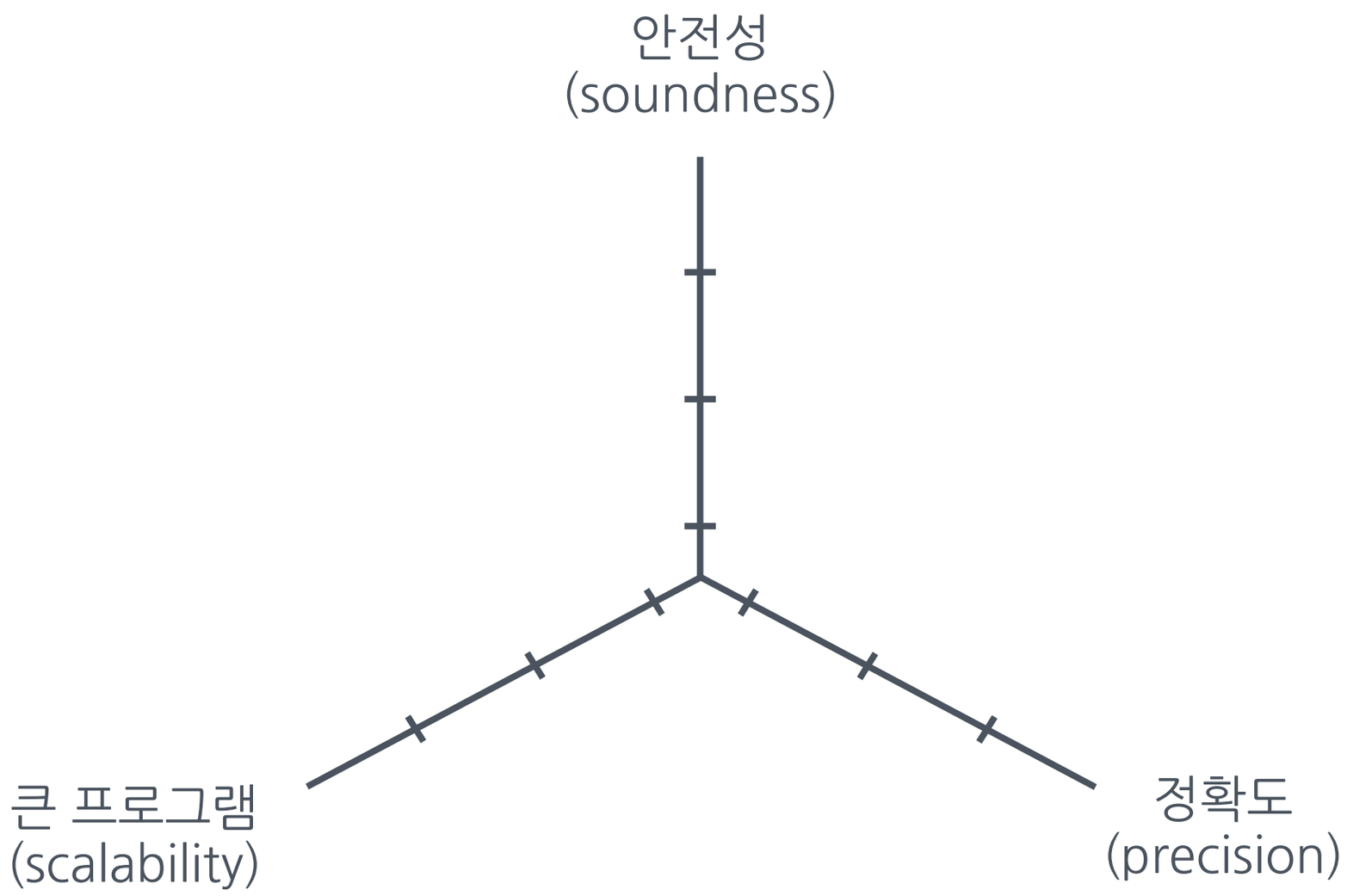
정적 분석의 정확도 선별적 향상 기법

오학주[1] 이원찬[2] 허기홍[1] 양홍석[2] 이광근[1]

