성공적인 연구를 위한 문제 발견하기

전민석

2025. 07. 18

세미나 @ 소프트웨어 분석 연구실

연구 한다는 것?

연구 한다는 것?

- 아래의 4가지 요소를 일정 수준 이상 해내는 것

문제 + 아이디어 + 구현 + 글쓰기

문제 + 아이디어 + 구현 + 글쓰기

리뷰 점수 ↑

"**Supremely** well-written and is clearly situated within related work"

리뷰 점수 ↓

"**Weakness.** Technical presentation can be improved"

문제 + 아이디어 + 구현 + 글쓰기

리뷰 점수 ↑

“The anonymous GitHub repository seems pretty comprehensive”

리뷰 점수 ↓

“My score remains a C since the author response still did not clarify the implementation.”

문제 + 아이디어 + 구현 + 글쓰기

리뷰 점수 ↑

"The idea in this paper is already prompting more ideas in me for how to extend this work, which is a very good thing in a research paper"

리뷰 점수 ↓

"The paper combines several ideas, each of which is an increment on previous work."

문제 + 아이디어 + 구현 + 글쓰기

리뷰 점수 ↑

"The paper addresses an important problem advancing the state of the art in the very interesting and promising area of data-driven program analysis."

리뷰 점수 ↓

"I was less convinced of the motivation behind utilizing project-specific patterns"

어느것이 가장 어려울까?

문제 찾기

아이디어

구현

글쓰기

문제 찾기

방법을 아무도 안 알려줌

- 정답(정해진 방법)이 없음
- 사람마다 방법이 다름
- 논문에는 안적혀있거나/포장되어 적혀있음

아이디어 구현 글쓰기

배울 수 있는 자료(수업 또는 논문)가 있음

- 잘 알려진 방법이 있음
- 연구 지도를 받을 수 있음
- 논문에 적혀있음/구현체가 공개 되어 있음

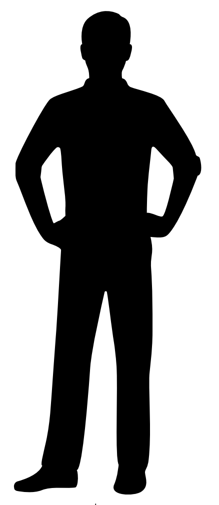
문제 발견 예시 1: 의심해보기

- 아래 문장은 어떤 동화를 한 문장으로 요약한 것이다, 어떤 동화를 요약한 것일까?

“The End”

- 아래 문장은 어떤 동화를 한 문장으로 요약한 것이다, 어떤 동화를 요약한 것일까?

“The End”



“간단하면서 보편적으로 적용 가능한 좋은 요약 방식이다”

Call-Site-Sensitivity (aka., k-CFA)

- The best-known flavor of context-sensitivity. It uses call-sites as contexts.
- In k-CFA, a method gets analyzed with the context that is a sequence of the last k call-sites (the current call-site of the method, the call-site of the caller method, the call-site of the caller method's caller, etc, up to a pre-defined depth, k).

Call-Site-Sensitivity (aka., k-CFA)

- The best-known flavor of context-sensitivity. It uses call-sites as contexts.
- In k-CFA, a method gets analyzed with the context that is a sequence of the last k call-sites (the current call-site of the method, the call-site of the caller method, the call-site of the caller method's caller, etc, up to a pre-defined depth, k).

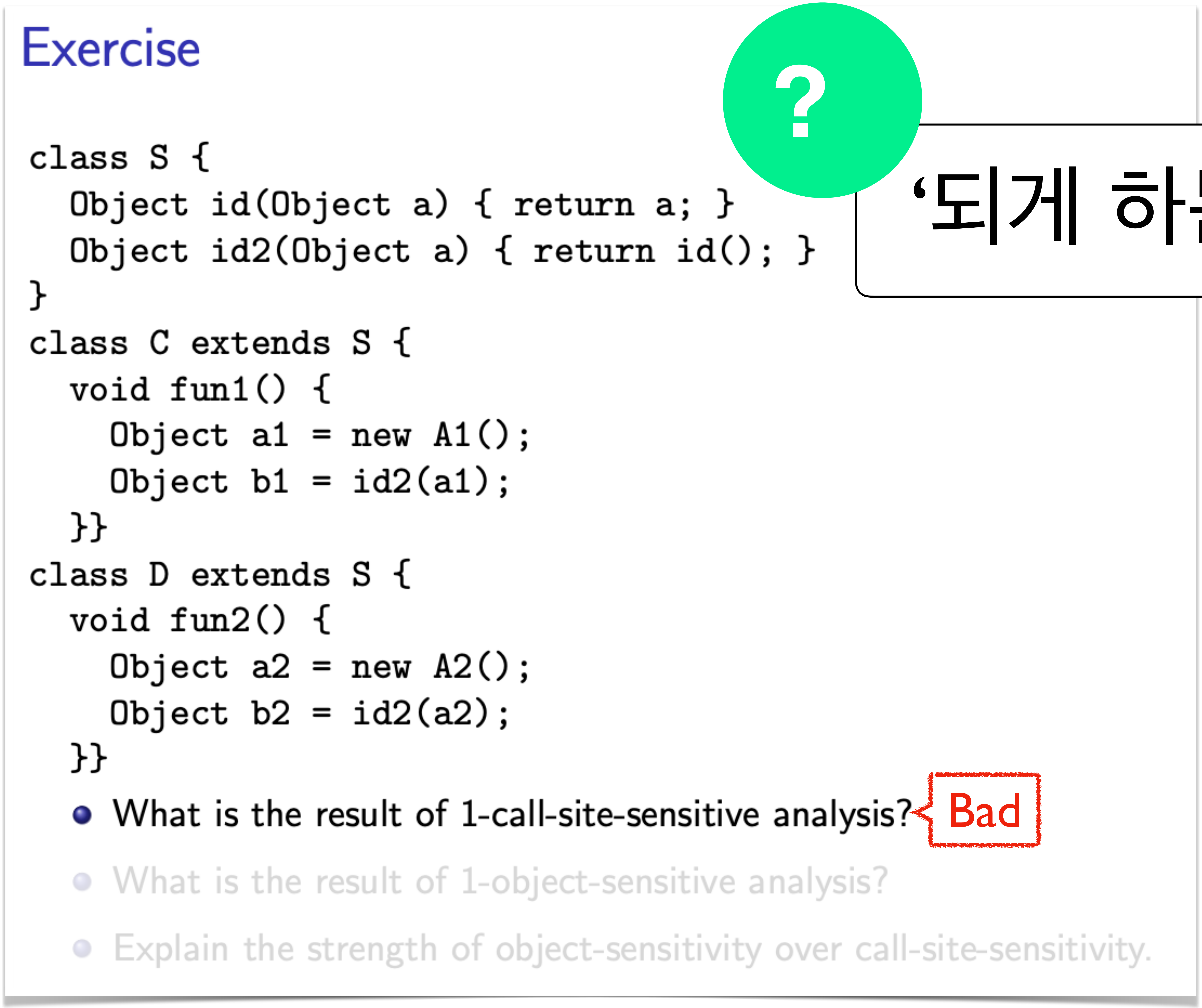
"A key part of the appeal of last k-based context abstraction is its simplicity and universal applicability."

- A reviewer [expert]

Exercise

```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

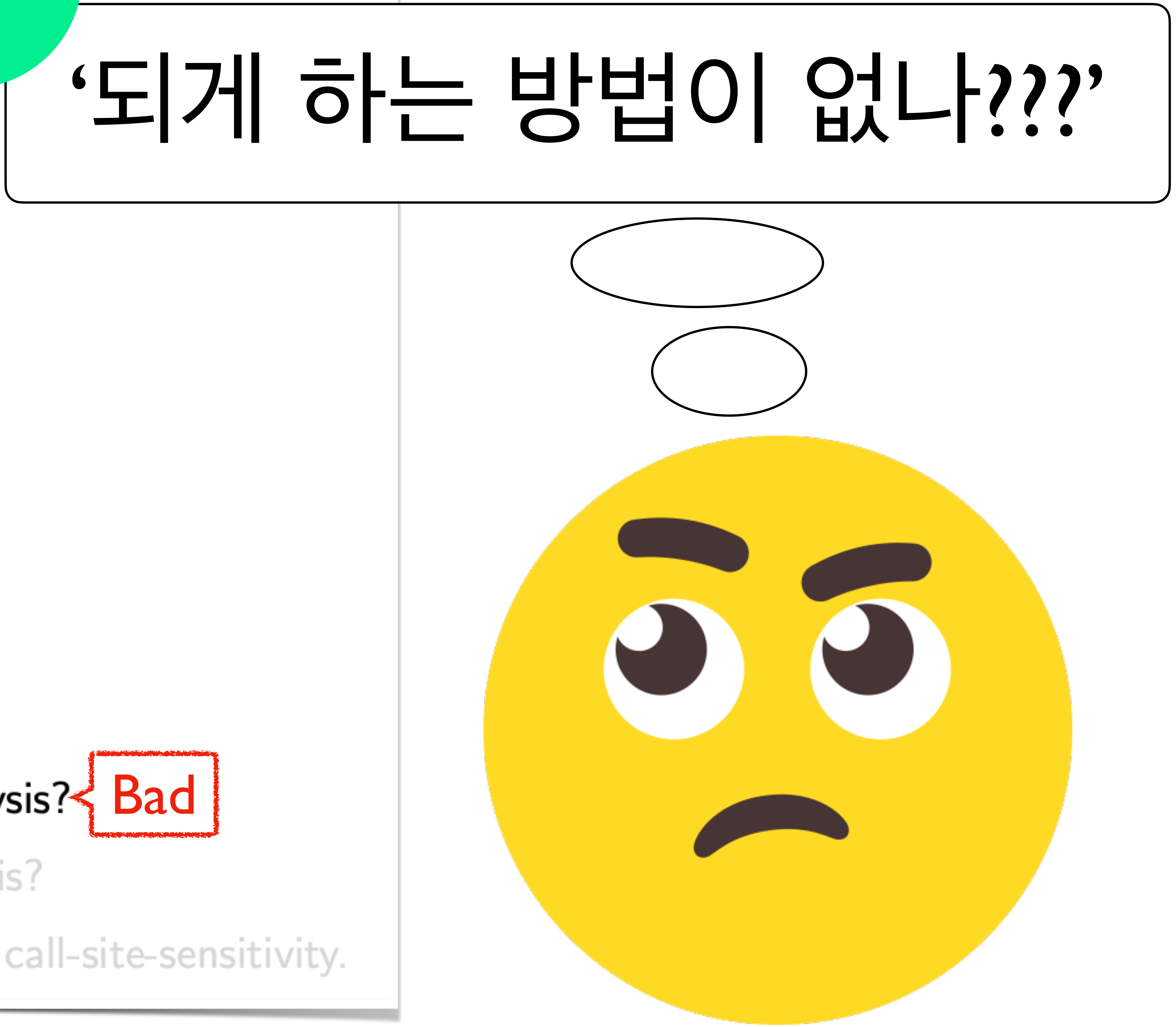
- What is the result of 1-call-site-sensitive analysis? Bad
- What is the result of 1-object-sensitive analysis? Good
- Explain the strength of object-sensitivity over call-site-sensitivity. Obj is better than Call



Exercise

```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis? Bad
- What is the result of 1-object-sensitive analysis?
- Explain the strength of object-sensitivity over call-site-sensitivity.



