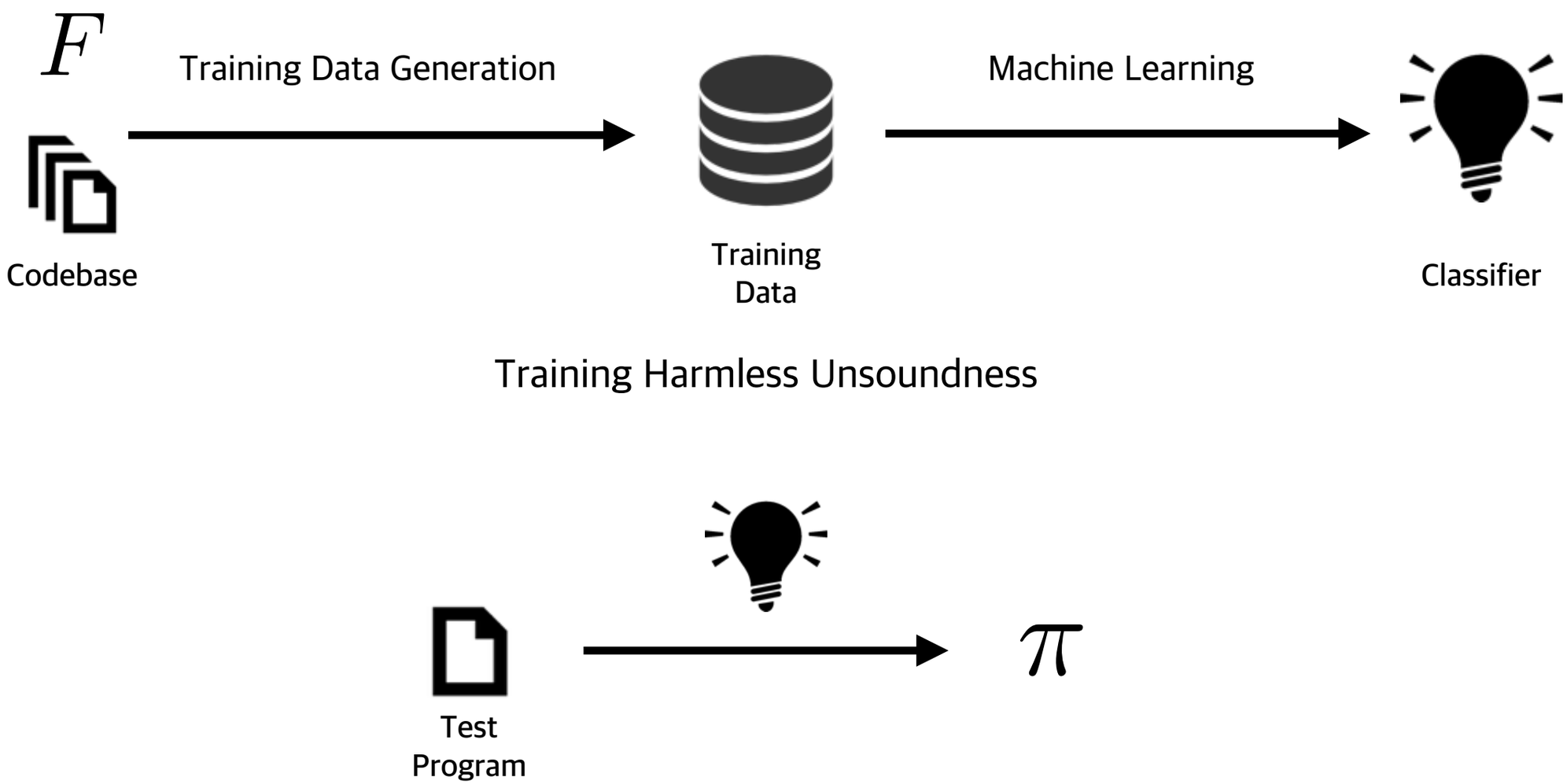
목표

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

- 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) `while(e){ C }` ▶ `if(e){ C }`