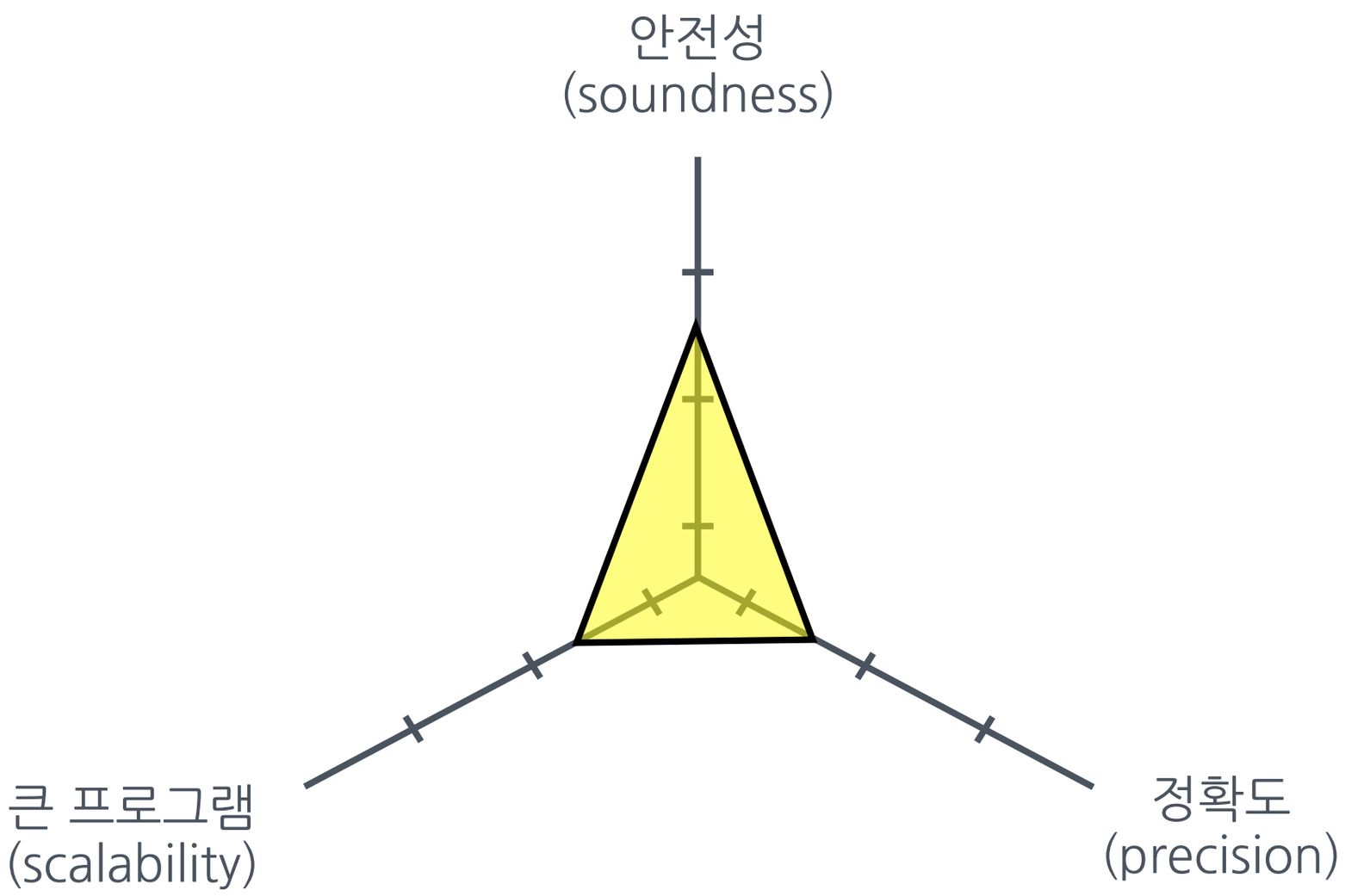
[1]서울대학교
[2]University of Oxford

Jan. 13, 2014 @ROSAEC workshop

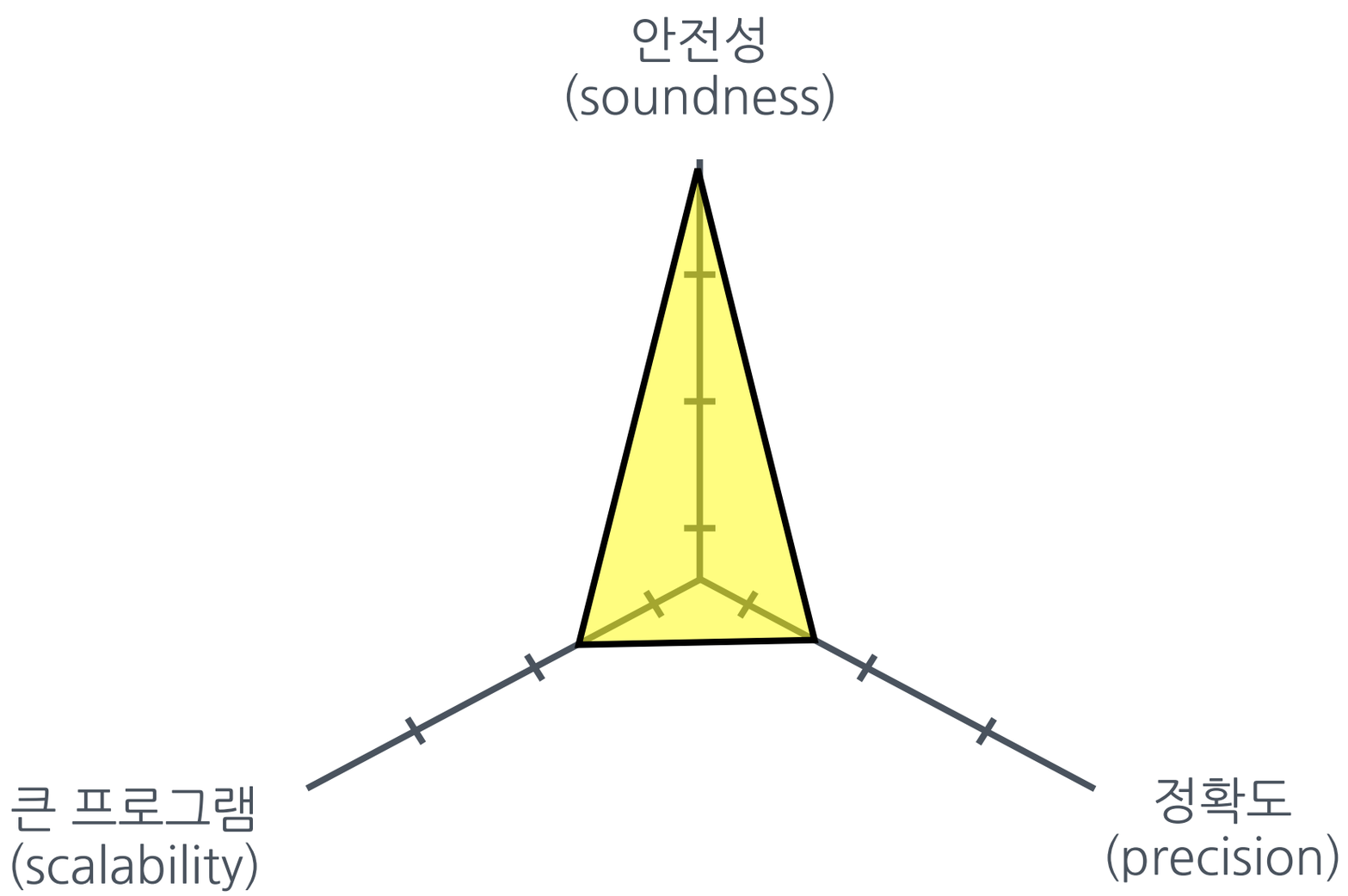
정적 분석 챌린지



현실

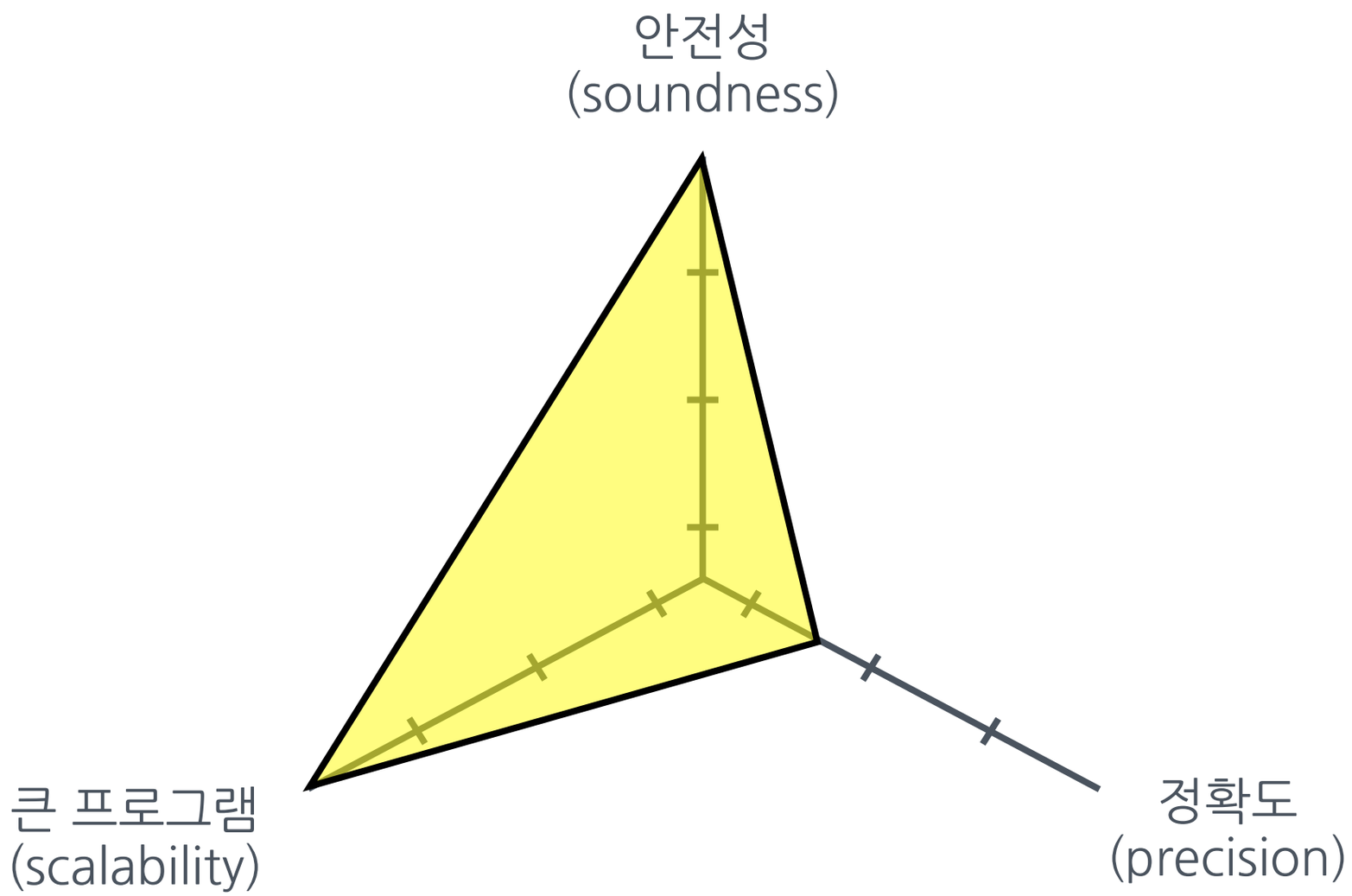


