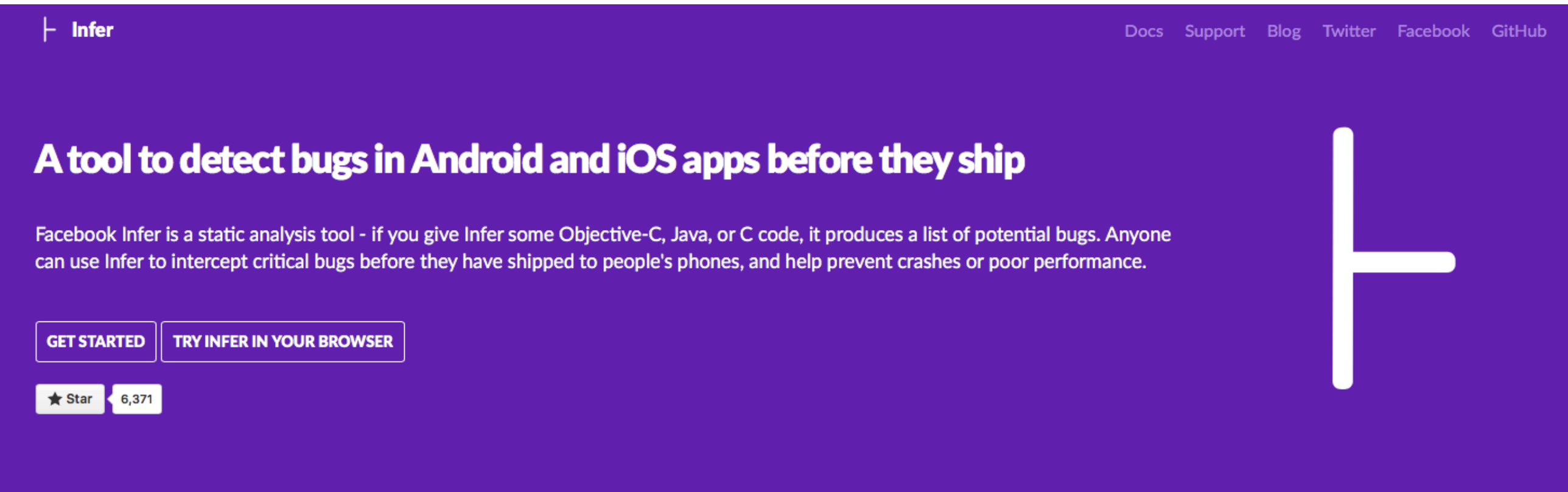
Inferbo: Infer 기반 버퍼 오버런 오류 분석기

허기홍 조성근 이광근
서울대학교

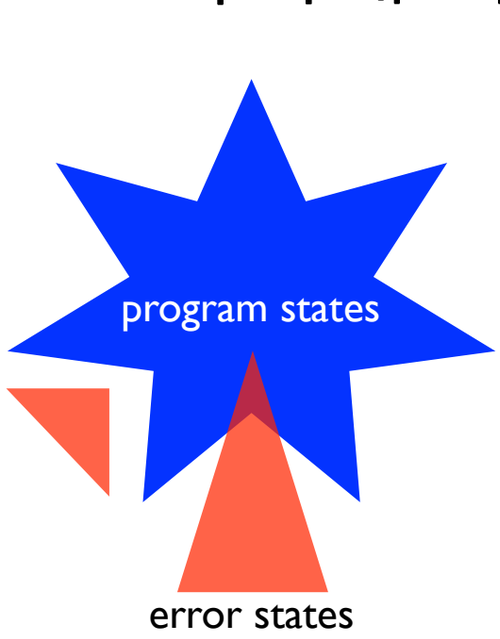
Infer

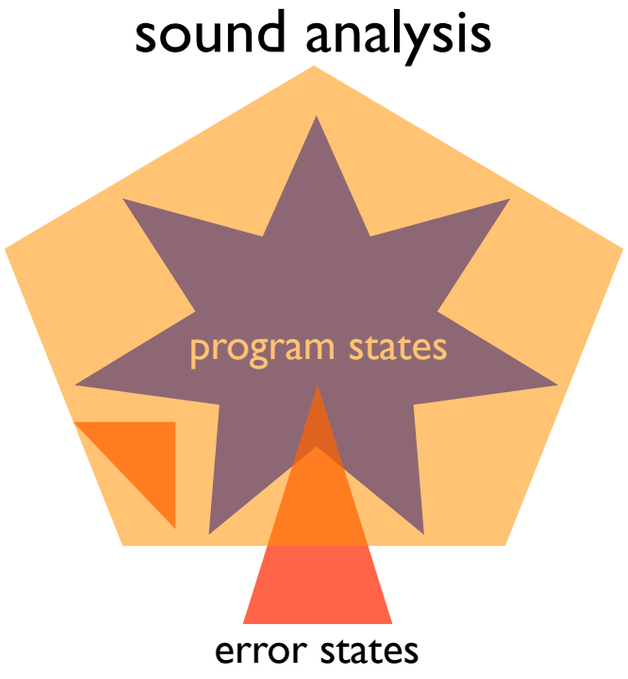
- Facebook 의 오픈소스 정적 분석기
  - 개별 분석 (modular analysis) 기반
  - 산업 현장에서 쓸수 있을 정도로 빠르고 정확
  - 대상: memory/resource leak, null dereference, buffer-overrun 등