    `A;lib();B;` ▶ `A;B;`

큰 그림


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

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

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

안전한 분석 결과 (알람 개수)

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

특정 부품에 대한 안전성을 하나씩 포기하며 분석

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

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)

```
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

- 거짓 경보를 유발하는 부품: 전형적, 풍부
  - 프로그래밍 패턴, 분석기 도메인에 따른 특징
- 거짓 경보 >> 실제 경보
- 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 순환문
- 종료조건에서 Null 을 검사
- 유한한 문자열을 순환
- 전역변수를 안씀
- …

“무해한” 부품

- 거짓 경보를 유발하는 부품: 전형적, 풍부
  - 프로그래밍 패턴, 분석기 도메인에 따른 특징
- 거짓 경보 >> 실제 경보
- 예)

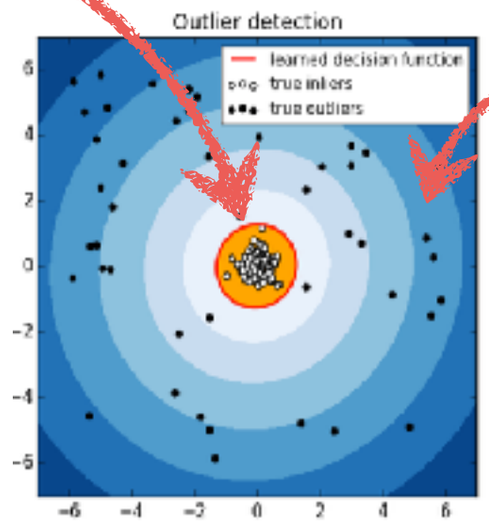
```
str = "hello world";
for(i=0; !str[i]; i++)// buffer access 1
  skip;

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

... = str[i];              // buff
```

유해한 순환문
- 종료 조건이 외부 입력에 의존
- 순환문 밖에 있는 배열의 인덱스
- …

기계 학습

- 데이터의 전형적인 특징을 찾는데 특화된 알고리즘 사용
  - 대표적으로 one-class SVM
- 많이 보던 (≈무해한) 경우에만 안전성 포기

무해한 부품



유해한 부품

특질 (Loop)

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 (Lib)

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

버그를 놓치는 경우

```
char arr[10];
int size = positive_input();
for (i = 0; i < size; i++)
  arr[i] = 1;
arr[i] = 0;

for(i = 0; arr[i]; i++)
```

버그를 놓치는 경우

```
char arr[10];
int size = positive_input();
for (i = 0; i < size; i++)
  arr[i] = 1;
arr[i] = 0;

for(i = 0; arr[i]; i++)
```

유해한 순환문
- 종료 조건이 외부 입력에 의존
- 순환문 밖에 있는 배열의 인덱스
- …

무해한 순환문
- 종료조건에서 Null 을 검사
- 유한한 문자열을 순환
- 전역변수를 안씀
- …

버그를 놓치는 경우

```
char arr[10];
int size = positive_input();
for (i = 0; i < size; i++)
  arr[i] = 1;
arr[i] = 0;