연구 동기



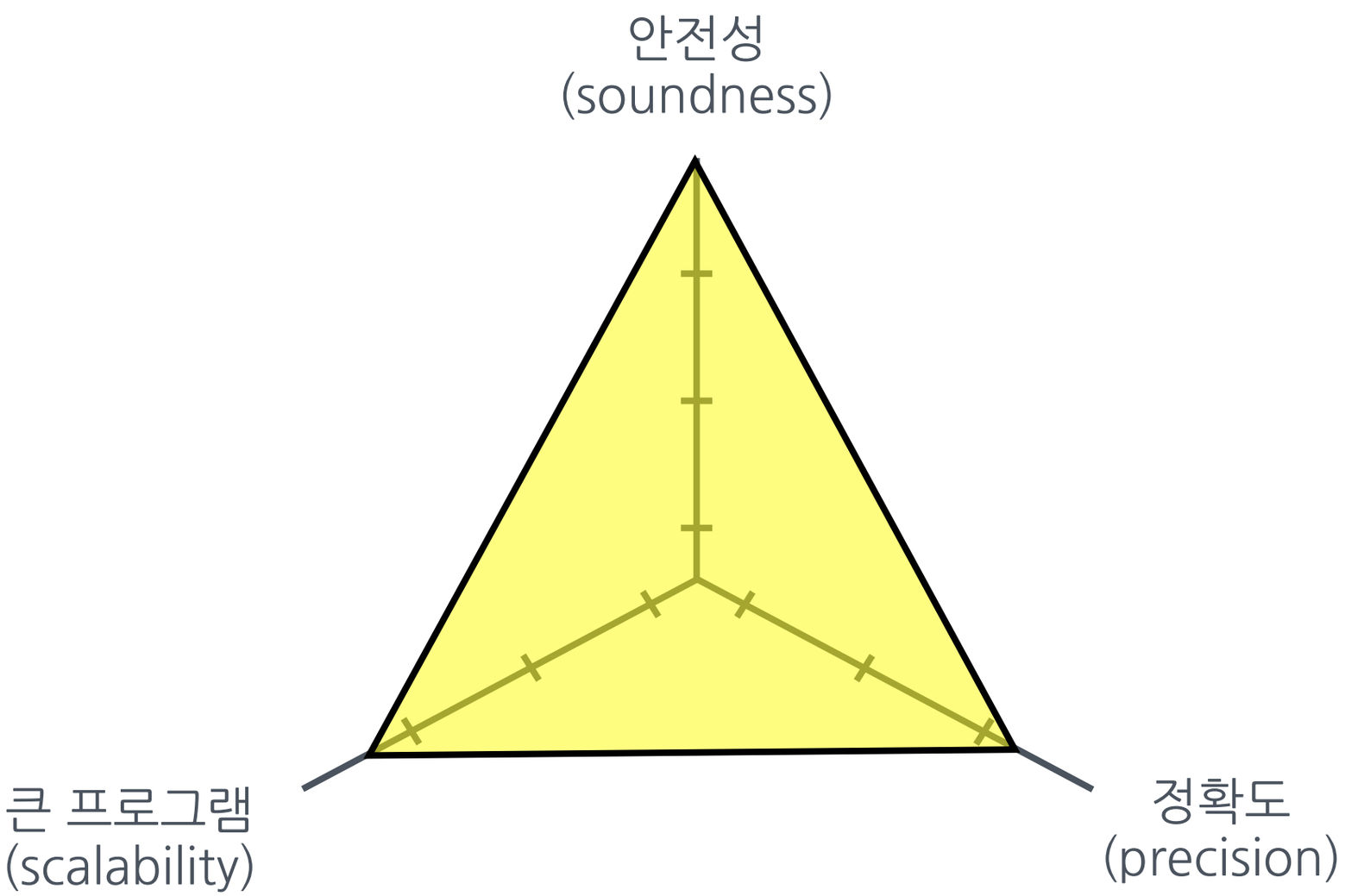
Sparrow (2007)
global analysis version

연구 동기



Sparse Sparrow (2012)

오늘 주제

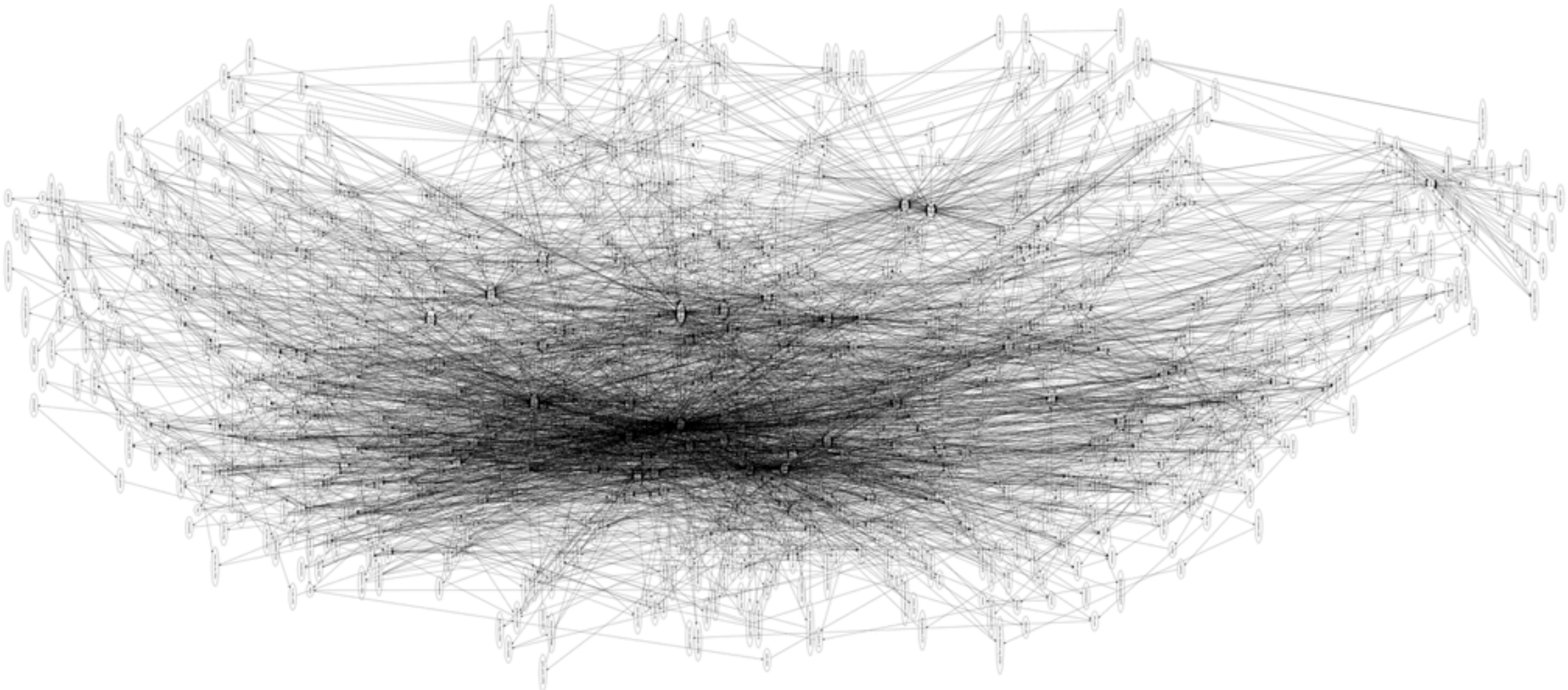


허위경보 원인들

- 함수 호출 문맥을 뭉침 (context-insensitive)
- 변수의 관계를 추적하지 않음 (non-relational)
- 배열의 원소를 구분하지 않음 (array smashing)
- 루프의 반복을 뭉침 (merging loop iterations)
- …

정확도를 함부로 높일수 없음

callgraph of nethack (240KLoC)