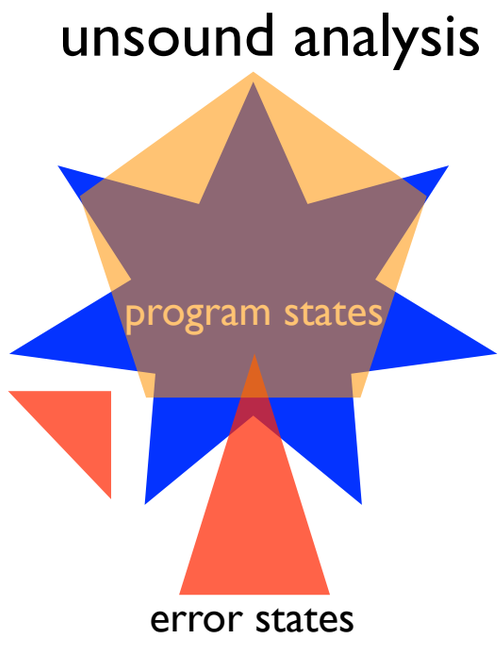


정적 분석

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

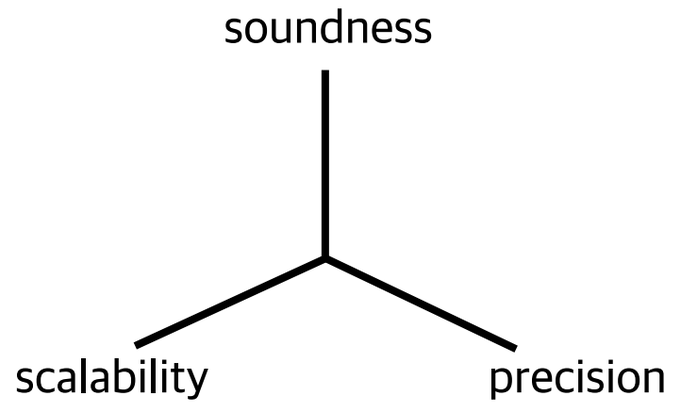




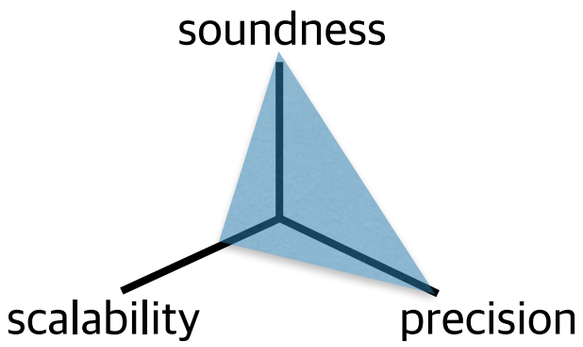


도전 과제

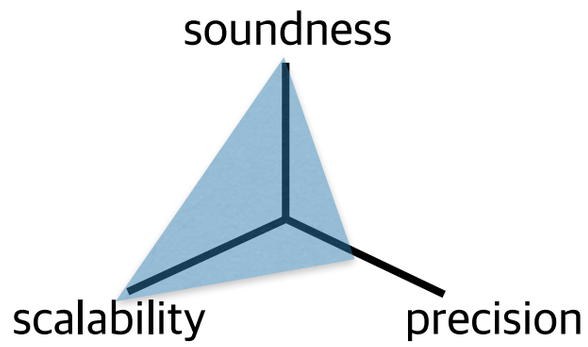
- 성능의 세가지 축: 모두 달성하는 것은 불가능



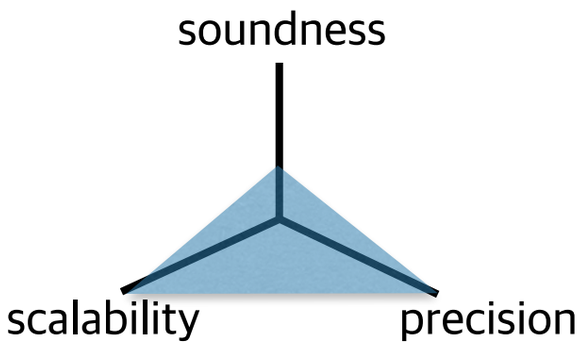
- 전통적인 분류,



무결성 검증용



코드 최적화용



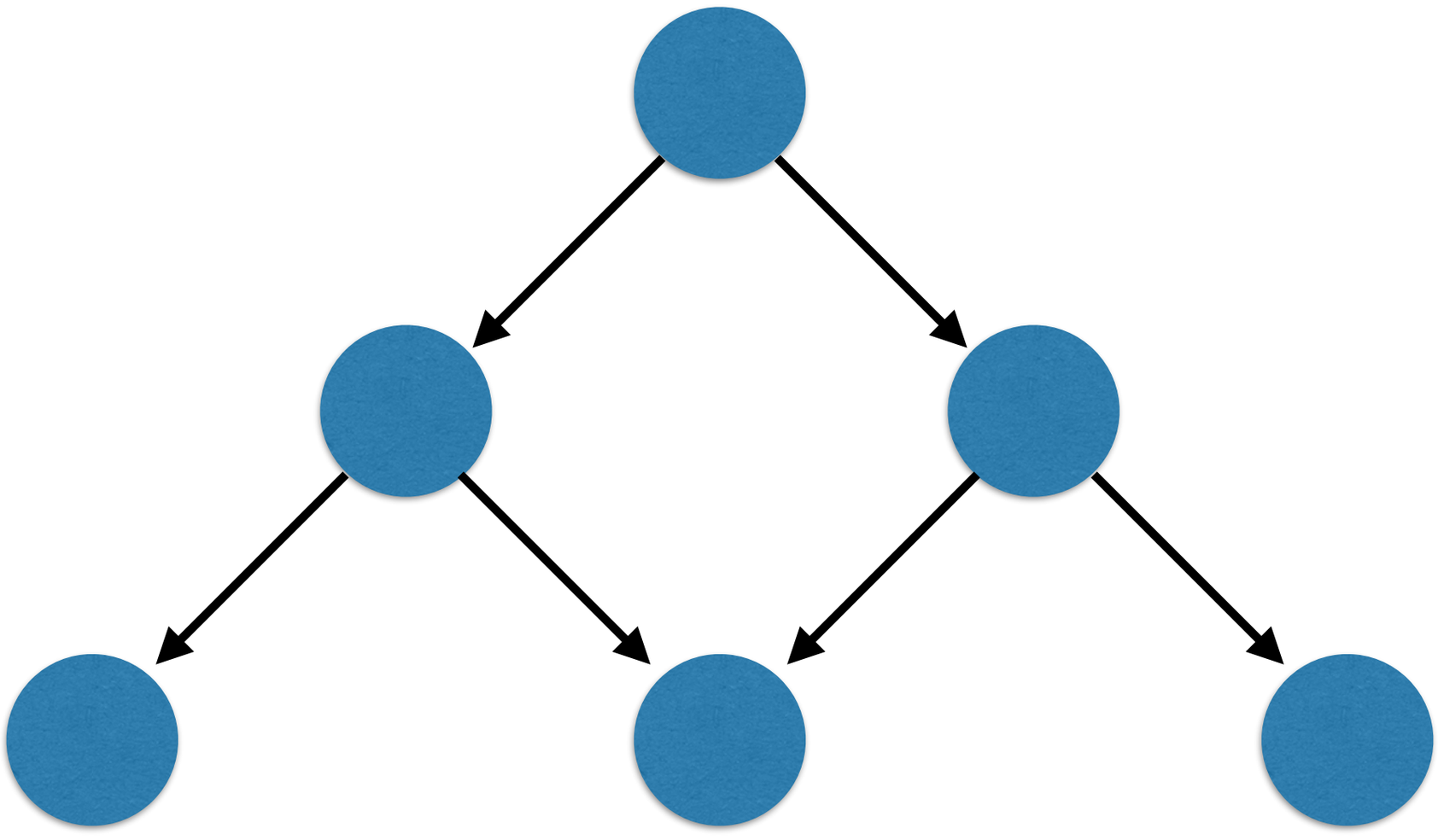
오류 검출용

개별 분석 vs 전체 분석?

(modular) (global)

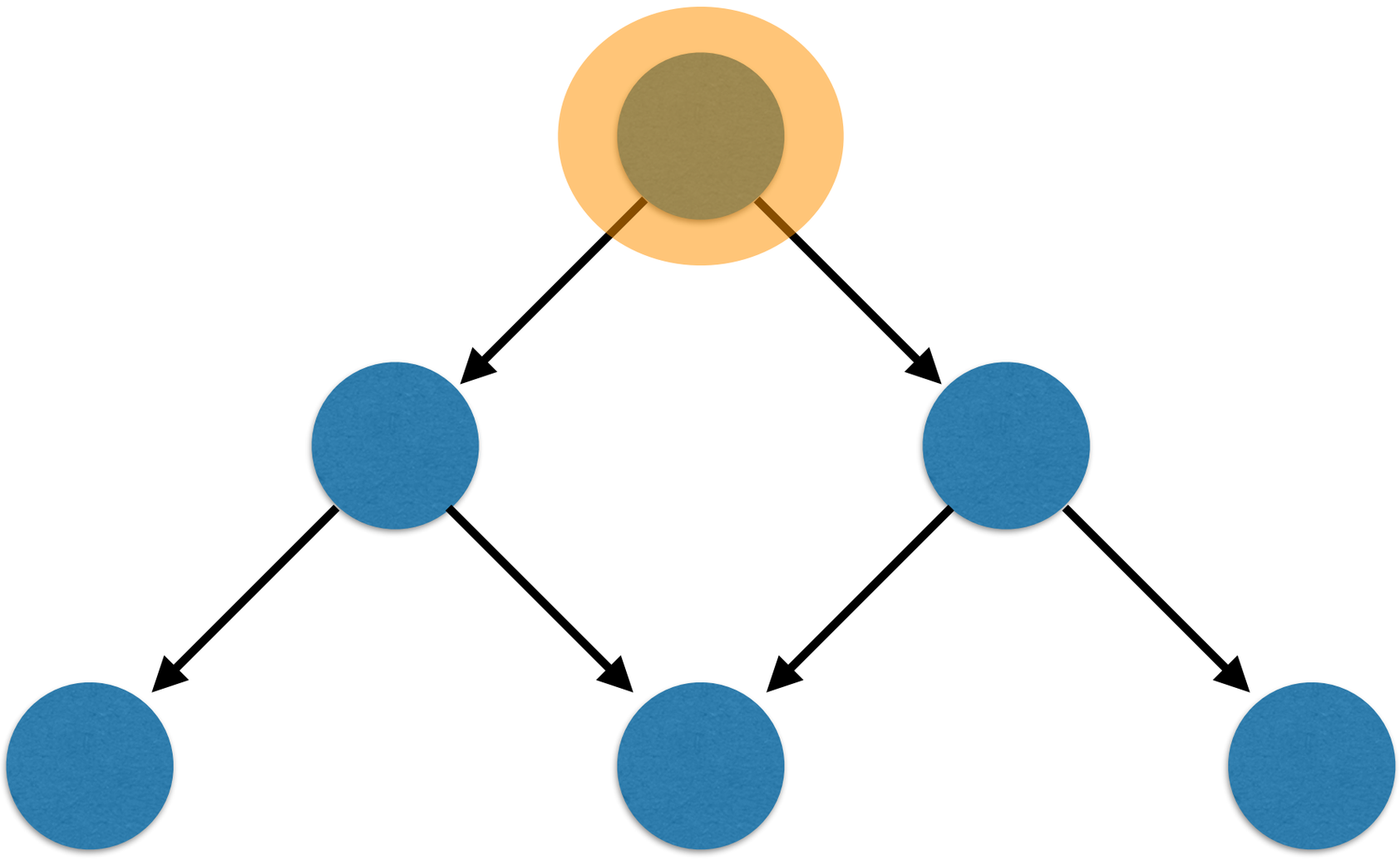
전체 분석 (global analysis)