Exercise

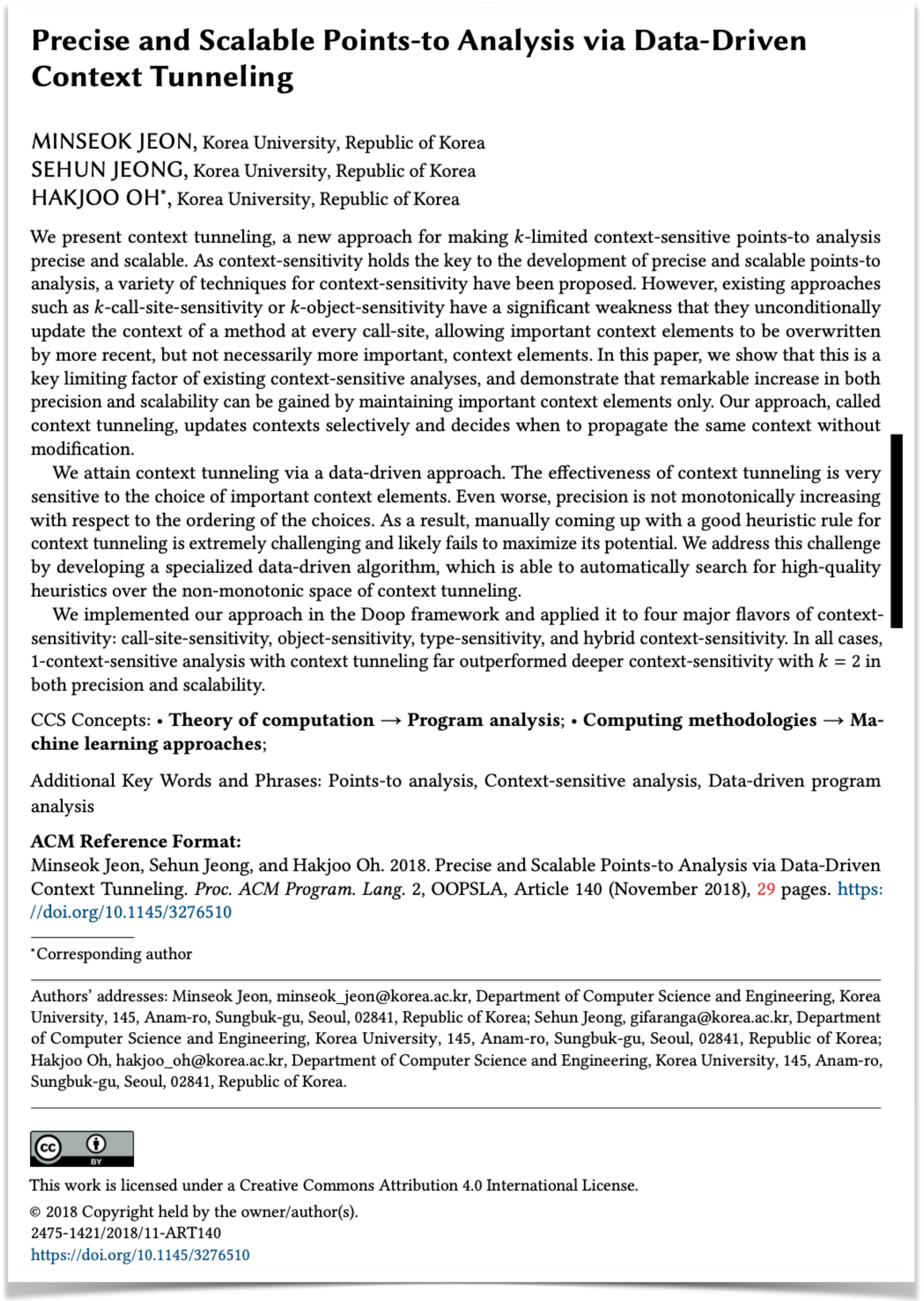
```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis?
- What is the result of 1-object-sensitive analysis?
- Explain the strength of object-sensitivity over call-site-sensitivity.

~~Bad~~
Good

last k -> important k

'이렇게 하면 되네!'



Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling

MINSEOK JEON, Korea University, Republic of Korea
SEHUN JEONG, Korea University, Republic of Korea
HAKJOO OH[*], Korea University, Republic of Korea

We present context tunneling, a new approach for making *k*-limited context-sensitive points-to analysis precise and scalable. As context-sensitivity holds the key to the development of precise and scalable points-to analysis, a variety of techniques for context-sensitivity have been proposed. However, existing approaches such as *k*-call-site-sensitivity or *k*-object-sensitivity have a significant weakness that they unconditionally update the context of a method at every call-site, allowing important context elements to be overwritten by more recent, but not necessarily more important, context elements. In this paper, we show that this is a key limiting factor of existing context-sensitive analyses, and demonstrate that remarkable increase in both precision and scalability can be gained by maintaining important context elements only. Our approach, called context tunneling, updates contexts selectively and decides when to propagate the same context without modification.

We attain context tunneling via a data-driven approach. The effectiveness of context tunneling is very sensitive to the choice of important context elements. Even worse, precision is not monotonically increasing with respect to the ordering of the choices. As a result, manually coming up with a good heuristic rule for context tunneling is extremely challenging and likely fails to maximize its potential. We address this challenge by developing a specialized data-driven algorithm, which is able to automatically search for high-quality heuristics over the non-monotonic space of context tunneling.

We implemented our approach in the Doop framework and applied it to four major flavors of context-sensitivity: call-site-sensitivity, object-sensitivity, type-sensitivity, and hybrid context-sensitivity. In all cases, 1-context-sensitive analysis with context tunneling far outperformed deeper context-sensitivity with $k = 2$ in both precision and scalability.

CCS Concepts: • **Theory of computation → Program analysis**; • **Computing methodologies → Machine learning approaches**;

Additional Key Words and Phrases: Points-to analysis, Context-sensitive analysis, Data-driven program analysis

ACM Reference Format:
Minseok Jeon, Sehun Jeong, and Hakjoo Oh. 2018. Precise and Scalable Points-to Analysis via Data-Driven Context Tunneling. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 140 (November 2018), 29 pages. https://doi.org/10.1145/3276510

[*]Corresponding author

Authors' addresses: Minseok Jeon, minseok_jeon@korea.ac.kr, Department of Computer Science and Engineering, Korea University, 145, Anam-ro, Sungbuk-gu, Seoul, 02841, Republic of Korea; Sehun Jeong, gifaranga@korea.ac.kr, Department of Computer Science and Engineering, Korea University, 145, Anam-ro, Sungbuk-gu, Seoul, 02841, Republic of Korea; Hakjoo Oh, hakjoo_oh@korea.ac.kr, Department of Computer Science and Engineering, Korea University, 145, Anam-ro, Sungbuk-gu, Seoul, 02841, Republic of Korea.

- k-context-sensitivity have a significant weakness that they retain last k context elements

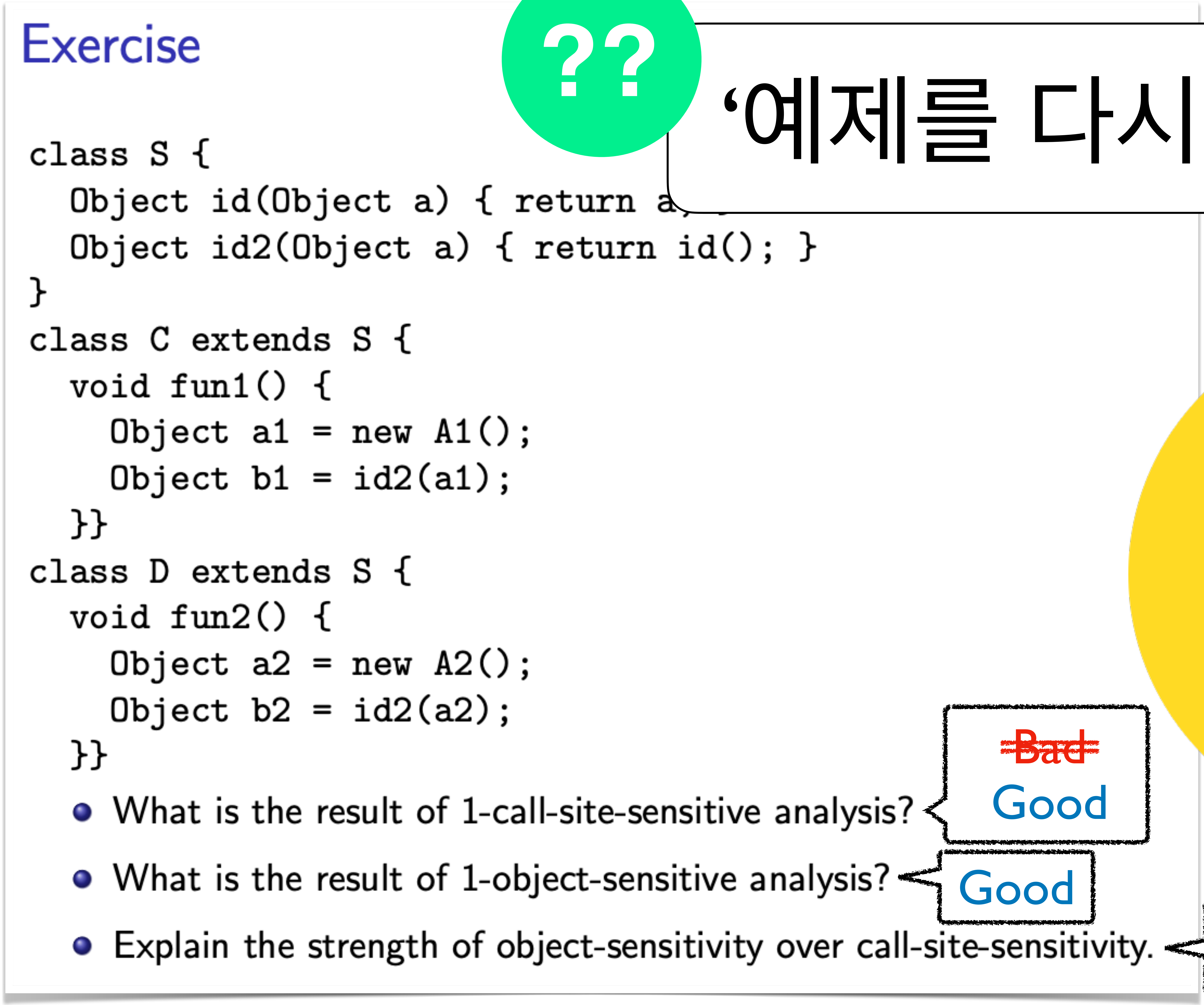
- We demonstrate that remarkable increase in precision can be gained by maintaining important context elements.

문제 발견 예시 2: 의심해보기 2

Exercise