목표 / 전략

정확도를 **선별적으로** 향상

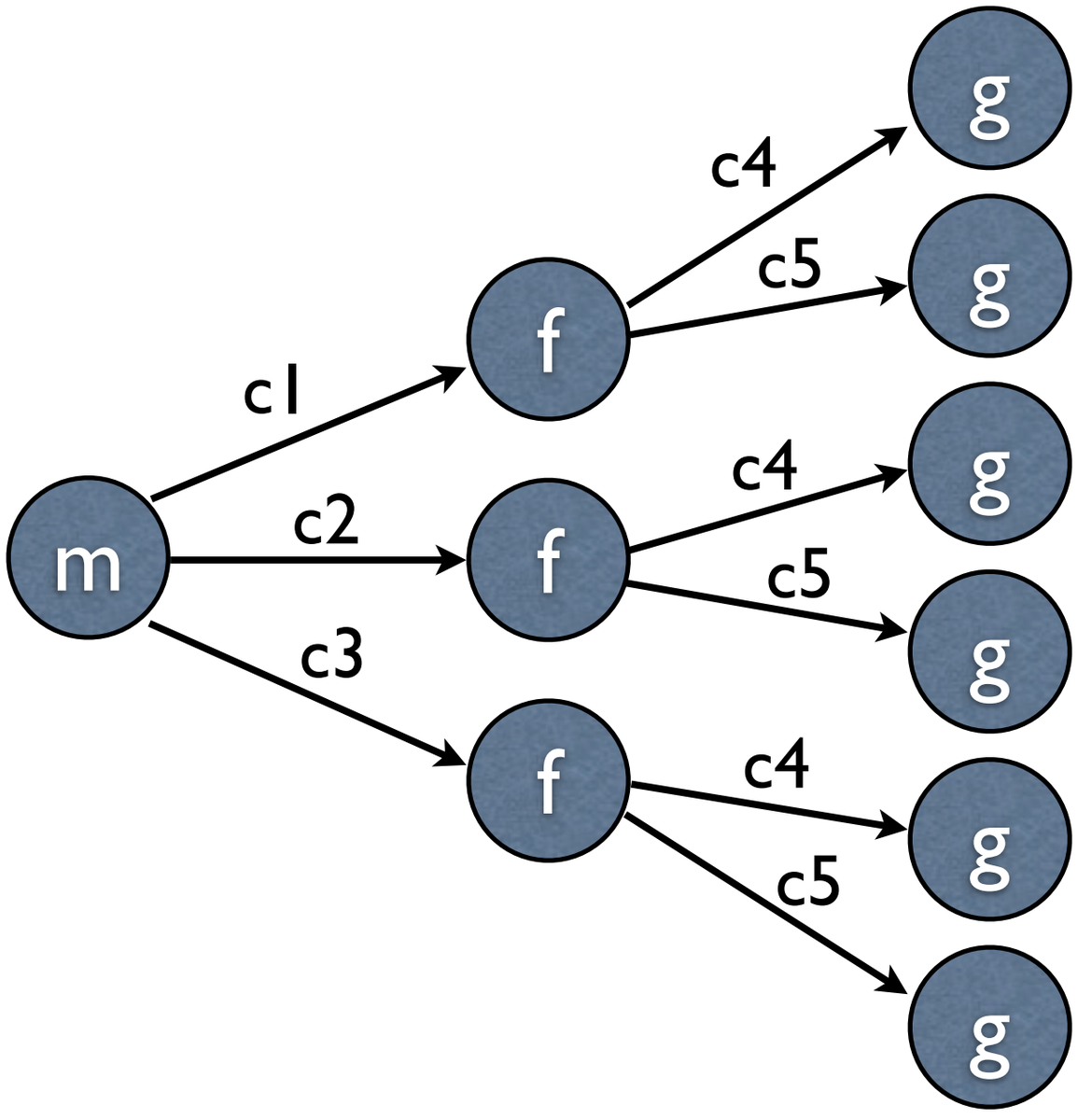
(최종 분석결과를 향상시킬때에만 정확도 향상)

예: 문맥 구분 분석

```
        void main() {
c1:       f(4);
c2:       f(8);
c3:       f(2);
        }

        void f(int a) {
c4:       x = g(a);
          assert (x > 1);
c5:       y = g(input());
          assert (y > 1);
        }

        void g(int a) {
          return a;
        }
```