- 프로그램을 구조를 따라 전체를 한꺼번에 분석 (예, Sparrow)



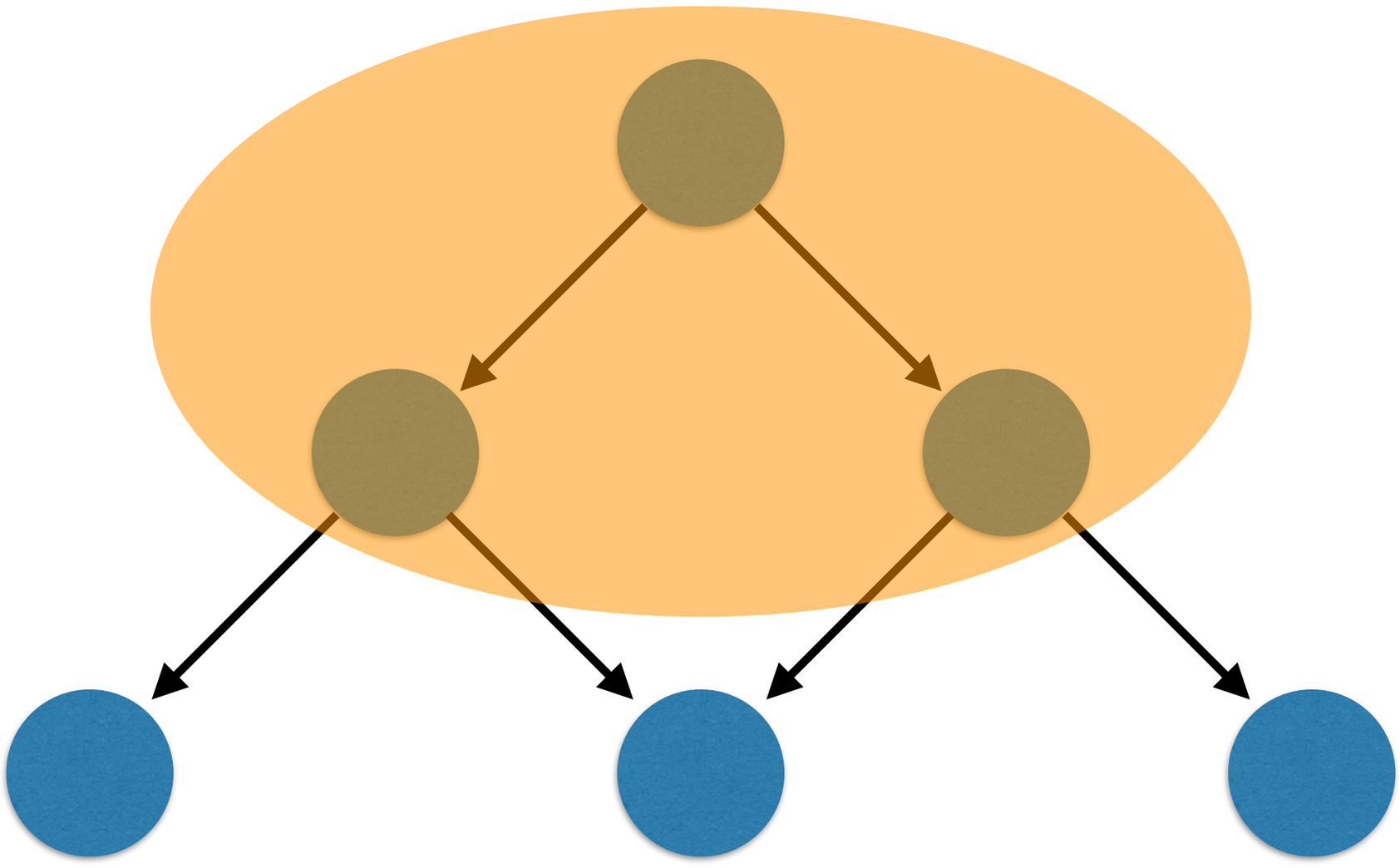
전체 분석 (global analysis)

- 프로그램을 구조를 따라 전체를 한꺼번에 분석 (예, Sparrow)



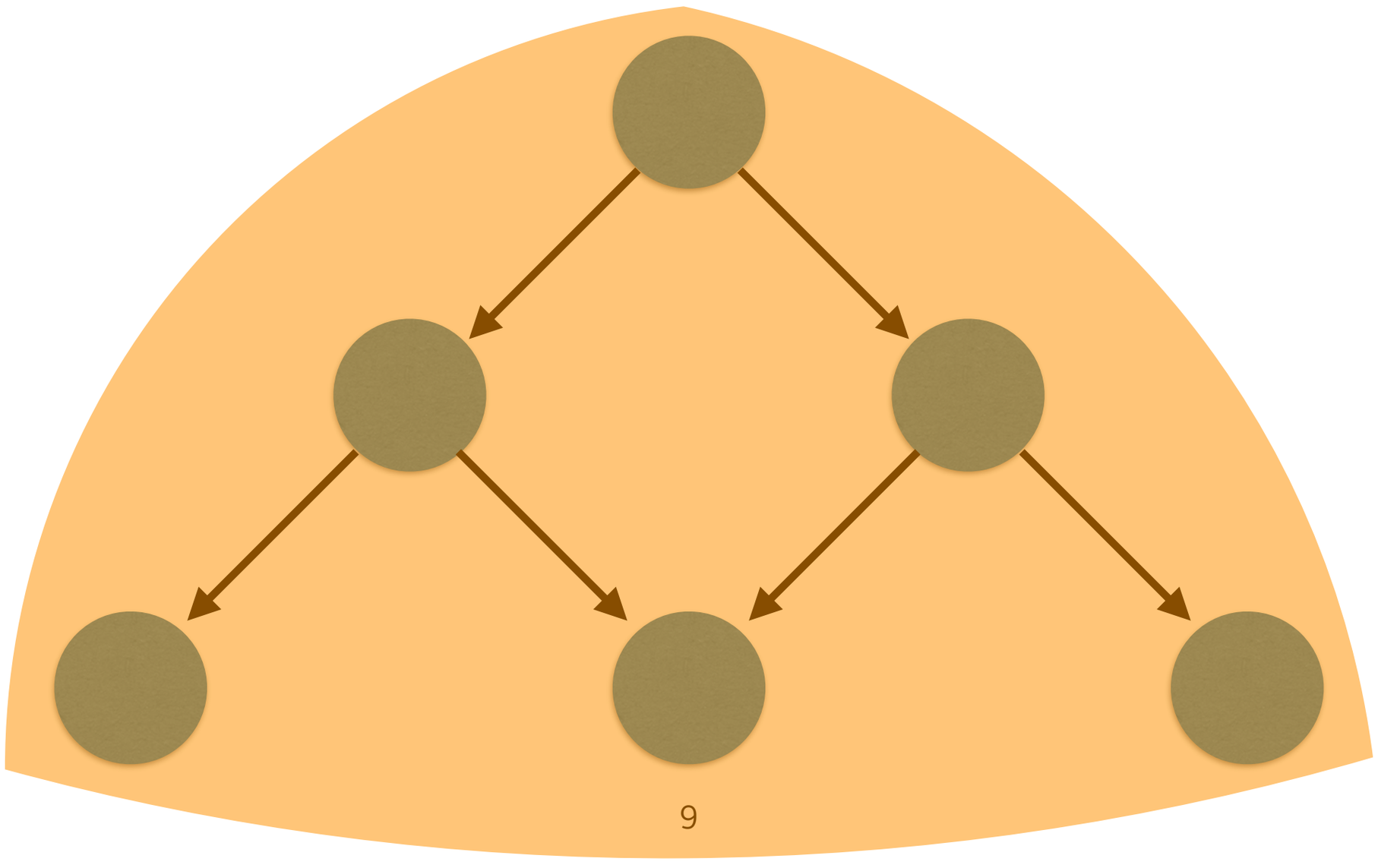
전체 분석 (global analysis)