```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis? Bad
- What is the result of 1-object-sensitive analysis? Good
- Explain the strength of object-sensitivity over call-site-sensitivity. Obj is better than Call



Exercise

```
class S {
  Object id(Object a) { return a
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis?
- What is the result of 1-object-sensitive analysis?
- Explain the strength of object-sensitivity over call-site-sensitivity.

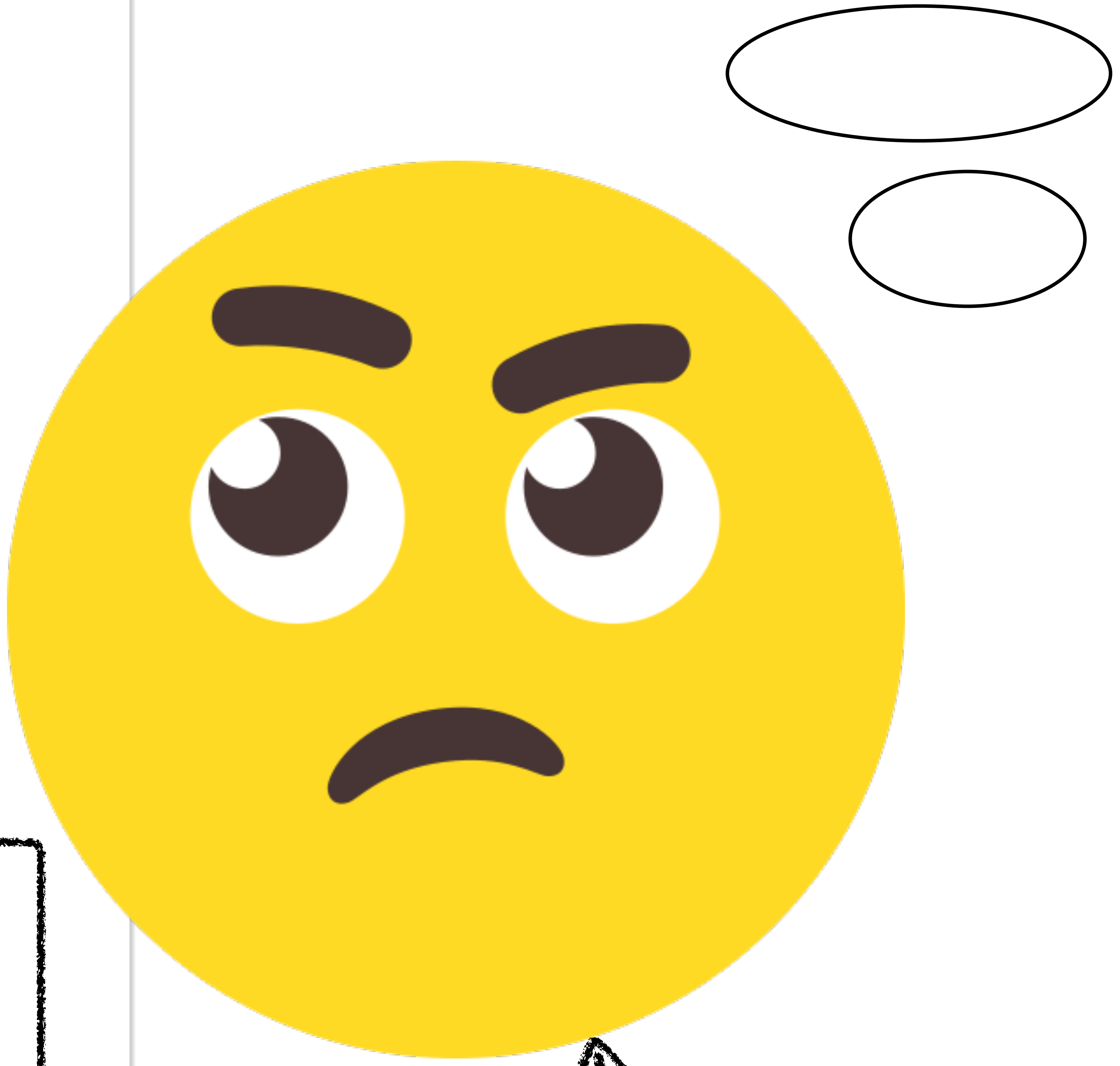
??

'예제를 다시 만들어야 하네?'

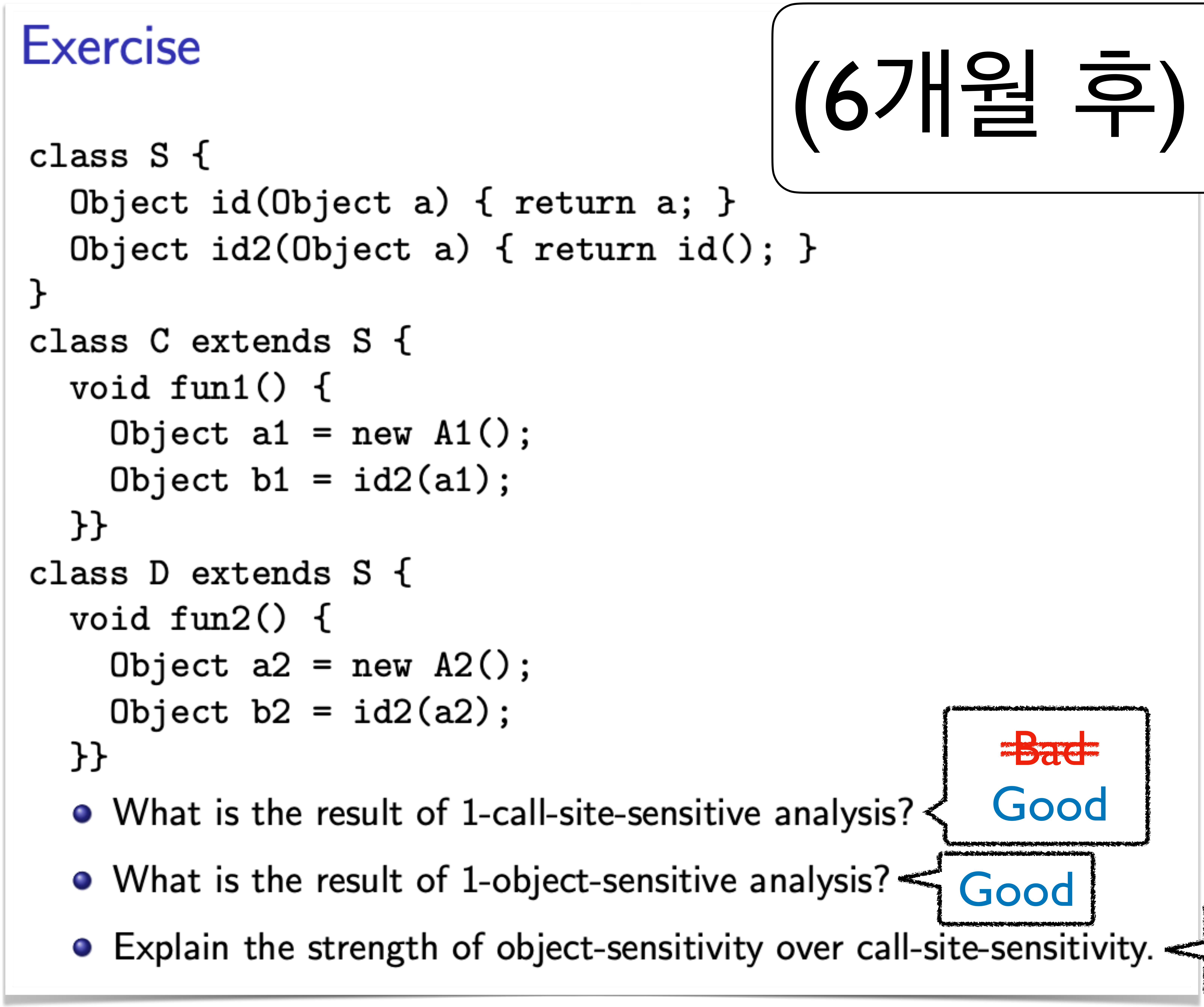
~~Bad~~
Good

Good

~~Obj is better than Call~~



(6개월 후) '안만들어지네??'



Exercise

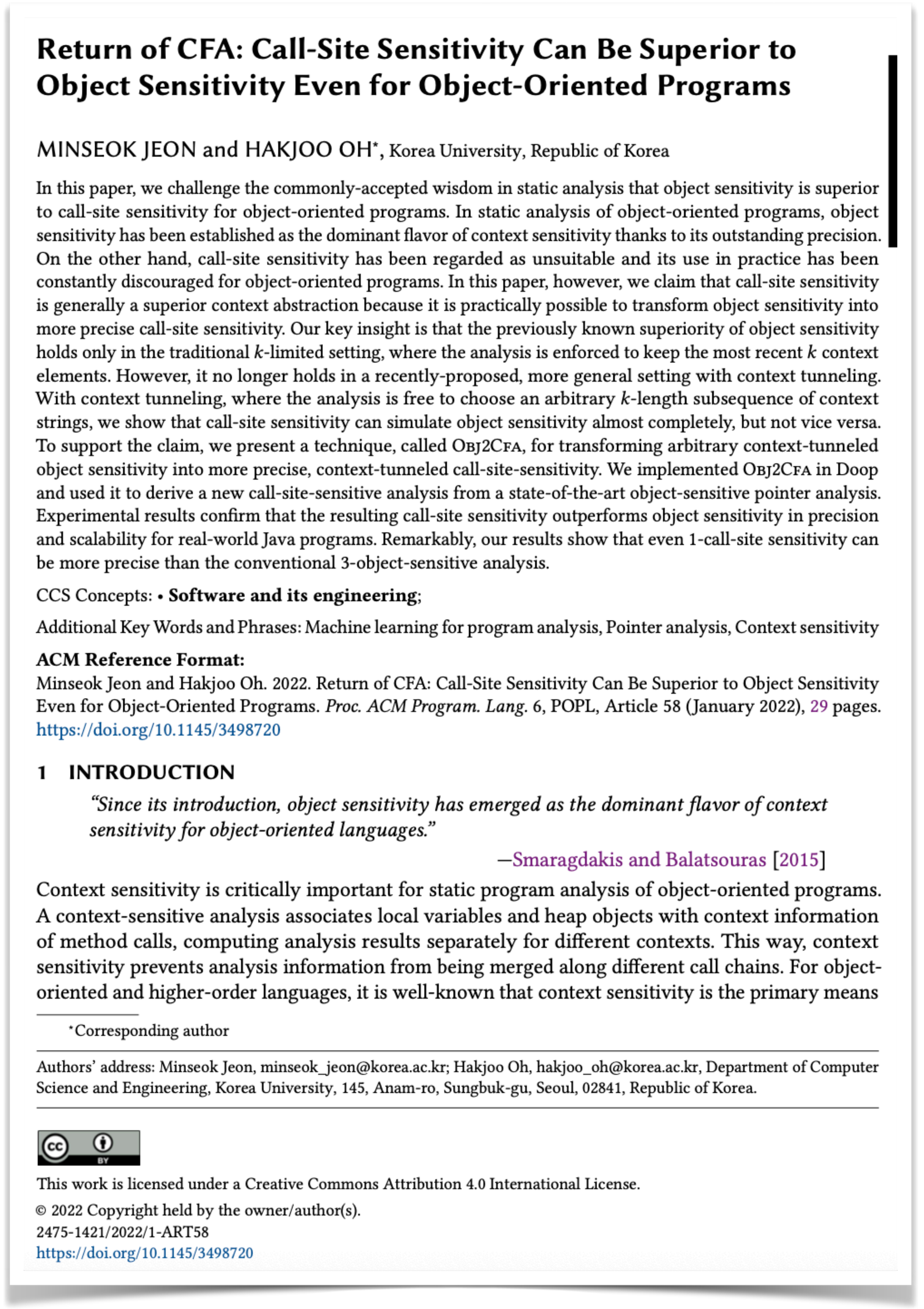
```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis?
- What is the result of 1-object-sensitive analysis?
- Explain the strength of object-sensitivity over call-site-sensitivity.