for(i = 0; arr[i]; i++)
```

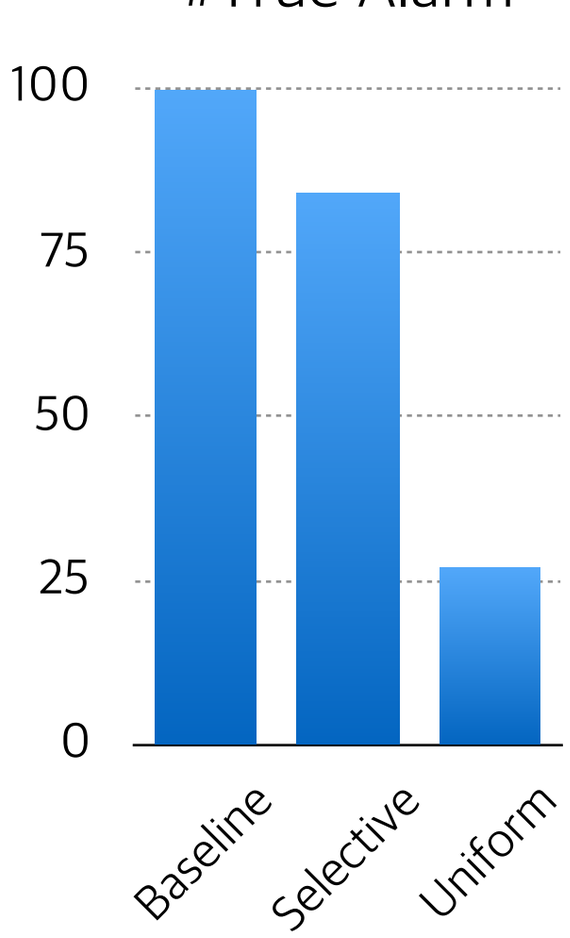
8

다행: 발견된 버그 하나를 고치면 나머지는 자연스레 해결

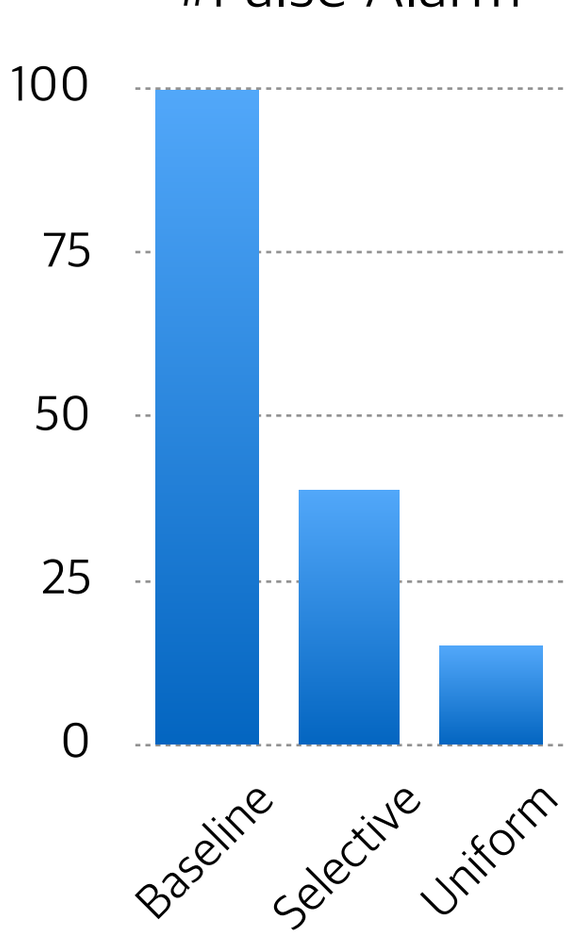
실험 결과

- 23 개 프로그램, 버퍼오버런 분석
  - 대상 : 순환문, 라이브러리 호출

#True Alarm



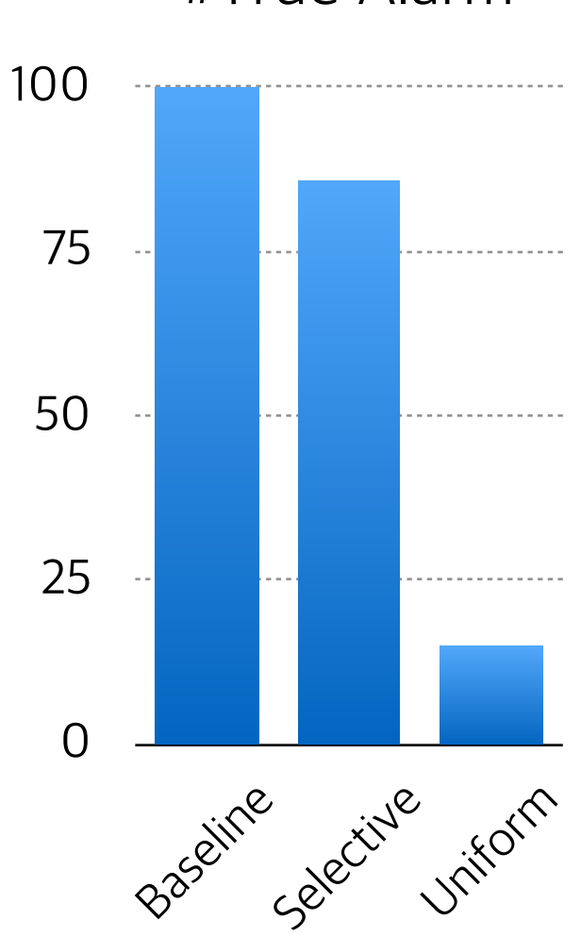
#False Alarm



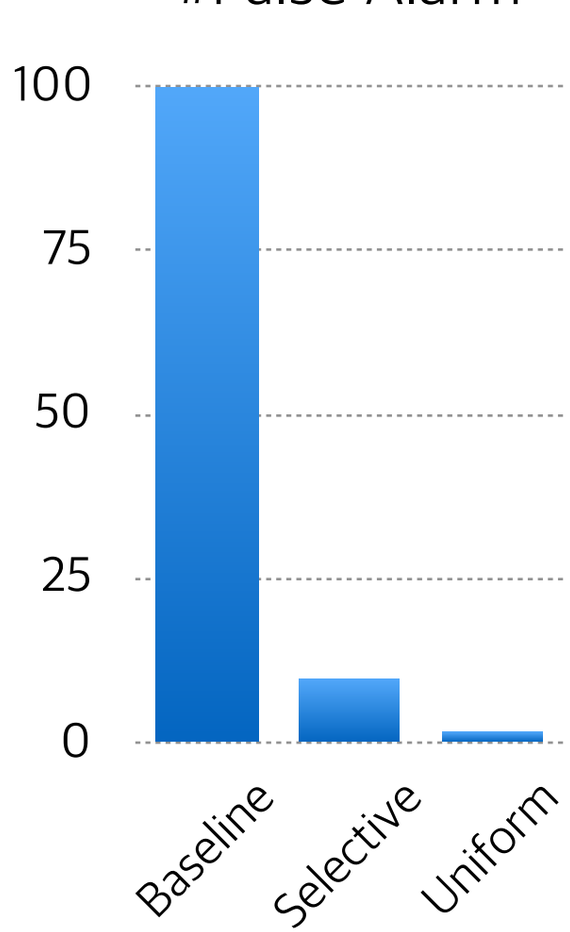
실험 결과

- 13 개 프로그램, 포맷 스트링 분석