- 프로그램을 구조를 따라 전체를 한꺼번에 분석 (예, Sparrow)



전체 분석 (global analysis)

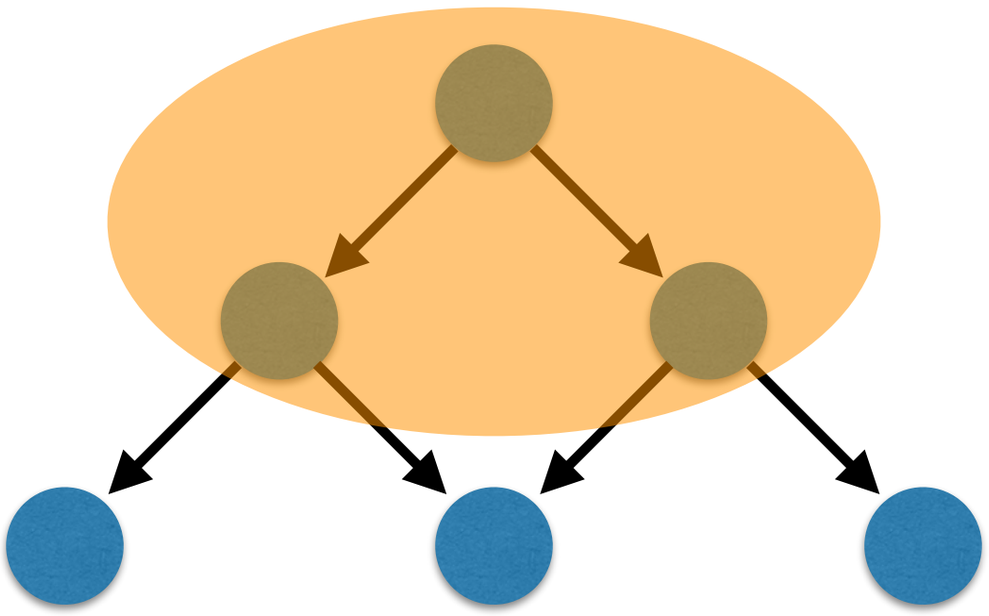
- 프로그램을 구조를 따라 전체를 한꺼번에 분석 (예, Sparrow)



전체 분석 (global analysis)

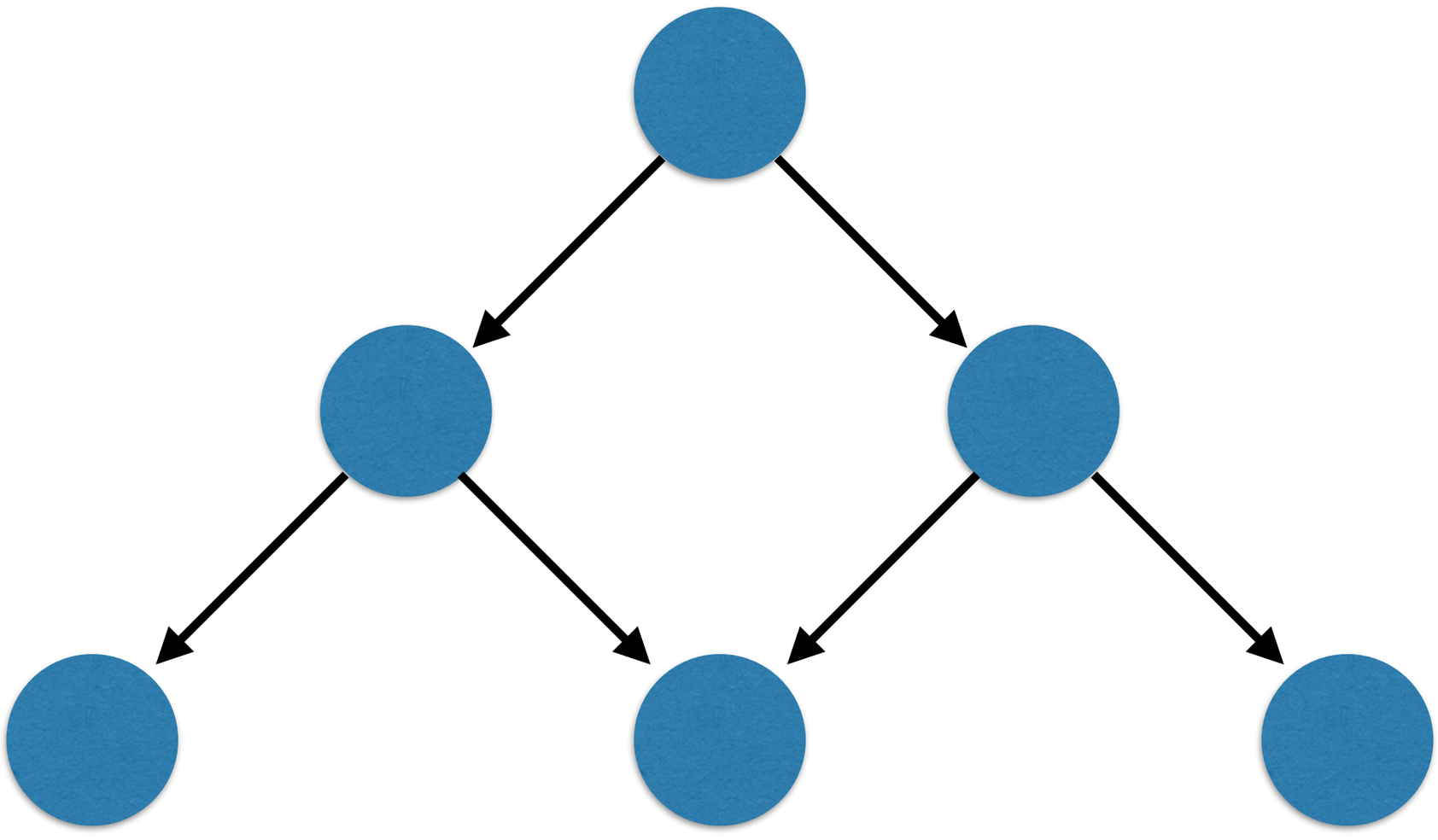
(일반적으로)