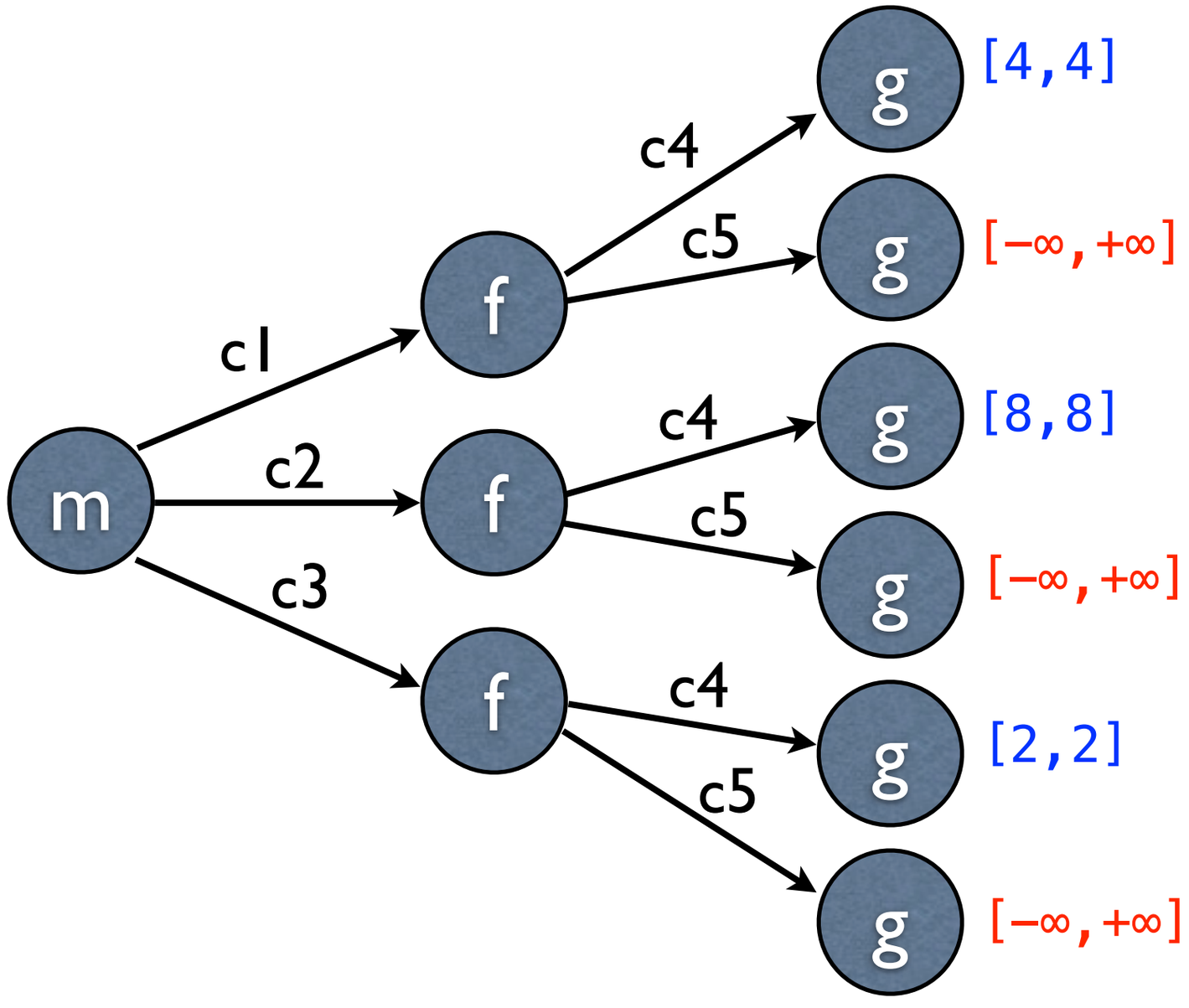
예: 문맥 구분 분석

```
      void main() {
c1:     f(4);
c2:     f(8);
c3:     f(2);
      }

      void f(int a) {
c4:     x = g(a);
        assert (x > 1);
c5:     y = g(input());
        assert (y > 1);
      }

      void g(int a) {
        return a;
      }
```



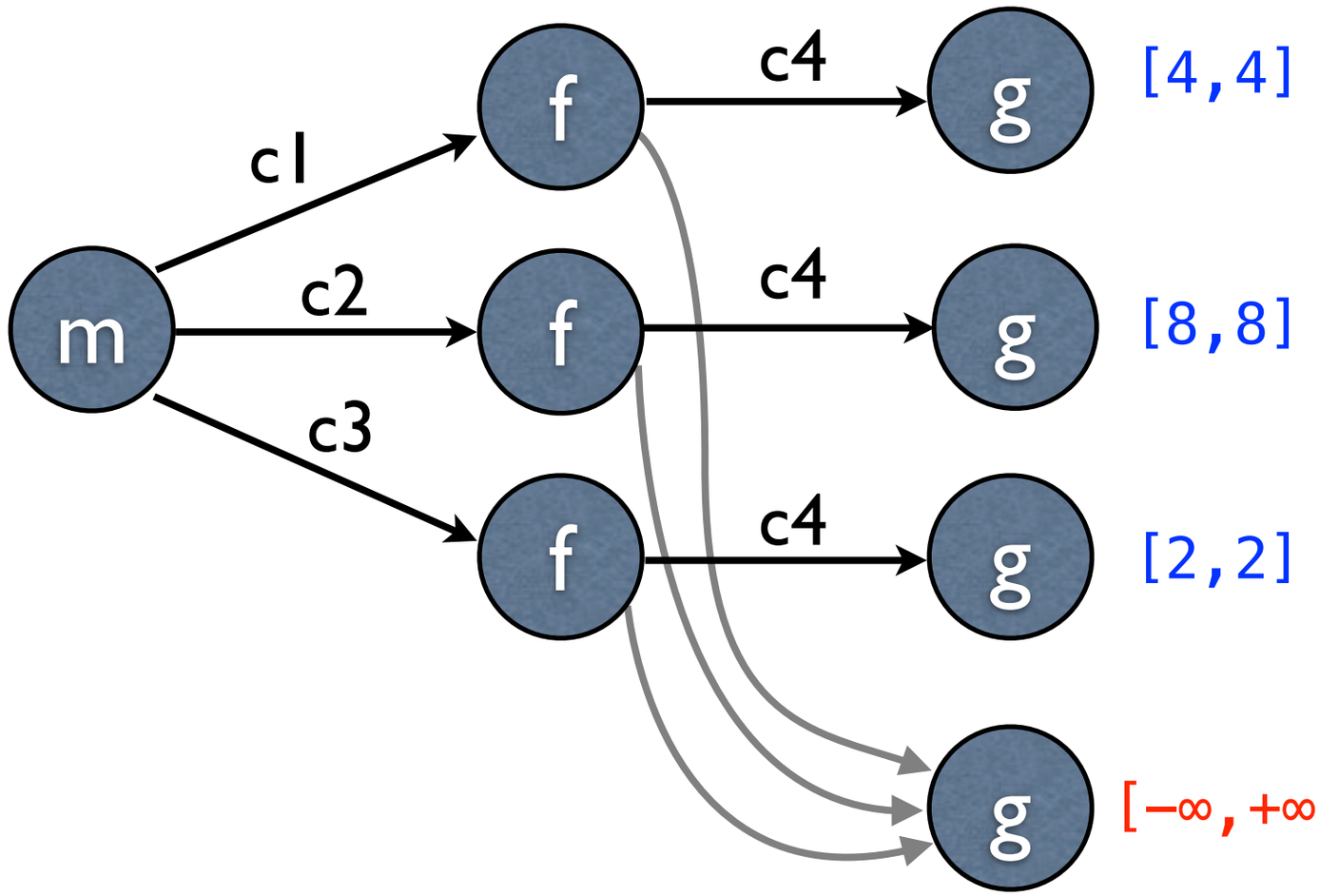
문맥 구분이 항상 유용하진 않음

```
      void main() {
c1:     f(4);
c2:     f(8);
c3:     f(2);
      }

      void f(int a) {
c4:     x = g(a);
        assert (x > 1);
c5:     y = g(input());
        assert (y > 1);
      }

      void g(int a) {
        return a;
      }
```