~~Bad~~
Good

Good

~~Obj is better than Call~~



Return of CFA: Call-Site Sensitivity Can Be Superior to Object Sensitivity Even for Object-Oriented Programs

MINSEOK JEON and HAKJOO OH*, Korea University, Republic of Korea

In this paper, we challenge the commonly-accepted wisdom in static analysis that object sensitivity is superior to call-site sensitivity for object-oriented programs. In static analysis of object-oriented programs, object sensitivity has been established as the dominant flavor of context sensitivity thanks to its outstanding precision. On the other hand, call-site sensitivity has been regarded as unsuitable and its use in practice has been constantly discouraged for object-oriented programs. In this paper, however, we claim that call-site sensitivity is generally a superior context abstraction because it is practically possible to transform object sensitivity into more precise call-site sensitivity. Our key insight is that the previously known superiority of object sensitivity holds only in the traditional *k*-limited setting, where the analysis is enforced to keep the most recent *k* context elements. However, it no longer holds in a recently-proposed, more general setting with context tunneling. With context tunneling, where the analysis is free to choose an arbitrary *k*-length subsequence of context strings, we show that call-site sensitivity can simulate object sensitivity almost completely, but not vice versa. To support the claim, we present a technique, called Obj2Cfa, for transforming arbitrary context-tunneled object sensitivity into more precise, context-tunneled call-site-sensitivity. We implemented Obj2Cfa in Doop and used it to derive a new call-site-sensitive analysis from a state-of-the-art object-sensitive pointer analysis. Experimental results confirm that the resulting call-site sensitivity outperforms object sensitivity in precision and scalability for real-world Java programs. Remarkably, our results show that even 1-call-site sensitivity can be more precise than the conventional 3-object-sensitive analysis.

CCS Concepts: • **Software and its engineering**;

Additional Key Words and Phrases: Machine learning for program analysis, Pointer analysis, Context sensitivity

ACM Reference Format:
Minseok Jeon and Hakjoo Oh. 2022. Return of CFA: Call-Site Sensitivity Can Be Superior to Object Sensitivity Even for Object-Oriented Programs. *Proc. ACM Program. Lang.* 6, POPL, Article 58 (January 2022), 29 pages. https://doi.org/10.1145/3498720

1 INTRODUCTION

"Since its introduction, object sensitivity has emerged as the dominant flavor of context sensitivity for object-oriented languages."

—Smaragdakis and Balatsouras [2015]

Context sensitivity is critically important for static program analysis of object-oriented programs. A context-sensitive analysis associates local variables and heap objects with context information of method calls, computing analysis results separately for different contexts. This way, context sensitivity prevents analysis information from being merged along different call chains. For object-oriented and higher-order languages, it is well-known that context sensitivity is the primary means

*Corresponding author

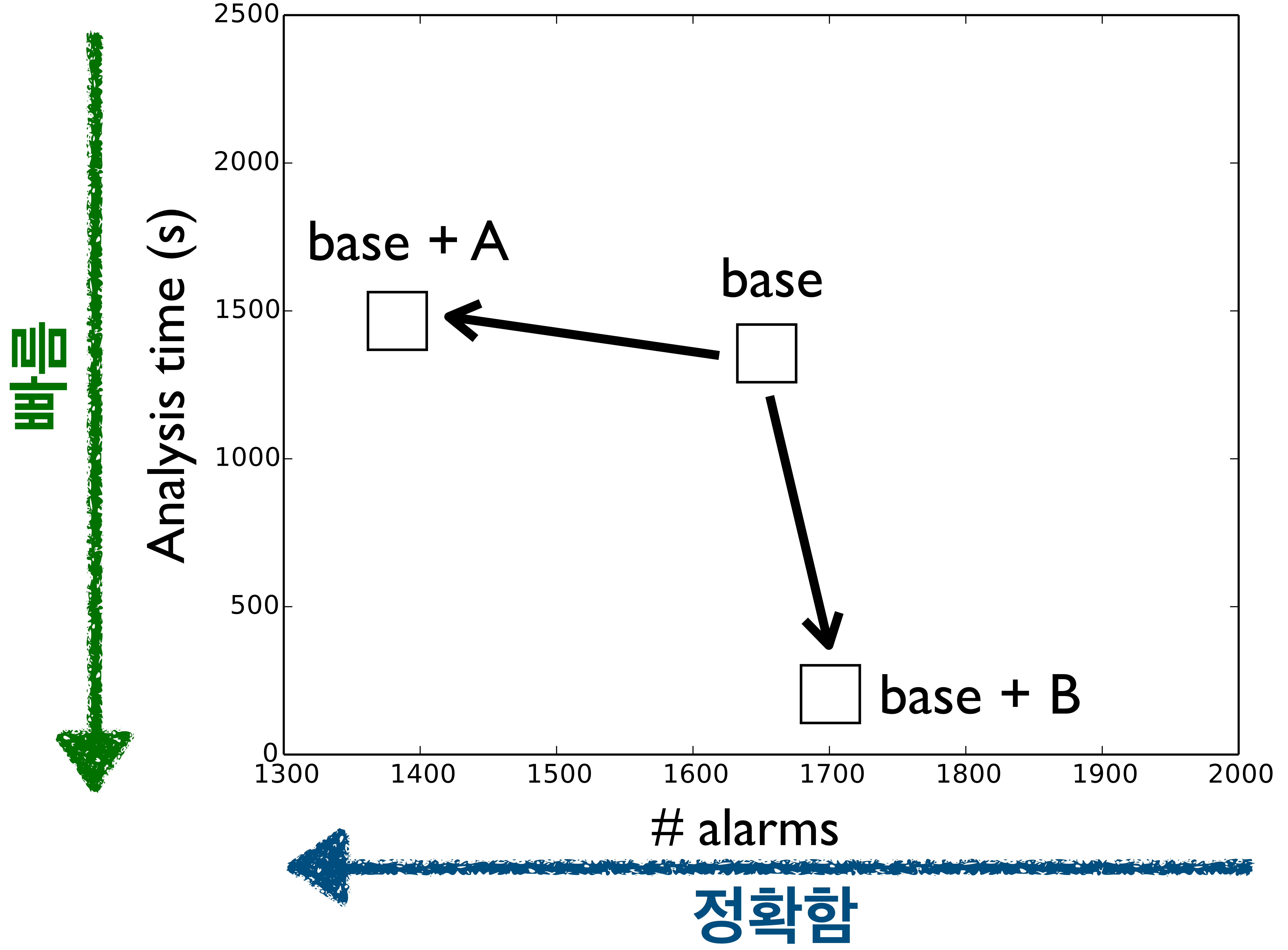
Authors' address: Minseok Jeon, minseok_jeon@korea.ac.kr; Hakjoo Oh, hakjoo_oh@korea.ac.kr, Department of Computer Science and Engineering, Korea University, 145, Anam-ro, Sungbuk-gu, Seoul, 02841, Republic of Korea.

- **The Status Quo**. Since its inception, object sensitivity has been established as the dominant context abstraction for object-oriented languages