- 장점: 문맥을 아는 상태로 분석 (파라미터, 전역변수 등) => 정확
- 단점: 같은 부분을 여러번 분석 => 느림



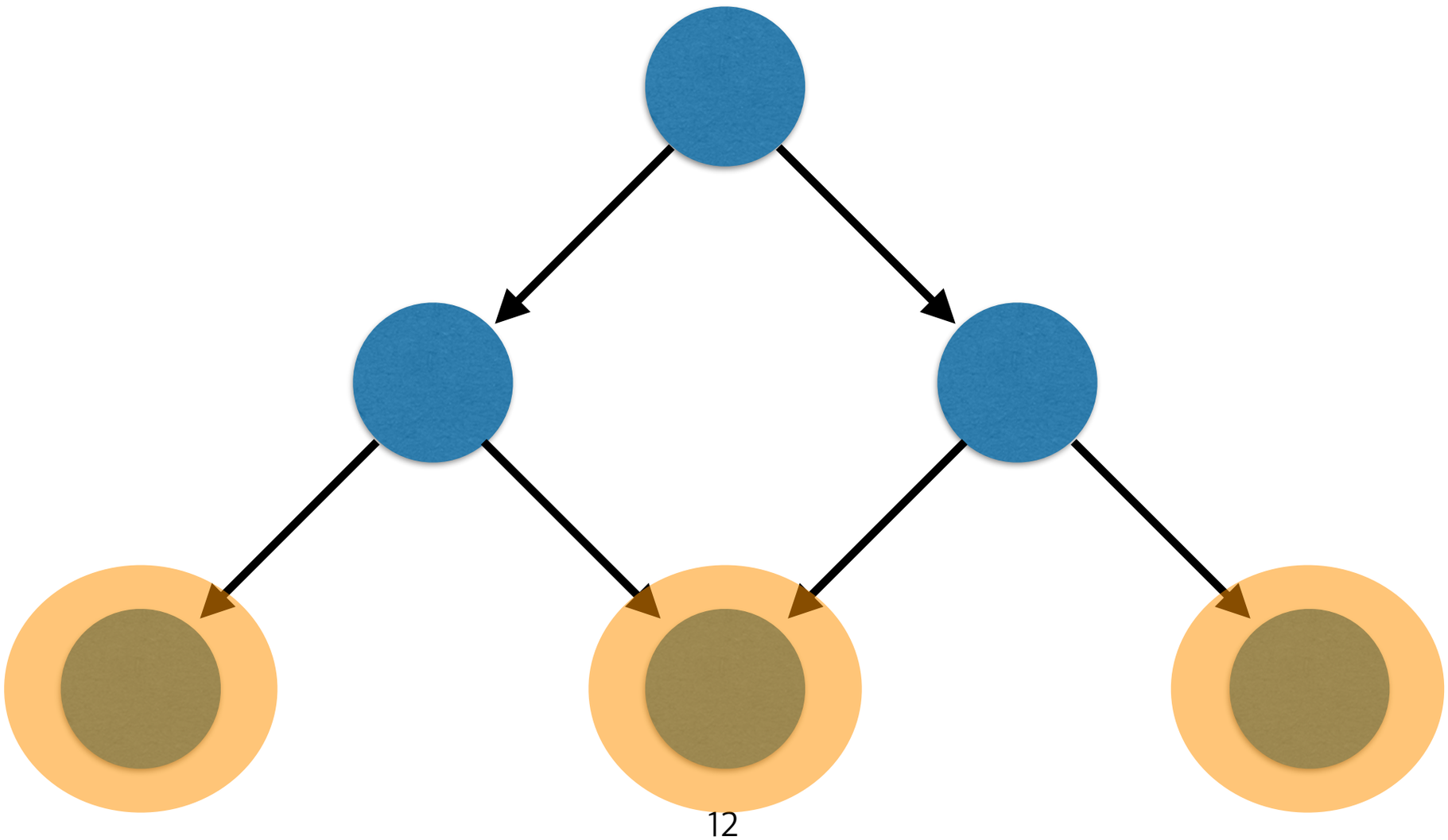
개별 분석 (modular analysis)

- 큰 프로그램의 각 부분(주로 함수)을 따로 분석 + 조합



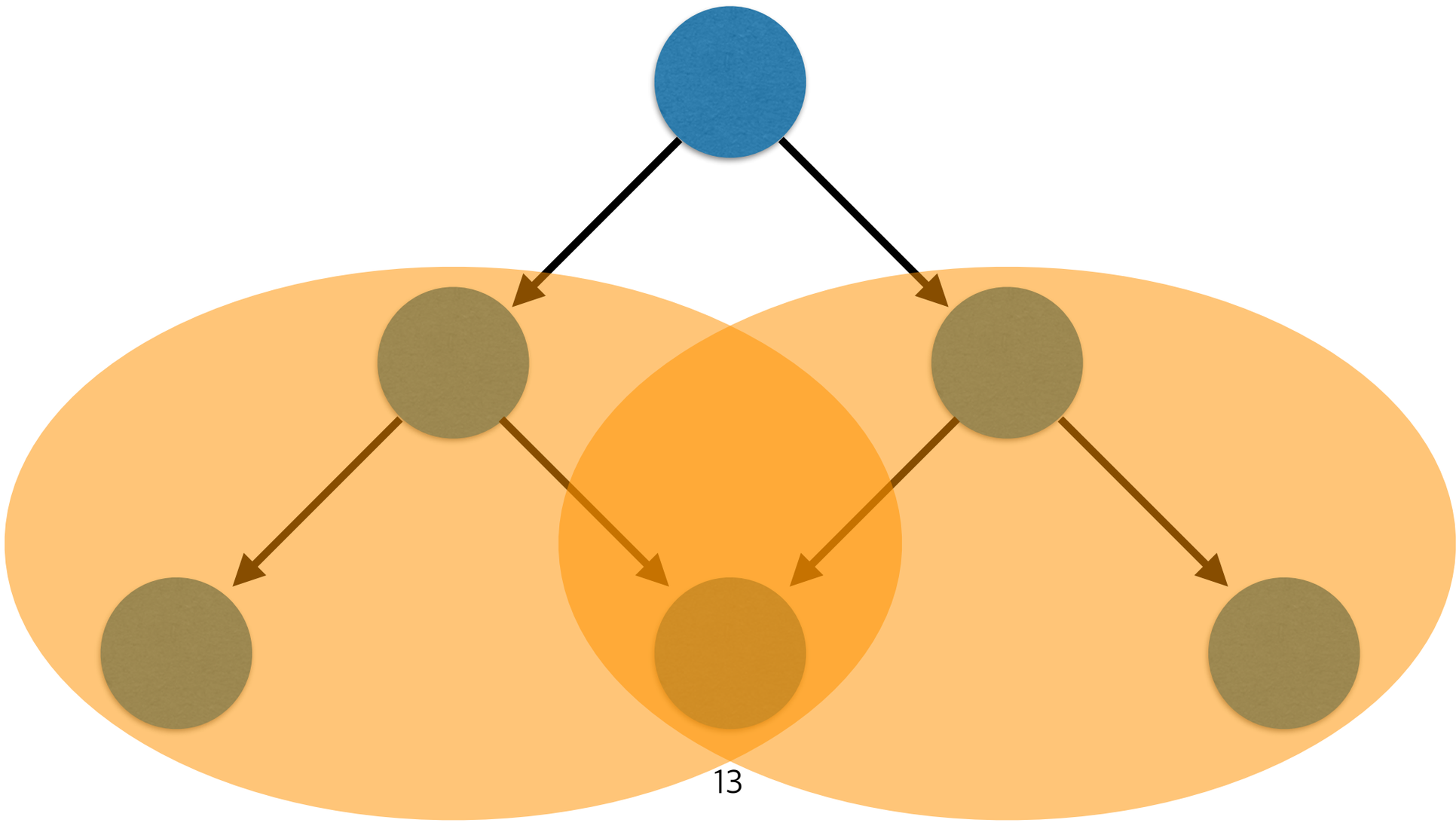
개별 분석 (modular analysis)