선별적 문맥 구분 분석

```
      void main() {
c1:     f(4);
c2:     f(8);
c3:     f(2);
      }

      void f(int a) {
c4:     x = g(a);
        assert (x > 1);
c5:     y = g(input());
        assert (y > 1);
      }

      void g(int a) {
        return a;
      }
```



문맥 선별은 전분석으로

- 본분석의 요약본 (over-approximation)
- 모든 문맥을 구분 (fully context-sensitive)

예: 인터벌 분석

(1) 본분석을 요약

본분석

$$\mathbb{I} = \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, +\infty\} \wedge l \leq u\}.$$

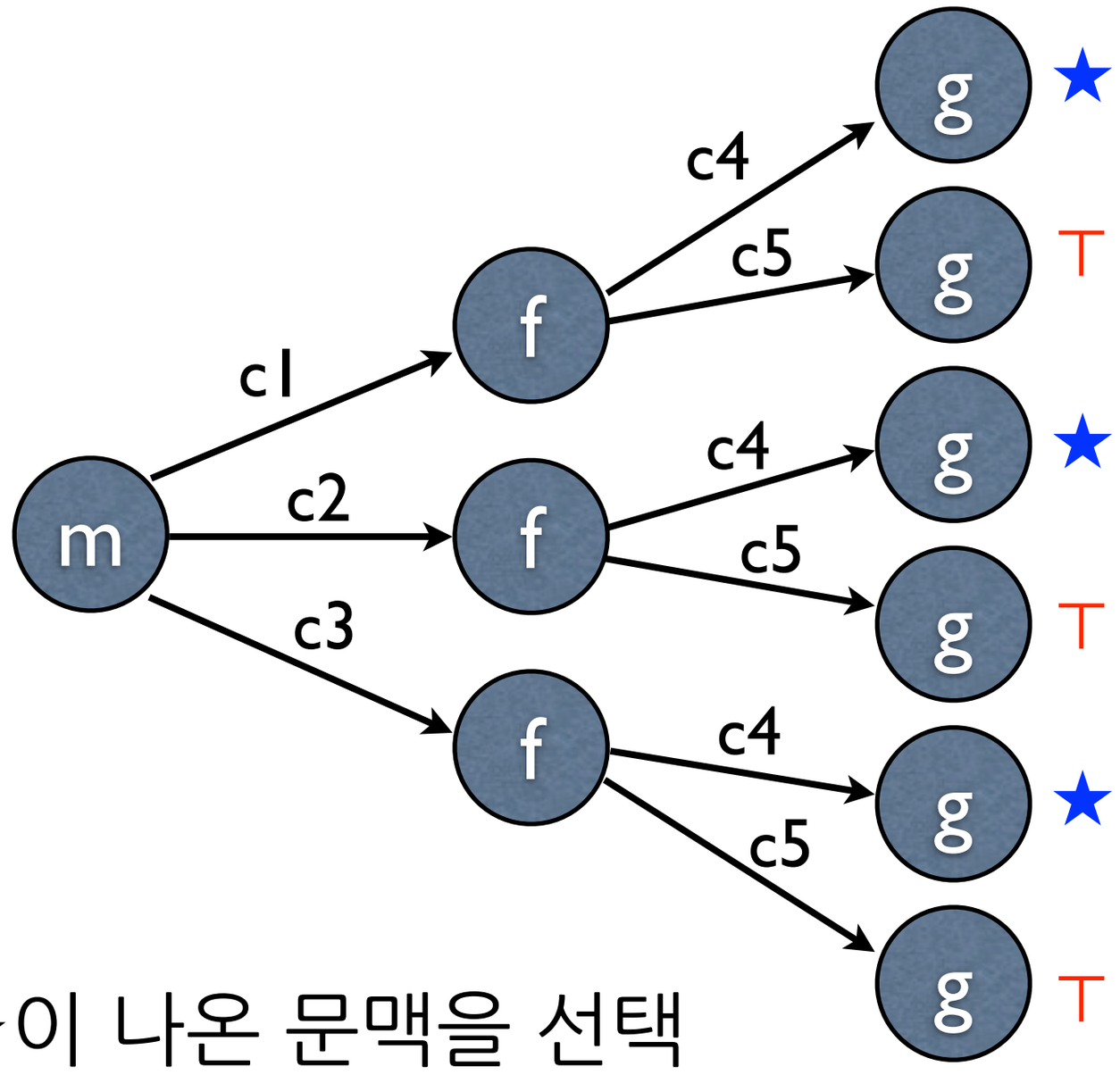
전분석

$$\mathbb{V} = \{\bot_v, \bigstar, \top_v\} \qquad \gamma_v(\bigstar) = \{[a, b] \in \mathbb{I} \mid 0 \leq a\},$$

예: 인터벌 분석

(2) 문맥은 모두 구분

```
      void main() {
c1:     f(4);
c2:     f(8);
c3:     f(2);
      }

      void f(int a) {
c4:     x = g(a);
        assert (x > 1);
c5:     y = g(input());
        assert (y > 1);
      }

      void g(int a) {
        return a;
      }
```