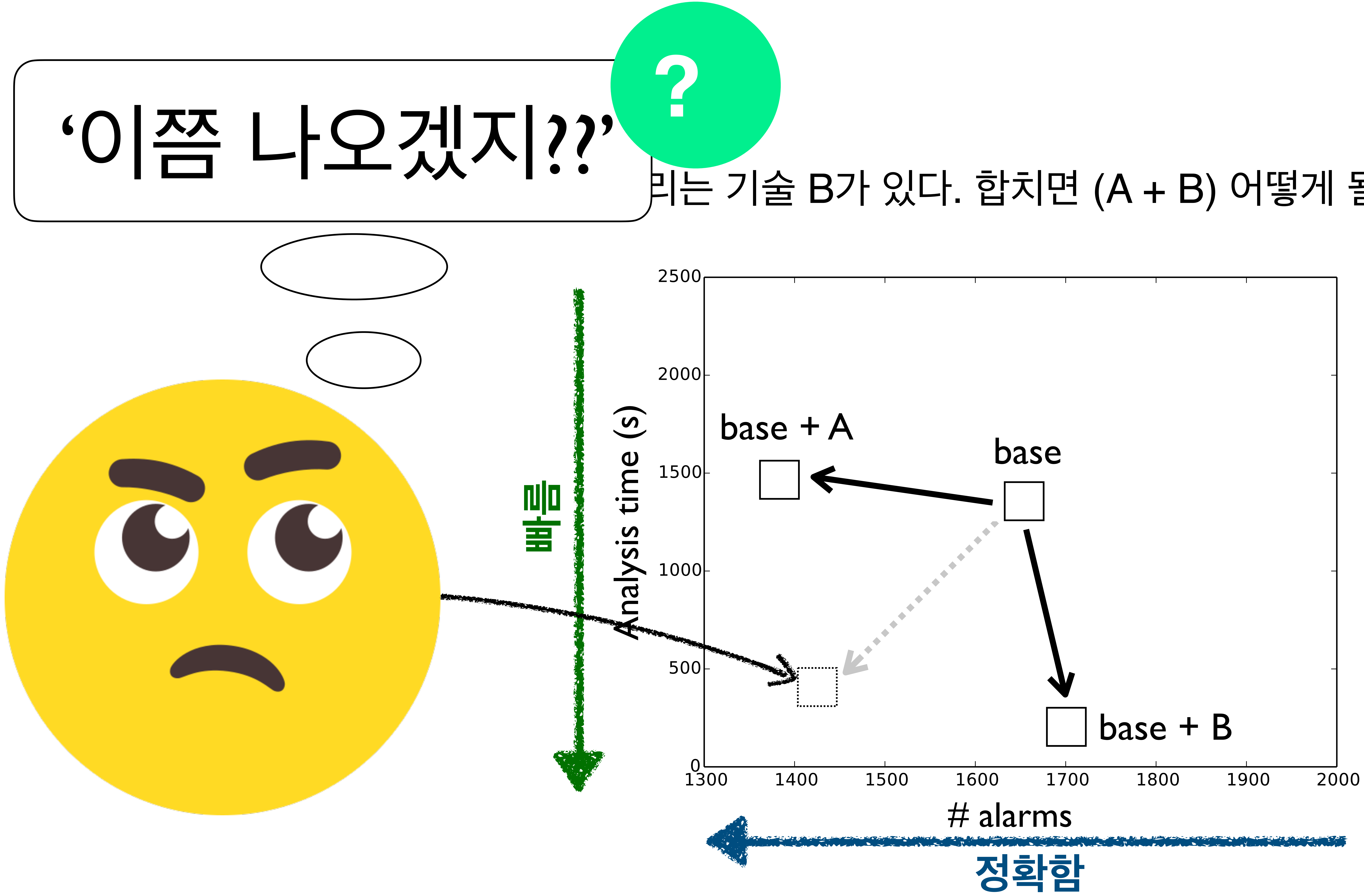
- **This Work.** We challenge this commonly-accepted wisdom by showing that call-site sensitivity is generally superior to object sensitivity even for object-oriented programs.

문제 발견 예시 3 : 이상함 감지

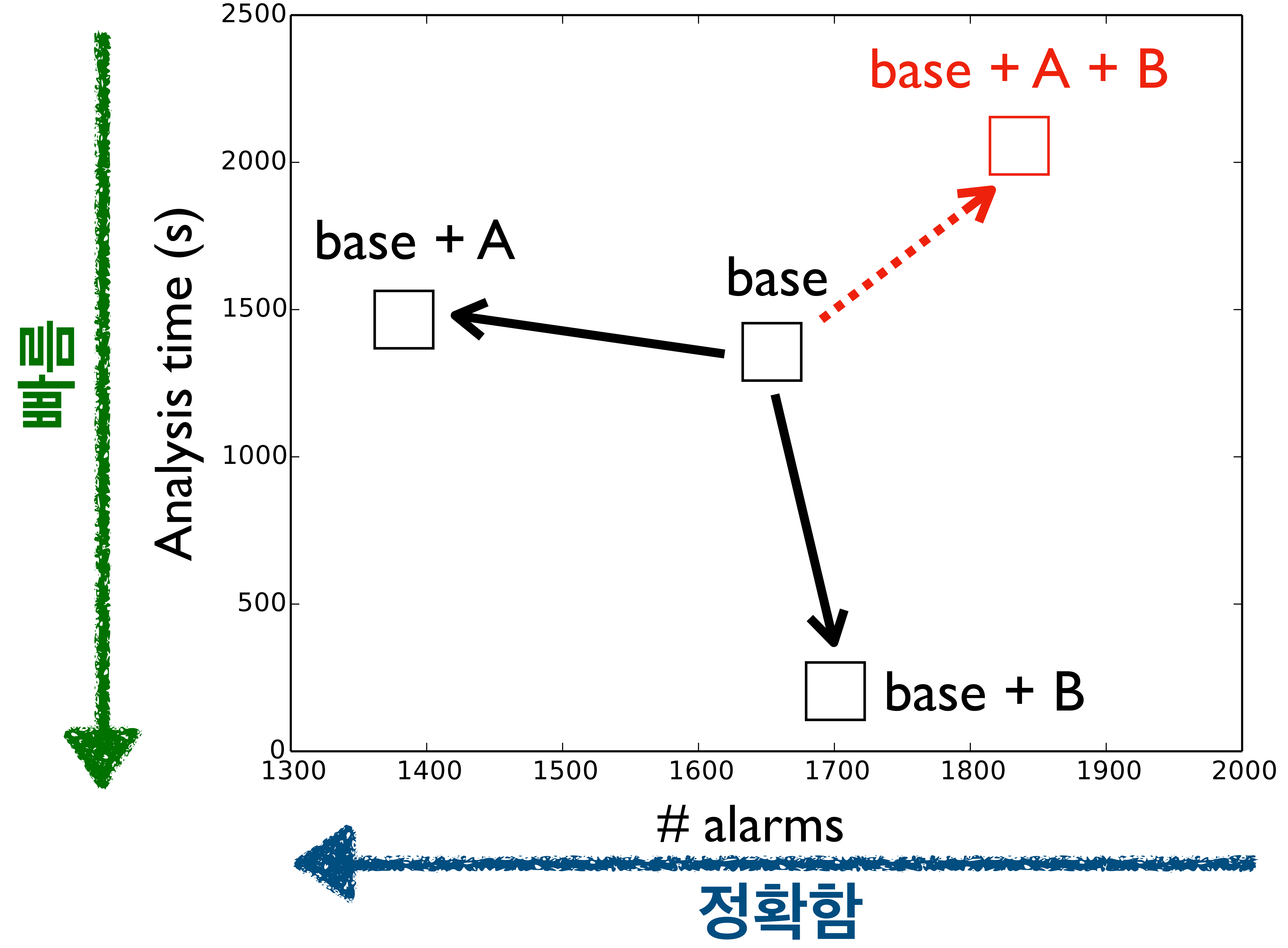
- 정확도를 올리는 기술 A와 속도를 올리는 기술 B가 있다. 합치면 (A + B) 어떻게 될까?

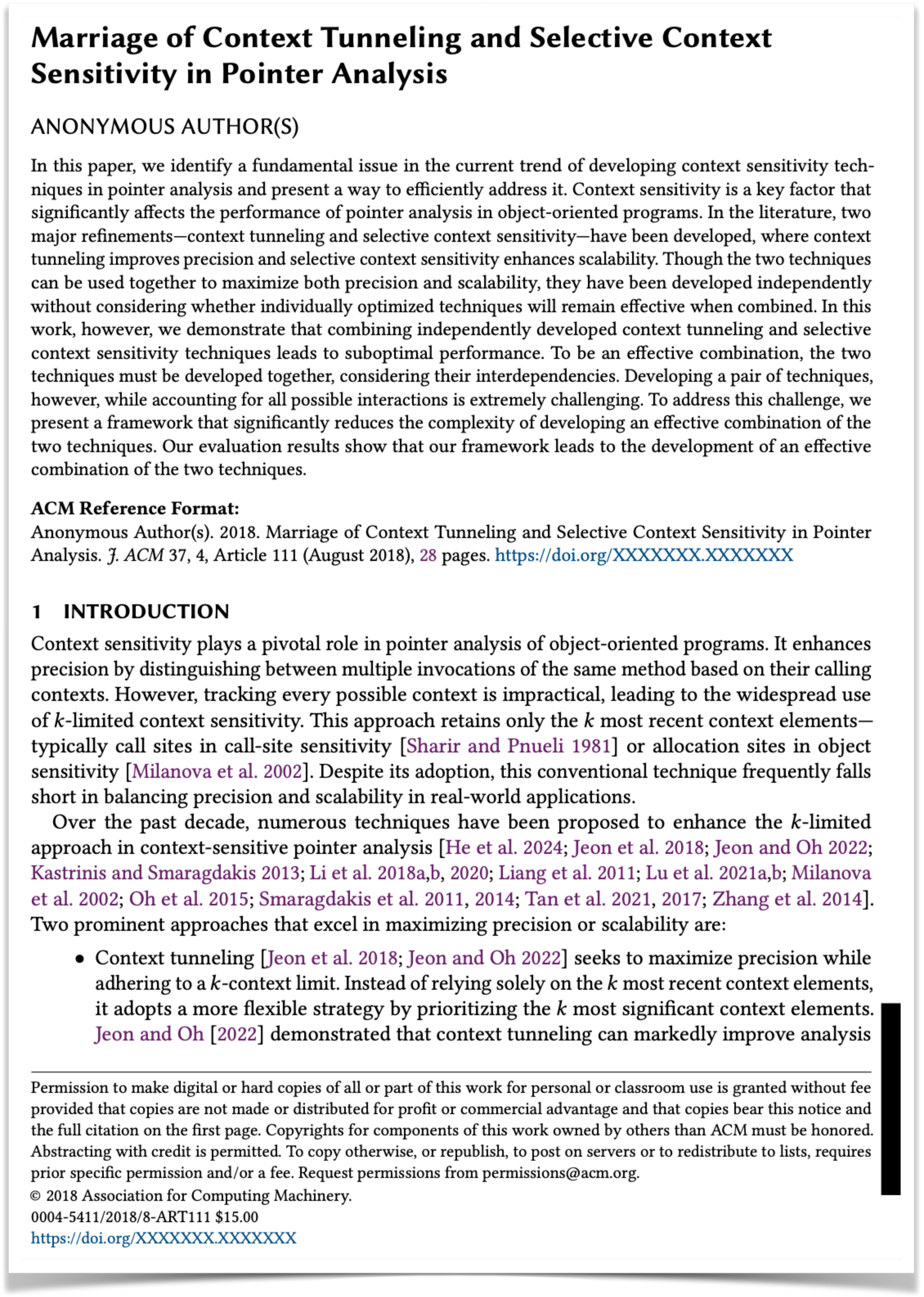


리는 기술 B가 있다. 합치면 (A + B) 어떻게 될까?



- 정확도를 올리는 기술 A와 속도를 올리는 기술 B가 있다. 합치면 (A + B) 어떻





Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis

ANONYMOUS AUTHOR(S)

In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis and present a way to efficiently address it. Context sensitivity is a key factor that significantly affects the performance of pointer analysis in object-oriented programs. In the literature, two major refinements—context tunneling and selective context sensitivity—have been developed, where context tunneling improves precision and selective context sensitivity enhances scalability. Though the two techniques can be used together to maximize both precision and scalability, they have been developed independently without considering whether individually optimized techniques will remain effective when combined. In this work, however, we demonstrate that combining independently developed context tunneling and selective context sensitivity techniques leads to suboptimal performance. To be an effective combination, the two techniques must be developed together, considering their interdependencies. Developing a pair of techniques, however, while accounting for all possible interactions is extremely challenging. To address this challenge, we present a framework that significantly reduces the complexity of developing an effective combination of the two techniques. Our evaluation results show that our framework leads to the development of an effective combination of the two techniques.