- 큰 프로그램의 각 부분(주로 함수)을 따로 분석 + 조합



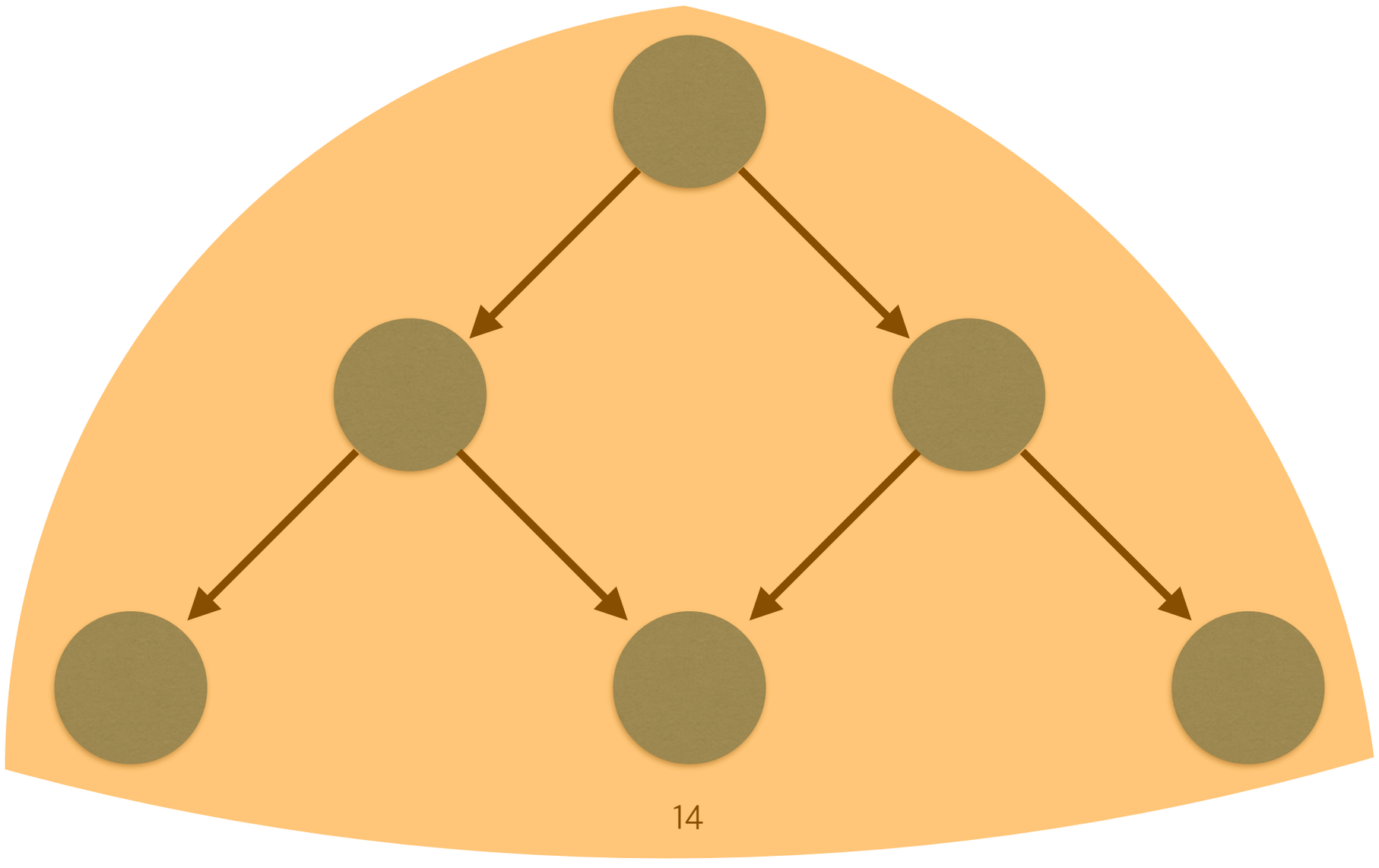
개별 분석 (modular analysis)

- 큰 프로그램의 각 부분(주로 함수)을 따로 분석 + 조합



개별 분석 (modular analysis)

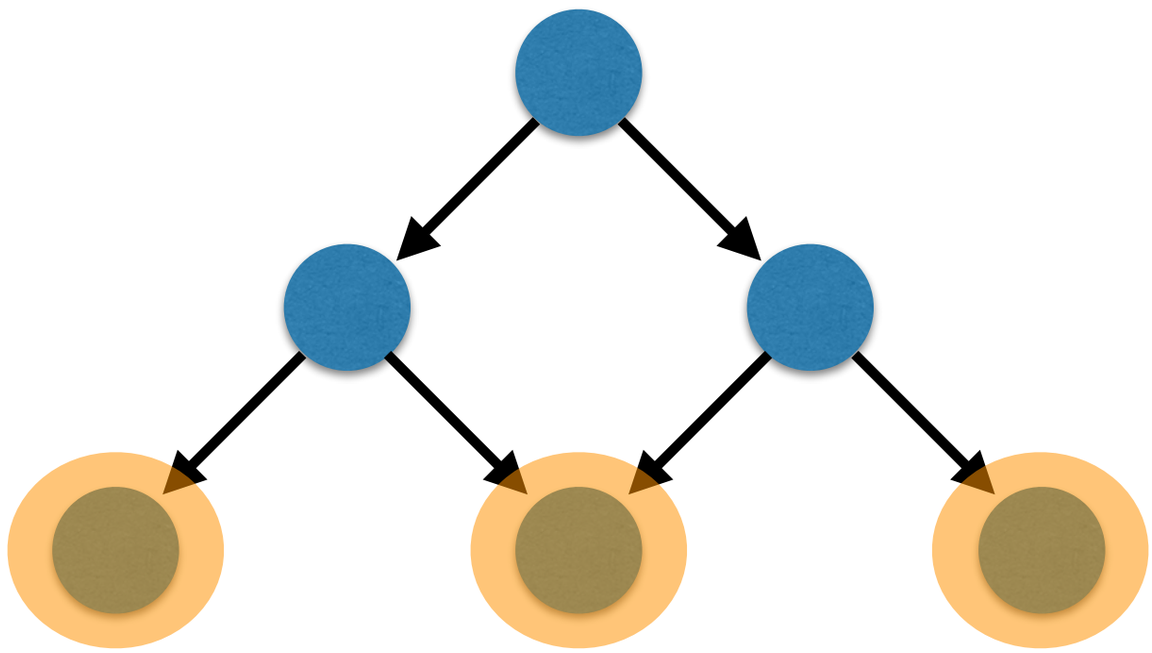
- 큰 프로그램의 각 부분(주로 함수)을 따로 분석 + 조합



개별 분석 (modular analysis)

(일반적으로)

- 장점: 각 함수를 한 번씩만 분석 => 빠름
- 단점: 문맥을 모름 => 부정확 혹은 불안전



개별 분석 (modular analysis)

- 장점 극대화, 단점 최소화하는 핵심 기술:
  부분 분석 결과를 재사용 가능하도록 잘 가공

- 전통적으로, 간단하거나 귀납적으로 잘 정의되는 성질 분석 (tree, list, 등)

- 예)

```
void foo(void* p){
  *p = 0;
}
```

p 가 null 인경우 위험한 함수

```
void bar(list* p){
  while(!p)
    p = p->next;
}
```

p 가 올바른 list 인경우 잘 순환하는 함수

Inferbo

- 개별분석 + 인터벌 도메인 기반 버퍼오버런 분석기
  - 인터벌: 복잡, 귀납적 x
- 도메인 디자인 원칙:
  - 재사용 가능한 인터벌 분석 결과 => Symbolic bound
  - Facebook 내부 코드에서 발견된 버그 패턴에 집중
- Infer 에 통합되어 Facebook 에서 사용 중

반응

오바다
2월 6일 ·

"Infer is an amazing robot suit for static analysis designers....Infer's modular analysis engine is a rocket that shoots us to the mesmerizing modular analysis orbit"

From Yi Kwangkeun's report on his sabbatical visit with the Infer team@Facebook, and the amazing work he did with his PhD students Kihong Heo and Sungkeun Cho

Inferbo: Infer-based buffer overrun analyzer

This is a report of my experience during a 2-month sabbatical I recently completed with the Facebook Infer static analysis team in London. During my sabbatical with my PhD students, we developed Inferbo, a static analyzer to detect C-like...

RESEARCH.FB.COM

좋아요 댓글 달기 공유하기

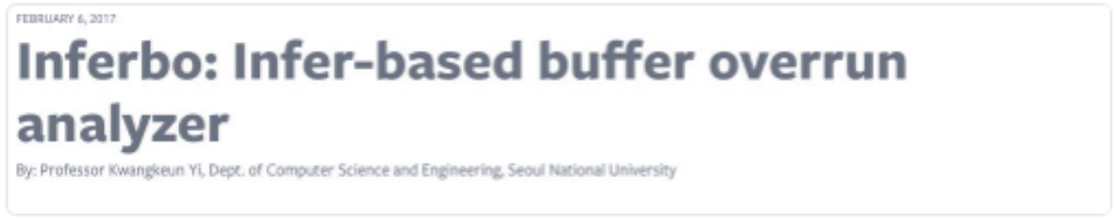
김동선님, Woosuk Lee님 외 41명

공유 4회

Infer @fbinfer · 2월 6일

An analyzer as realistic and modular as Inferbo in about 5 weeks? I would not have believed it". Prof Kwangkeun Yi research.fb.com/inferbo-infer-…

영어 번역하기



FEBRUARY 6, 2017

Inferbo: Infer-based buffer overrun analyzer

By: Professor Kwangkeun Yi, Dept. of Computer Science and Engineering, Seoul National University

Sam Blackshear님이 링크를 공유했습니다.
2월 7일

FB ❤ AbsInt: "[Infer's] extensible abstract interpretation framework is super-convenient to add a new kind of analysis like Inferbo, and its multi-lingual front end frees us from worrying about targeting varied source languages."