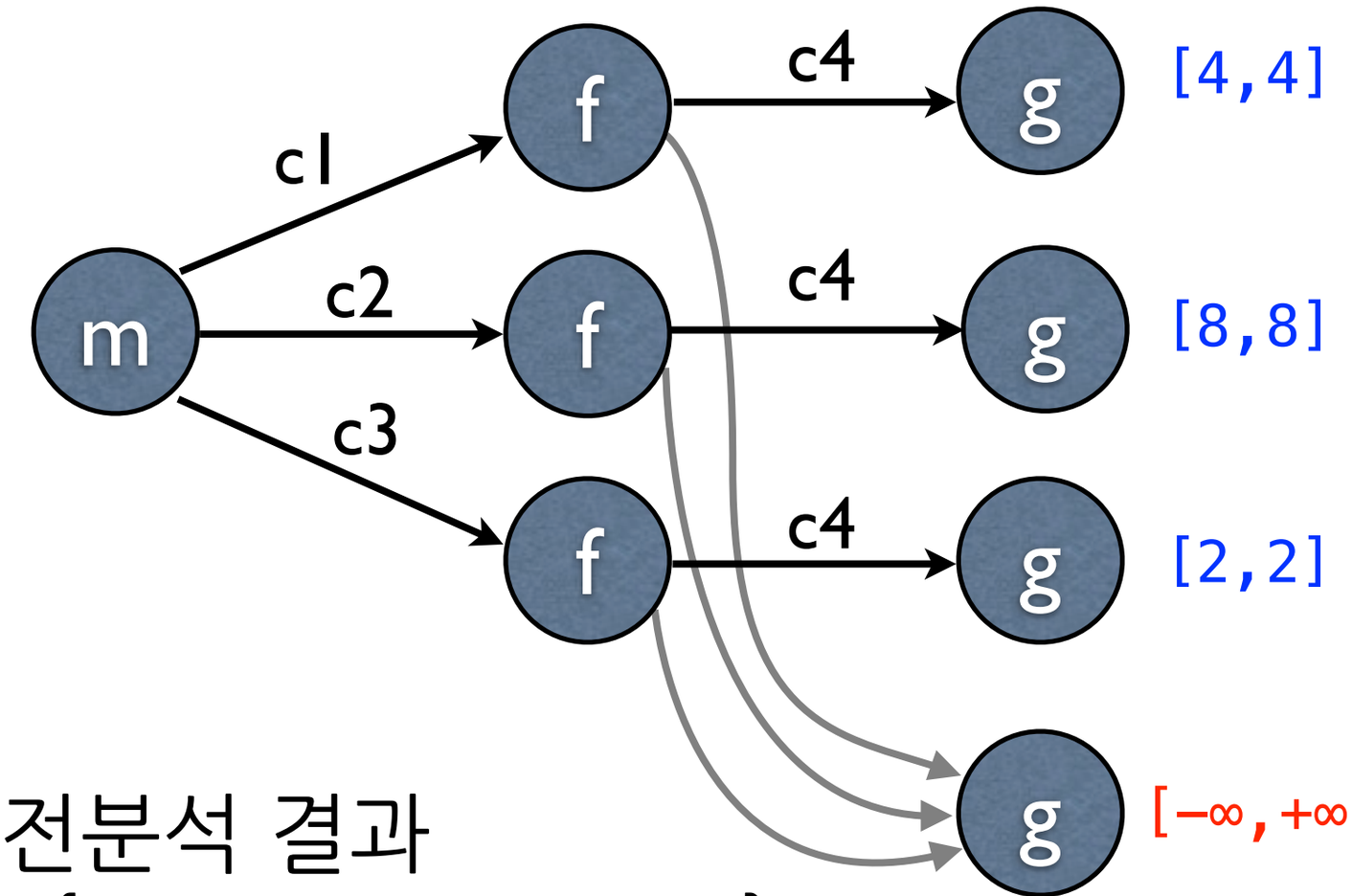


★이 나온 문맥을 선택
: {c1c4, c2c4, c3c4}

예: 인터벌 분석

(3) 본분석: 선별 문맥 구분

```
      void main() {
c1:     f(4);
c2:     f(8);
c3:     f(2);
      }

      void f(int a) {
c4:     x = g(a);
        assert (x > 1);
c5:     y = g(input());
        assert (y > 1);
      }

      void g(int a) {
        return a;
      }
```



전분석 결과
: {c1c4, c2c4, c3c4}

기술적 이슈 (1/2)

- 본분석이 전분석보다 항상 정확하진 않음

$$\mathsf{A}_{K_\infty}$$

$$\sqcap\!\!|$$

$$\hat{\mathsf{A}}_{K_\infty} \xrightarrow{\hat{K}_\star} \mathsf{A}_{\hat{K}_\star}$$

전분석 본분석

기술적 이슈 (2/2)

- 모든 문맥을 구분하는 전분석 비용
  - 효율을 위한 조건과 분석 알고리즘

실험결과

평균 27.8% 추가 비용, 24.4% 버퍼오버런 경보 제거

(전분석: 14.7%)

program	ctx-insensitive		selective ctx-sensitve	
	시간(s)	알람수	시간(s)	알람수
make	84.4	1,500	106.2	1,028
grep	12.1	735	15.9	653
wget	69.0	1,307	82.1	942
a2ps	118.1	3,681	177.7	2,121
bison	136.3	1,894	173.4	1,742

다른 응용

- 전분석을 이용한 선별적 “X-구분” 분석
- 본분석을 요약
- 하지만 X는 모두 구분

다른 응용

- X = 변수 관계 분석
- 전분석을 이용한 선별적 관계 분석
  - 본분석을 요약
  - 변수들의 관계는 모두 구분

예: 옥타곤 관계 분석

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

예: 관계 분석

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

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

선별적 관계 분석

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

	a	b	i
a	0	0	-1
b	0	0	-1
i	∞	∞	0

$$c = [-\infty, +\infty]$$

전분석

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

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

	a	b	c	i
a	★	★	⊤	★
b	★	★	⊤	★
c	⊤	⊤	★	⊤
i	⊤	⊤	⊤	★

실험결과

기존방식에 비해서 3배 더 증명, 5배 빠름

program	#query	구문기반 선별방식		우리방식(의미기반)	
		시간	증명	시간	증명
barcode	37	12	16	30	37
httptunel	28	26	17	15	26
bc	10	247	2	117	9
tar	11	1043	7	661	11
less	13	3031	0	2849	13

결론

- 전분석을 이용한 선별적 프로그램 분석
  - 선별적 문맥 구분 분석
  - 선별적 관계 분석
- 다른 응용도 가능할 것
  - 선별적으로 흐름을 구분하는 분석
  - 선별적으로 배열을 구분하는 분석
  - 선별적으로 룹을 구분하는 분석
  - …