ACM Reference Format:
Anonymous Author(s). 2018. Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis. *J. ACM* 37, 4, Article 111 (August 2018), 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Context sensitivity plays a pivotal role in pointer analysis of object-oriented programs. It enhances precision by distinguishing between multiple invocations of the same method based on their calling contexts. However, tracking every possible context is impractical, leading to the widespread use of *k*-limited context sensitivity. This approach retains only the *k* most recent context elements—typically call sites in call-site sensitivity [Sharir and Pnueli 1981] or allocation sites in object sensitivity [Milanova et al. 2002]. Despite its adoption, this conventional technique frequently falls short in balancing precision and scalability in real-world applications.

Over the past decade, numerous techniques have been proposed to enhance the *k*-limited approach in context-sensitive pointer analysis [He et al. 2024; Jeon et al. 2018; Jeon and Oh 2022; Kastrinis and Smaragdakis 2013; Li et al. 2018a,b, 2020; Liang et al. 2011; Lu et al. 2021a,b; Milanova et al. 2002; Oh et al. 2015; Smaragdakis et al. 2011, 2014; Tan et al. 2021, 2017; Zhang et al. 2014]. Two prominent approaches that excel in maximizing precision or scalability are:

- Context tunneling [Jeon et al. 2018; Jeon and Oh 2022] seeks to maximize precision while adhering to a *k*-context limit. Instead of relying solely on the *k* most recent context elements, it adopts a more flexible strategy by prioritizing the *k* most significant context elements. Jeon and Oh [2022] demonstrated that context tunneling can markedly improve analysis

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

- In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis

- We demonstrate that combining independently developed context tunneling and selective context sensitivity techniques leads to suboptimal performance.

번외:
내가 만든 문제 vs 남이 만든 문제

내가 만든 문제

- 선행 연구가 **없음**

남이 만든 문제

- 선행 연구가 **존재함**

내가 만든 문제

- 선행 연구가 **없음**
  - 독자 설득이 어려움
  - 평가 방법을 만들어야 함
  - 비교군이 없음

남이 만든 문제

- 선행 연구가 **존재함**
  - 독자 설득이 쉬움
  - 비교군이 존재
  - 평가 방법이 존재

내가 만든 문제

- 선행 연구가 **없음**
  - 경쟁자가 없음
  - 문제에 대한 몰입/애착 큼
    - 위기에 강함

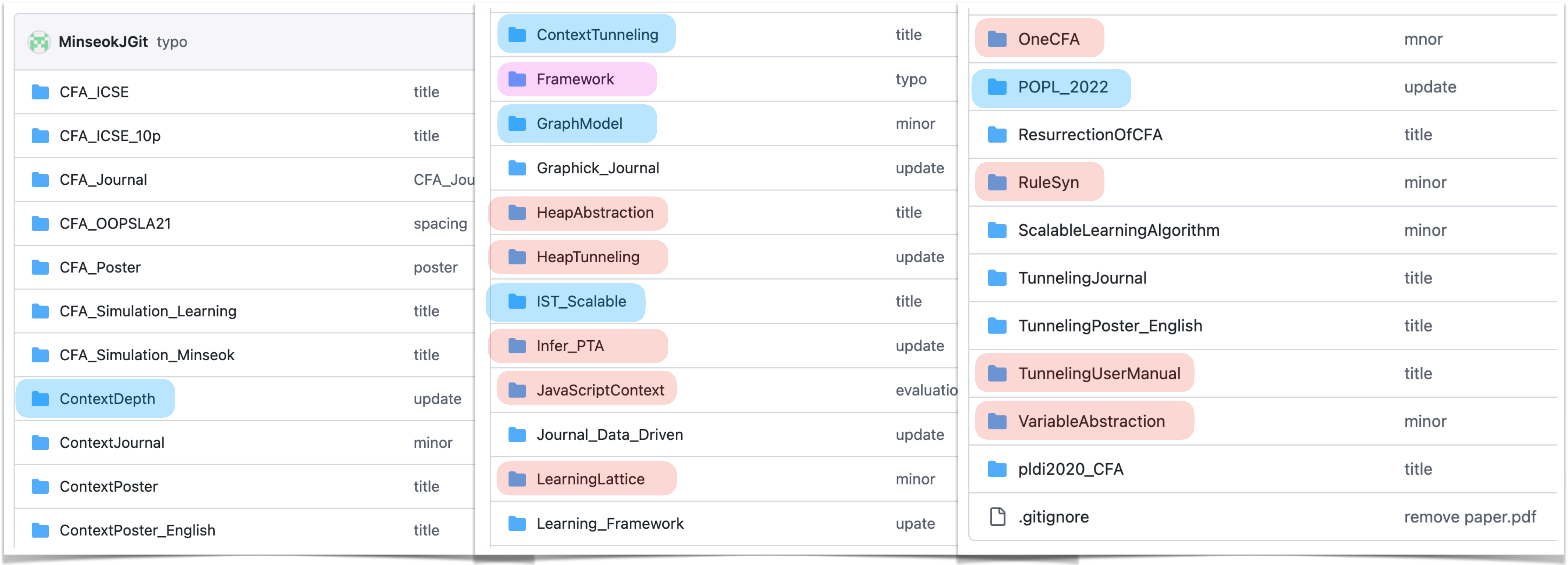
남이 만든 문제

- 선행 연구가 **존재함**
  - 경쟁자가 많음
  - 문제에 대한 몰입/애착 작음
    - 위기에 취약함

연습 방법: 많이 시도 해보기 + 실패 두려워 하지 않기

🟦 : 성공한 연구 (논문 출판)

🟥 : 실패한 연구 (중도 포기)

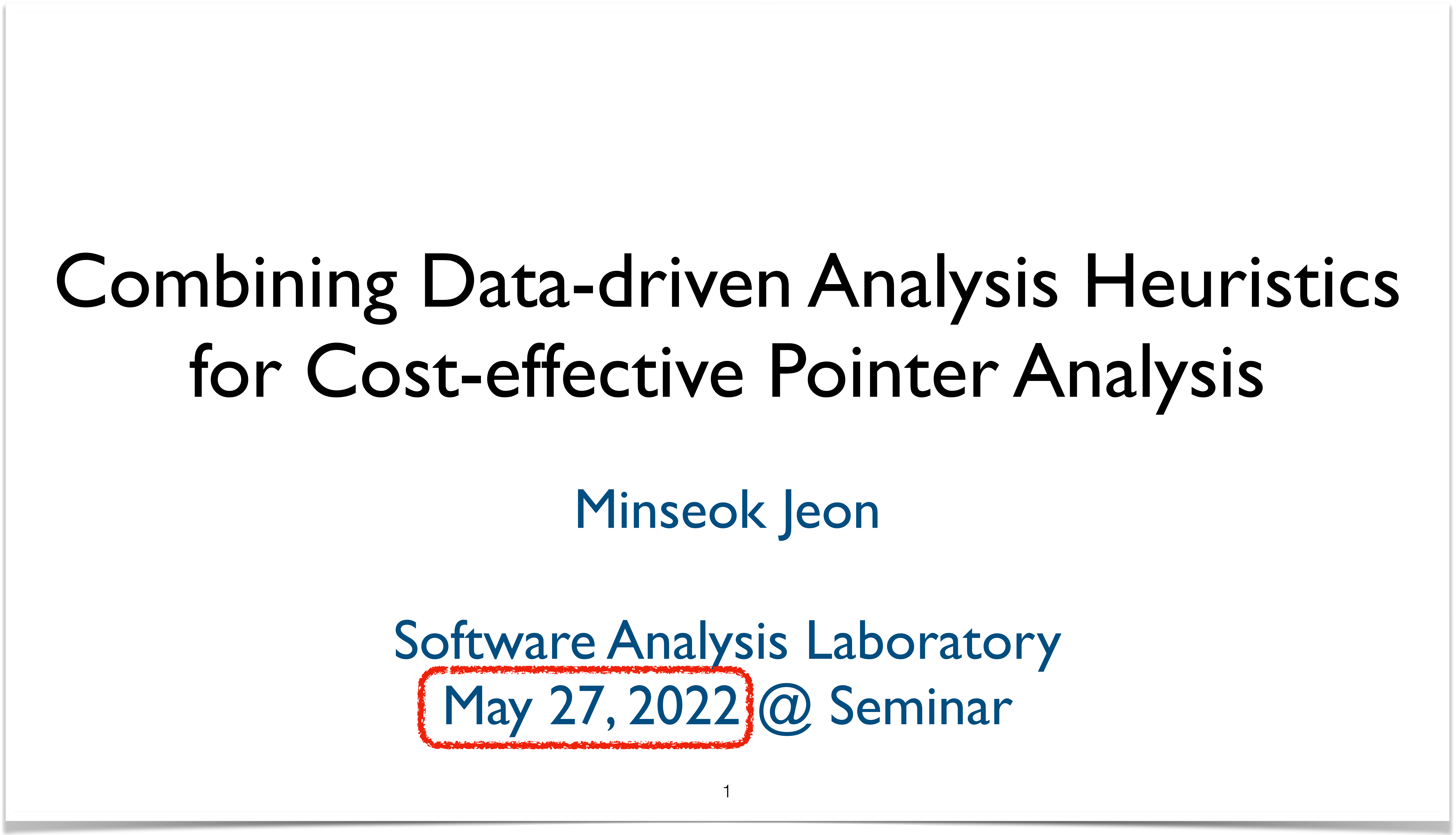


MinseokJGit typo

CFA_ICSE	title
CFA_ICSE_10p	title
CFA_Journal	CFA_Jou
CFA_OOPSLA21	spacing
CFA_Poster	poster
CFA_Simulation_Learning	title
CFA_Simulation_Minseok	title
ContextDepth	update
ContextJournal	minor
ContextPoster	title
ContextPoster_English	title
ContextTunneling	title
Framework	typo
GraphModel	minor
Graphick_Journal	update
HeapAbstraction	title
HeapTunneling	update
IST_Scalable	title
Infer_PTA	update
JavaScriptContext	evaluatio
Journal_Data_Driven	update
LearningLattice	minor
Learning_Framework	upate
OneCFA	mnor
POPL_2022	update
ResurrectionOfCFA	title
RuleSyn	minor
ScalableLearningAlgorithm	minor
TunnelingJournal	title
TunnelingPoster_English	title
TunnelingUserManual	title
VariableAbstraction	minor
pldi2020_CFA	title
.gitignore	remove paper.pdf

https://github.com/kupl/PointerAnalysisPaper

남의 생각에 너무 휘둘리지 말자



Combining Data-driven Analysis Heuristics for Cost-effective Pointer Analysis

Minseok Jeon

Software Analysis Laboratory
May 27, 2022 @ Seminar

1

Comment 1:
별로다. 다른 주제를 연구 하는게 좋아보인다.