Facebook Academics
2월 7일 ·

A great overview of Inferbo: Infer-based buffer overrun analyze developed by Facebook visiting professor Professor Kwangkeun Yi, from the Dept. of Computer Science and Engineering at Seoul National University.

예제

```
char * malloc_wrapper(int n) {
 return malloc(n);
}
```

Memory:
  n |-> [s$0, s$1]
  ret |-> (offset: [0, 0], size: [s$0, s$1])

예제

```
void set_i(int* arr, int index){
 arr[index] = 0;
}
```

Memory:
  arr |-> (offset: [s$4, s$5], size: [s$6, s$7])
  index |-> [s$8, s$9]
Safety Condition:
  [s$4 + s$8, s$5 + s$9] < [s$6, s$7]

예제

```
char * maximum(int x) {
 if (x < 9) return x;
 else return 8
}
```

Memory:
x |-> [s$10, s$11]
ret |-> [min(8, s$10), 8]

예제

```
void interprocedural() {
  char *arr = malloc_wrapper(9);
  int i;
  for (i = 0; i < 9; i+=1) {
    set_i(arr, i); // safe
    set_i(arr, i + 1); // alarm
    set_i(arr, maximum(i+1)); //safe
  }
}
```

Summary of malloc_wrapper:
  n |-> [s$0, s$1]
  ret |-> (offset: [0, 0], size: [s$0, s$1])

Memory:
  arr |-> (offset: [0, 0], size: [9, 9])

예제

```
void interprocedural() {
 char *arr = malloc_wrapper(9);
 int i;
 for (i = 0; i < 9; i+=1) {
  set_i(arr, i); // safe
  set_i(arr, i + 1); // alarm
  set_i(arr, maximum(i+1)); //safe
 }
}
```

Summary of set_i:
  Memory:
    arr |-> (offset: [s$4, s$5], size: [s$6, s$7])
    index |-> [s$8, s$9]
  Safety Condition:
    [s$4 + s$8, s$5 + s$9] < [s$6, s$7]

Memory:
  arr |-> (offset: [0, 0], size: [9, 9])
  i |-> [0, 8]
Safety Condition
  [0 + 0, 0 + 8] < [9, 9]

예제

```
void interprocedural() {
 char *arr = malloc_wrapper(9);
 int i;
 for (i = 0; i < 9; i+=1) {
  set_i(arr, i); // safe
  set_i(arr, i + 1); // alarm
  set_i(arr, maximum(i+1)); //safe
 }
}
```

Summary of set_i:
  Memory:
    arr |-> (offset: [s$4, s$5], size: [s$6, s$7])
    index |-> [s$8, s$9]
  Safety Condition:
    [s$4 + s$8, s$5 + s$9] < [s$6, s$7]

Memory:
  arr |-> (offset: [0, 0], size: [9, 9])
  i |-> [1, 9]
Safety Condition
  [0 + 1, 0 + 9] < [9, 9]

예제

```
void interprocedural() {
 char *arr = malloc_wrapper(9);
 int i;
 for (i = 0; i < 9; i+=1) {
  set_i(arr, i); // safe
  set_i(arr, i + 1); // alarm
  set_i(arr, maximum(i+1)); //safe
 }
}
```

Summary of maximum:
Memory:
x |-> [s$10, s$11]
ret |-> [min(8, s$10), 8]

Memory:
arr |-> (offset: [0, 0], size: [9, 9])
i |-> [1, 9]

예제

```
void interprocedural() {
 char *arr = malloc_wrapper(9);
 int i;
 for (i = 0; i < 9; i+=1) {
  set_i(arr, i); // safe
  set_i(arr, i + 1); // alarm
  set_i(arr, maximum(i+1)); //safe
 }
}
```

Summary of maximum:
Memory:
x |-> [s$10, s$11]
ret |-> [min(8, s$10), 8]

Memory:
arr |-> (offset: [0, 0], size: [9, 9])
i |-> [1, 9]
Safety Condition
[0 + 1, 0 + 8] < [9, 9]

실험 결과

- 오픈 소스 프로그램에 수동으로 버그 삽입후 테스트

Software	KLoC	#injected bugs	#true alarms	#false alarms	Notes
spell	2	7	7	0	caught all, no false alarms
unhtml-2.3.9	2	11	6	0	missed 5/11, no false alarms
spell++-2.0.2	7	7	6	0	missed 1/7, no false alarms
bc-1.06	16	25	24	0	missed 1/25, no false alarms
gzip-1.8	61	5	5	6	caught all, 6 false alarms

도메인

$$\hat{State} = Node \to \hat{Mem}$$
$$\hat{Mem} = \hat{Loc} \to \hat{Val}$$
$$\hat{Val} = \hat{\mathbb{Z}} \times 2^{AllocSite \times \hat{\mathbb{Z}} \times \hat{\mathbb{Z}}}$$
$$\hat{\mathbb{Z}} = \{ [l, u] \mid l, u \in Bound \} \cup \{\bot\}$$
$$Bound = SymLinear \cup SymMinMax \cup \{-\infty, +\infty\}$$
$$SymLinear = \{c_0 + \Sigma c_i s_i \mid c_k \in \mathbb{Z}, s_i \in Symbols\}$$
$$SymMinMax = \{\mathsf{min}(c, s) \mid c \in \mathbb{Z}, s \in Symbols\} \cup \{\mathsf{max}(c, s) \mid c \in \mathbb{Z}, s \in Symbols\}$$

도메인

- Order

$$[l_1, u_1] \sqsubseteq [l_2, u_2] \quad \mathit{iff} \quad l_2 \mathbin{\hat{\leq}} l_1 \ \wedge \ u_1 \mathbin{\hat{\leq}} u_2$$

$$\begin{aligned}
c_0 + \Sigma c_i s_i \mathbin{\hat{\leq}} d_0 + \Sigma d_i s_i \quad & \mathit{iff} \quad c_0 \leq d_0 \ \wedge \ (\forall i \geq 1.\ c_i = d_i) \\
\mathsf{min}(c, s) \mathbin{\hat{\leq}} \mathsf{min}(d, s) \quad & \mathit{iff} \quad c \leq d \\
\mathsf{max}(c, s) \mathbin{\hat{\leq}} \mathsf{max}(c, s) \quad & \mathit{iff} \quad c \leq d \\
\mathsf{min}(c, s) \mathbin{\hat{\leq}} d_0 + d_1 s \quad & \mathit{iff} \quad (c \leq d_0 \ \wedge \ d_1 = 0) \ \vee \ (d_0 = 0 \ \wedge \ d_1 = 1) \\
c_0 + c_1 s \mathbin{\hat{\leq}} \mathsf{max}(d, s) \quad & \mathit{iff} \quad (c_0 \leq d \ \wedge \ c_1 = 0) \ \vee \ (c_0 = 0 \ \wedge \ c_1 = 1) \\
\mathsf{min}(c, s_1) \mathbin{\hat{\leq}} \mathsf{max}(d, s_2) \quad & \mathit{iff} \quad c \leq d \ \vee \ s_1 = s_2
\end{aligned}$$