  - 대상 : 라이브러리 호출

#True Alarm

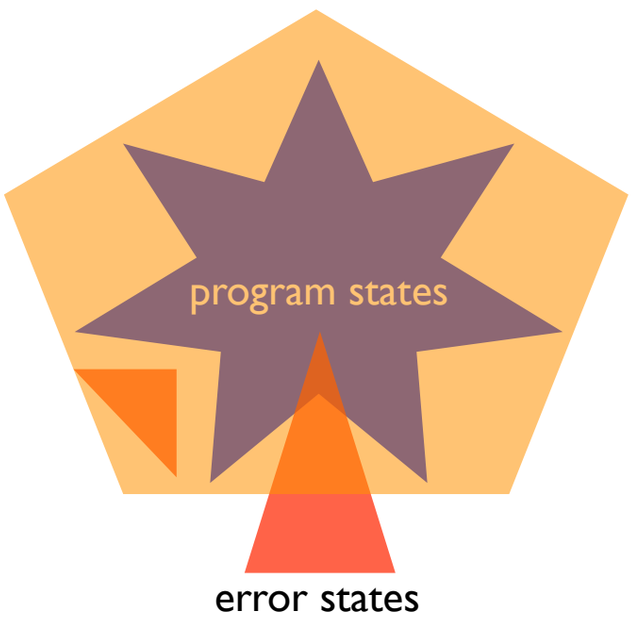


#False Alarm

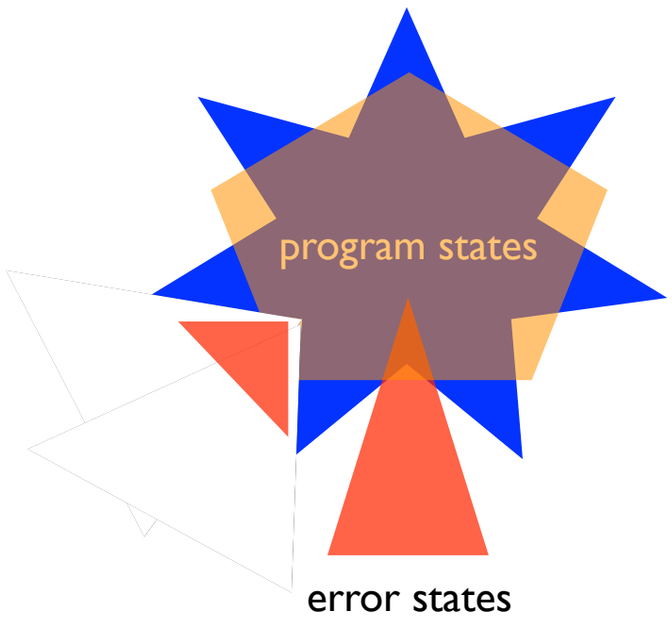


정리

- 불필요한 안전성을 조심스럽게 선별적으로 포기
  - 기존 버그 데이터 + 분석결과를 학습하여



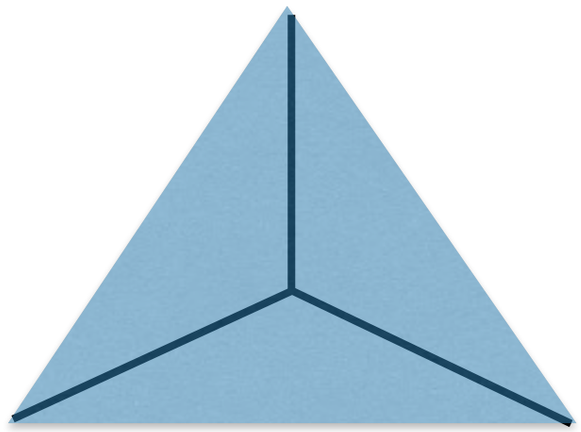
sound



selectively unsound

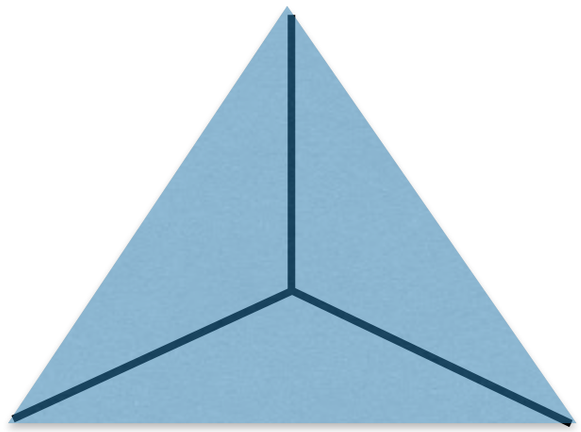
unsound

결론



- 안전, 정확, 빠른 정적 분석의 핵심: **선별적**
  - 예비 분석을 이용하여 **선별적**으로 정확하게 [PLDI'14, TOPLAS'16]
  - 기계 학습을 이용하여 **선별적**으로 정확하게 [SAS'16]
  - 기계 학습을 이용하여 **선별적**으로 안전하게 [ICSE'17]

결론



- 안전, 정확, 빠른 정적 분석의 핵심: **선별적**
  - 예비 분석을 이용하여 **선별적**으로 정확하게 [PLDI'14, TOPLAS'16]
  - 기계 학습을 이용하여 **선별적**으로 정확하게 [SAS'16]
  - 기계 학습을 이용하여 **선별적**으로 안전하게 [ICSE'17]

고맙습니다