Comment 2:
Incremental한 연구다.

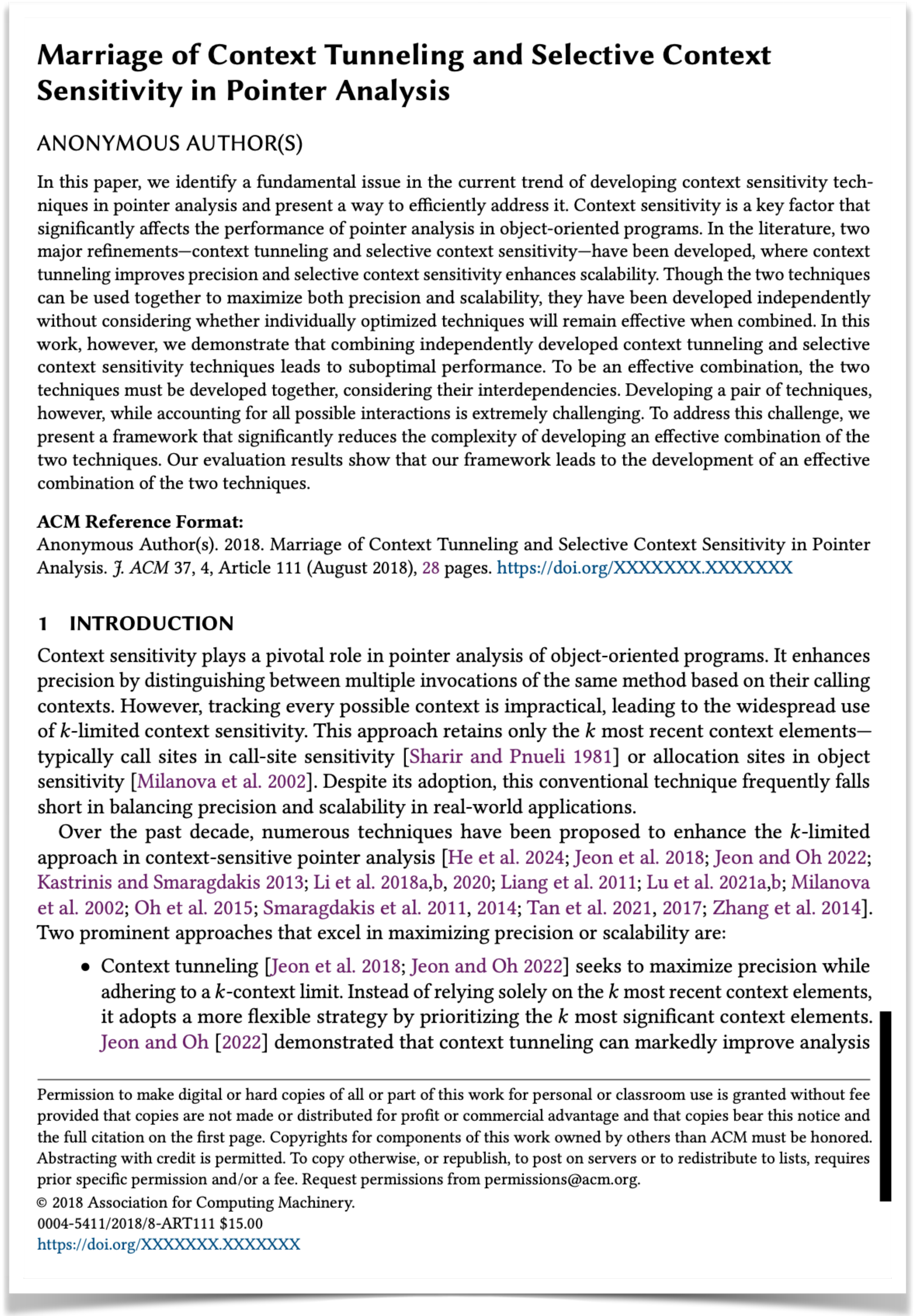
…

https://salbox.korea.ac.kr/drive/#file_id=726284236899792540

남의 생각에 너무 휘둘리지 말자

Your Submissions

#501 Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis **Submitted**



Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis

ANONYMOUS AUTHOR(S)

In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis and present a way to efficiently address it. Context sensitivity is a key factor that significantly affects the performance of pointer analysis in object-oriented programs. In the literature, two major refinements—context tunneling and selective context sensitivity—have been developed, where context tunneling improves precision and selective context sensitivity enhances scalability. Though the two techniques can be used together to maximize both precision and scalability, they have been developed independently without considering whether individually optimized techniques will remain effective when combined. In this work, however, we demonstrate that combining independently developed context tunneling and selective context sensitivity techniques leads to suboptimal performance. To be an effective combination, the two techniques must be developed together, considering their interdependencies. Developing a pair of techniques, however, while accounting for all possible interactions is extremely challenging. To address this challenge, we present a framework that significantly reduces the complexity of developing an effective combination of the two techniques. Our evaluation results show that our framework leads to the development of an effective combination of the two techniques.

ACM Reference Format:
Anonymous Author(s). 2018. Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis. *J. ACM* 37, 4, Article 111 (August 2018), 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Context sensitivity plays a pivotal role in pointer analysis of object-oriented programs. It enhances precision by distinguishing between multiple invocations of the same method based on their calling contexts. However, tracking every possible context is impractical, leading to the widespread use of k-limited context sensitivity. This approach retains only the k most recent context elements—typically call sites in call-site sensitivity [Sharir and Pnueli 1981] or allocation sites in object sensitivity [Milanova et al. 2002]. Despite its adoption, this conventional technique frequently falls short in balancing precision and scalability in real-world applications.

Over the past decade, numerous techniques have been proposed to enhance the k-limited approach in context-sensitive pointer analysis [He et al. 2024; Jeon et al. 2018; Jeon and Oh 2022; Kastrinis and Smaragdakis 2013; Li et al. 2018a,b, 2020; Liang et al. 2011; Lu et al. 2021a,b; Milanova et al. 2002; Oh et al. 2015; Smaragdakis et al. 2011, 2014; Tan et al. 2021, 2017; Zhang et al. 2014]. Two prominent approaches that excel in maximizing precision or scalability are:

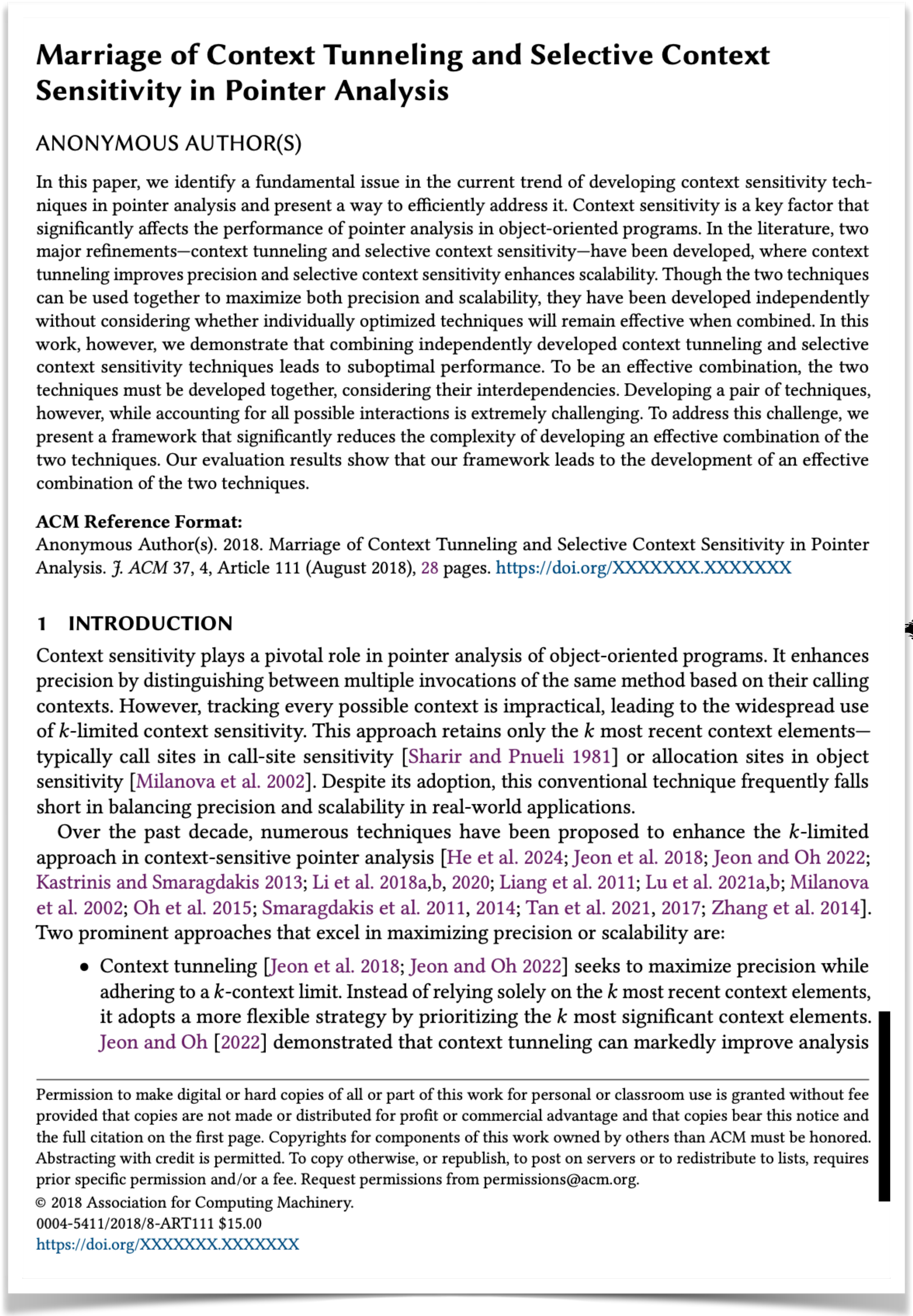
- Context tunneling [Jeon et al. 2018; Jeon and Oh 2022] seeks to maximize precision while adhering to a k-context limit. Instead of relying solely on the k most recent context elements, it adopts a more flexible strategy by prioritizing the k most significant context elements. Jeon and Oh [2022] demonstrated that context tunneling can markedly improve analysis

- In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis
- We demonstrate that combining independently developed context tunneling and selective context sensitivity techniques leads to suboptimal performance.