도메인

- Order

$$[l_1, u_1] \sqsubseteq [l_2, u_2] \quad \mathit{iff} \quad l_2 \mathbin{\hat{\leq}} l_1 \ \wedge \ u_1 \mathbin{\hat{\leq}} u_2$$

$$c_0 + \Sigma c_i s_i \mathbin{\hat{\leq}} d_0 + \Sigma d_i s_i \quad \mathit{iff} \quad c_0 \leq d_0 \ \wedge \ (\forall i \geq 1.\ c_i = d_i)$$

$$\mathsf{min}(c, s) \leq \mathsf{min}(d, s) \quad \mathit{iff} \quad c \leq d$$

$$\mathsf{max}(c, s) \mathbin{\hat{\leq}} \mathsf{max}(c, s) \quad \mathit{iff} \quad c \leq d$$

$$\mathsf{min}(c, s) \leq d_0 + d_1 s \quad \mathit{iff} \quad (c \leq d_0 \ \wedge \ d_1 = 0) \ \vee \ (d_0 = 0 \ \wedge \ d_1 = 1)$$

$$c_0 + c_1 s \mathbin{\hat{\leq}} \mathsf{max}(d, s) \quad \mathit{iff} \quad (c_0 \leq d \ \wedge \ c_1 = 0) \ \vee \ (c_0 = 0 \ \wedge \ c_1 = 1)$$

$$\mathsf{min}(c, s_1) \mathbin{\hat{\leq}} \mathsf{max}(d, s_2) \quad \mathit{iff} \quad c \leq d \ \vee \ s_1 = s_2$$

도메인

- Join

$$[l_1, u_1] \sqcup [l_2, u_2] = [min(l_1, l_2), max(u_1, u_2)]$$

$$min(x, y) = \begin{cases} x & \text{if } x \hat{\leq} y \\ y & \text{if } y \hat{\leq} x \\ \mathsf{min}(c, s) & \text{if } x = c \ \wedge \ y = s \\ \mathsf{min}(c, s) & \text{if } y = c \ \wedge \ x = s \\ -\infty & \text{o.w.} \end{cases}$$

연산

$$[l_1, u_1] \ + \ [l_2, u_2] = [l_1 \ \hat{+}_l \ l_2, u_1 \ \hat{+}_u \ u_2]$$

$$x \ \hat{+}_l \ y = \begin{cases} (c_0 + d_0) + \Sigma(c_i + d_i)s_i & \text{if } x = c_0 + \Sigma c_i s_i \ \wedge \ y = d_0 + \Sigma d_i s_i \\ c + d & \text{if } x = c \ \wedge \ y = \mathsf{max}(d, s) \\ c + d & \text{if } x = \mathsf{max}(c, s) \ \wedge \ y = d \\ -\infty & \text{o.w.} \end{cases}$$

연산

$$[l_1, u_1] \ + \ [l_2, u_2] = [l_1 \ \hat{+}_l \ l_2, u_1 \ \hat{+}_u \ u_2]$$

$$x \ \hat{+}_l \ y = \begin{cases} (c_0 + d_0) + \Sigma(c_i + d_i)s_i & \text{if } x = c_0 + \Sigma c_i s_i \ \wedge \ y = d_0 + \Sigma d_i s_i \\ c + d & \text{if } x = c \ \wedge \ y = \mathsf{max}(d, s) \\ c + d & \text{if } x = \mathsf{max}(c, s) \ \wedge \ y = d \\ -\infty & \text{o.w.} \end{cases}$$

연산

$$prune : \hat{\mathbb{Z}} \times \hat{\mathbb{Z}} \to \hat{\mathbb{Z}}$$

$$prune_{\leq}(\,[l_1, u_1]\,, [l_2, u_2]\,) = \begin{cases} [l_1, u_2] & \text{if } u_1 = +\infty \\ [l_1, min(c_0, d_0) + \Sigma c_i s_i] & \text{if } u_1 = c_0 + \Sigma c_i s_i \ \wedge \ u_2 = d_0 + \Sigma d_i s_i \\ & \quad \wedge \ \forall i \geq 1.\ c_i = d_i \\ [l_1, \mathsf{min}(c, s)] & \text{if } (u_1 = c \ \wedge \ u_2 = s) \ \vee \ (u_1 = s \ \wedge \ u_2 = c) \\ [l_1, \mathsf{min}(min(c, d), s)] & \text{if } (u_1 = c \ \wedge \ u_2 = \mathsf{min}(d, s)) \\ & \quad \vee \ (u_1 = \mathsf{min}(c, s) \ \wedge \ u_2 = d) \\ & \quad \vee \ (u_1 = \mathsf{min}(c, s) \ \wedge \ u_2 = \mathsf{min}(d, s)) \\ [l_1, u_1] & \text{o.w.} \end{cases}$$

연산

$$prune : \hat{\mathbb{Z}} \times \hat{\mathbb{Z}} \to \hat{\mathbb{Z}}$$

$$prune_{\leq}(\texttt{[}l_1, u_1\texttt{]}, \texttt{[}l_2, u_2\texttt{]}) = \begin{cases} \texttt{[}l_1, u_2\texttt{]} & \text{if } u_1 = +\infty \\ \texttt{[}l_1, min(c_0, d_0) + \Sigma c_i s_i\texttt{]} & \text{if } u_1 = c_0 + \Sigma c_i s_i \ \wedge \ u_2 = d_0 + \Sigma d_i s_i \\ & \quad \wedge \ \forall i > 1.\ c_i = d_i \\ \texttt{[}l_1, \mathsf{min}(c, s)\texttt{]} & \text{if } (u_1 = c \ \wedge \ u_2 = s) \ \vee \ (u_1 = s \ \wedge \ u_2 = c) \\ \texttt{[}l_1, \mathsf{min}(min(c, d), s)\texttt{]} & \text{if } (u_1 = c \ \wedge \ u_2 = \mathsf{min}(d, s)) \\ & \quad \vee \ (u_1 = \mathsf{min}(c, s) \ \wedge \ u_2 = d) \\ & \quad \vee \ (u_1 = \mathsf{min}(c, s) \ \wedge \ u_2 = \mathsf{min}(d, s)) \\ \texttt{[}l_1, u_1\texttt{]} & \text{o.w.} \end{cases}$$

함수 서머리

- 정의

$$\sigma \to \lambda[s_1, s_2].\ \langle i, \kappa \rangle$$
$$\kappa \to \mathsf{True} \mid \kappa\ \wedge\ (i\ <\ i)$$

- 예) 함수 f 의 서머리

$$\sigma_f = \lambda[s_1, s_2].\langle i_f, \kappa_f \rangle$$

where

$$i_f = \bigsqcup_{i \in RetVal_f} i$$

$$\kappa_f = \bigwedge_{\langle i_{idx}, i_{size} \rangle \in BufAccess_f} (i_{idx} < i_{size})\ \wedge \bigwedge_{g([l,u]) \in CallSite_f} (\sigma_g[l, u]).2$$

함수 서머리

- 정의

$$\sigma \rightarrow \lambda[s_1, s_2].\ \langle i, \kappa \rangle$$

<리턴값, 조건식>

$$\kappa \rightarrow \mathsf{True} \mid \kappa\ \wedge\ (i\ <\ i)$$

- 예) 함수 f 의 서머리

$$\sigma_f = \lambda[s_1, s_2].\langle i_f, \kappa_f \rangle$$

where

$$i_f = \bigsqcup_{i \in RetVal_f} i$$

모든 리턴값 포섭

$$\kappa_f = \bigwedge_{\langle i_{idx}, i_{size} \rangle \in BufAccess_f} (i_{idx} < i_{size})\ \wedge \bigwedge_{g([l,u]) \in CallSite_f} (\sigma_g[l, u]).2$$

f 의 조건식

모든 callee 의 조건식

경험

- 잘 정리된 분석기 제작틀
  - multi-lingual & modular analysis framework
- OCaml 프로그래밍의 미학: 가독성 + 성능
- 전 세계와 함께 숨쉬는 즐거움 (공동 개발, 활발한 질답)

결론

- 개별분석 + 인터벌 도메인 기반 버퍼오버런 분석기
- 페이스북 내부 테스트를 통해 실용성 입증
- 계획: 여러 언어로 만든 프로그램에 테스트/튜닝 (C++, Java, Obj-C, etc)


THE FACEBOOK WALL
What's on your mind?
BOXIN' GATORS
INFERIBO/INFER
@burulia
AMTIMES
Happy Birthday Jaspreet
ATM
gig 'em Aggies!
Cassandra, Jaime '15
UA the best
@AMTIMES
HAPPY B'DAY JUDI!
Married
Kids were here

고맙습니다