남의 생각에 너무 휘둘리지 말자

Your Submissions

#501 Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis **Submitted**



Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis

ANONYMOUS AUTHOR(S)

In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis and present a way to efficiently address it. Context sensitivity is a key factor that significantly affects the performance of pointer analysis in object-oriented programs. In the literature, two major refinements—context tunneling and selective context sensitivity—have been developed, where context tunneling improves precision and selective context sensitivity enhances scalability. Though the two techniques can be used together to maximize both precision and scalability, they have been developed independently without considering whether individually optimized techniques will remain effective when combined. In this work, however, we demonstrate that combining independently developed context tunneling and selective context sensitivity techniques leads to suboptimal performance. To be an effective combination, the two techniques must be developed together, considering their interdependencies. Developing a pair of techniques, however, while accounting for all possible interactions is extremely challenging. To address this challenge, we present a framework that significantly reduces the complexity of developing an effective combination of the two techniques. Our evaluation results show that our framework leads to the development of an effective combination of the two techniques.

ACM Reference Format:
Anonymous Author(s). 2018. Marriage of Context Tunneling and Selective Context Sensitivity in Pointer Analysis. *J. ACM* 37, 4, Article 111 (August 2018), 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Context sensitivity plays a pivotal role in pointer analysis of object-oriented programs. It enhances precision by distinguishing between multiple invocations of the same method based on their calling contexts. However, tracking every possible context is impractical, leading to the widespread use of k-limited context sensitivity. This approach retains only the k most recent context elements—typically call sites in call-site sensitivity [Sharir and Pnueli 1981] or allocation sites in object sensitivity [Milanova et al. 2002]. Despite its adoption, this conventional technique frequently falls short in balancing precision and scalability in real-world applications.

Over the past decade, numerous techniques have been proposed to enhance the k-limited approach in context-sensitive pointer analysis [He et al. 2024; Jeon et al. 2018; Jeon and Oh 2022; Kastrinis and Smaragdakis 2013; Li et al. 2018a,b, 2020; Liang et al. 2011; Lu et al. 2021a,b; Milanova et al. 2002; Oh et al. 2015; Smaragdakis et al. 2011, 2014; Tan et al. 2021, 2017; Zhang et al. 2014]. Two prominent approaches that excel in maximizing precision or scalability are:

- Context tunneling [Jeon et al. 2018; Jeon and Oh 2022] seeks to maximize precision while adhering to a k-context limit. Instead of relying solely on the k most recent context elements, it adopts a more flexible strategy by prioritizing the k most significant context elements. Jeon and Oh [2022] demonstrated that context tunneling can markedly improve analysis

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.
0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

- In this paper, we identify a fundamental issue in the current trend of developing context sensitivity techniques in pointer analysis

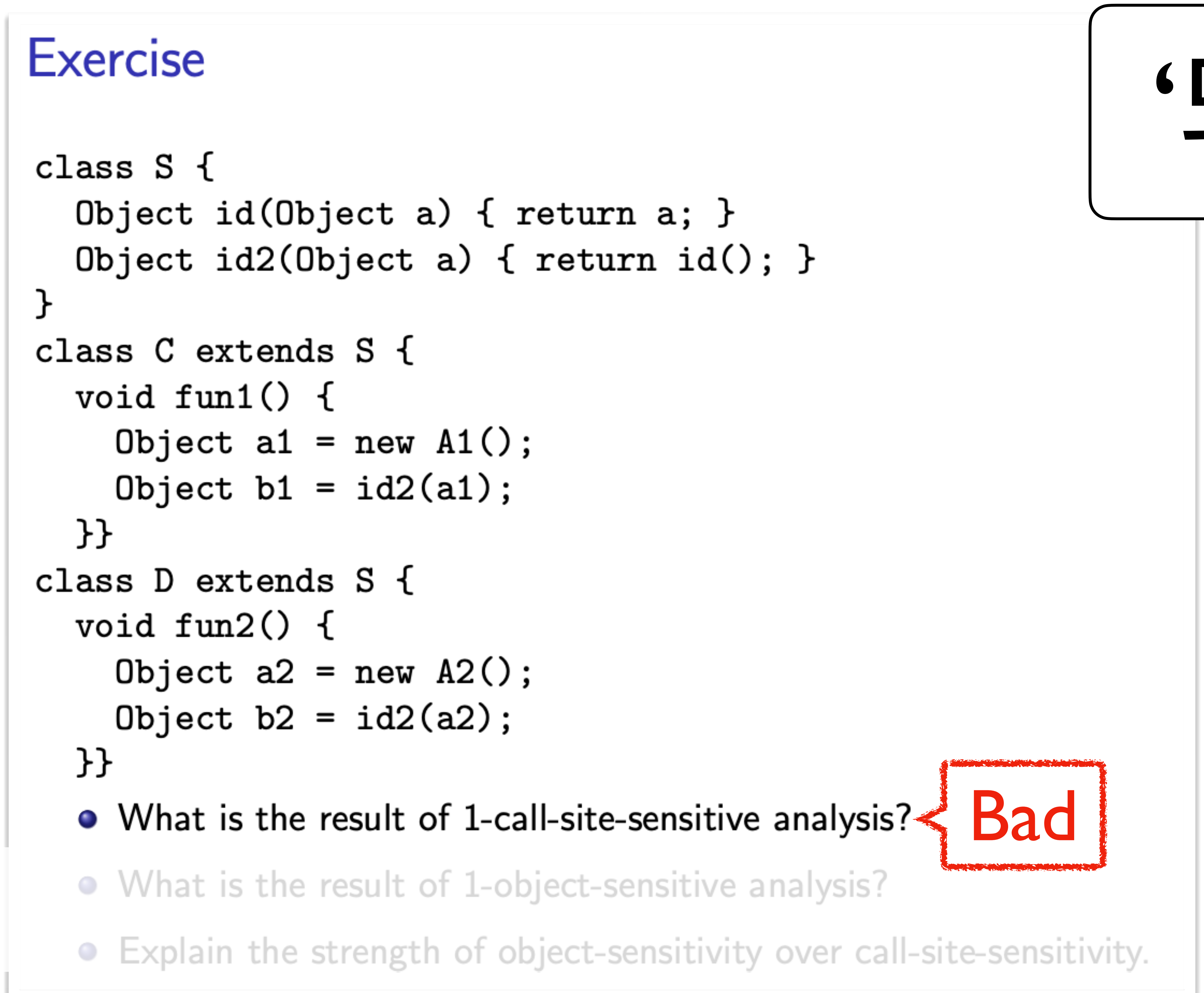
A review comment in the previous submission to OOPSLA 2025

"The paper makes a strong case for reconsidering choice of baseline analyses for future investigations."

- Reviewer B [Accept]

결론

1. 좋은 문제는 작고 개인적인 질문에서 발견된다.

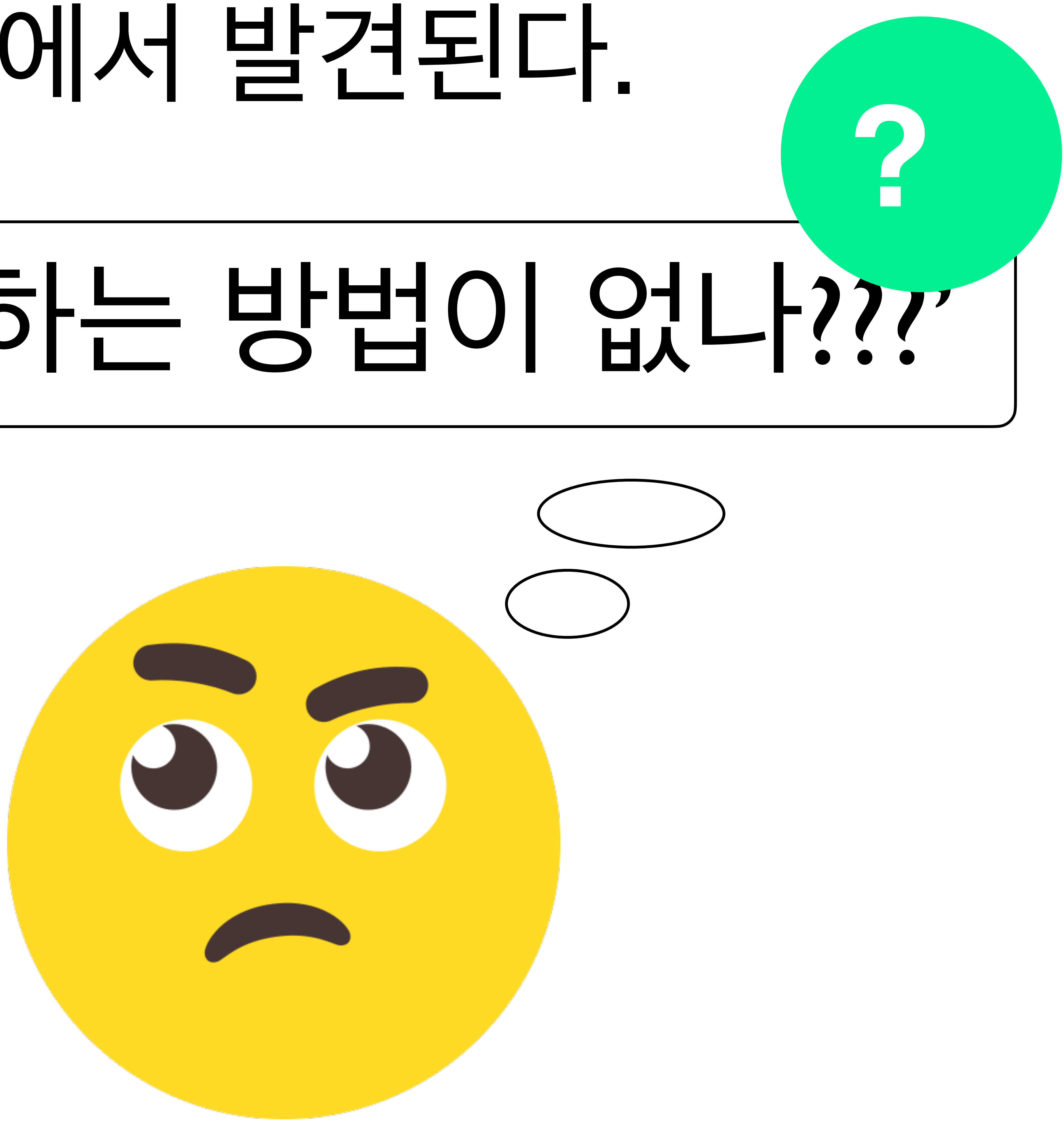


Exercise

```
class S {
  Object id(Object a) { return a; }
  Object id2(Object a) { return id(); }
}
class C extends S {
  void fun1() {
    Object a1 = new A1();
    Object b1 = id2(a1);
  }}
class D extends S {
  void fun2() {
    Object a2 = new A2();
    Object b2 = id2(a2);
  }}
```

- What is the result of 1-call-site-sensitive analysis? Bad
- What is the result of 1-object-sensitive analysis?
- Explain the strength of object-sensitivity over call-site-sensitivity.

'되게 하는 방법이 없나???'

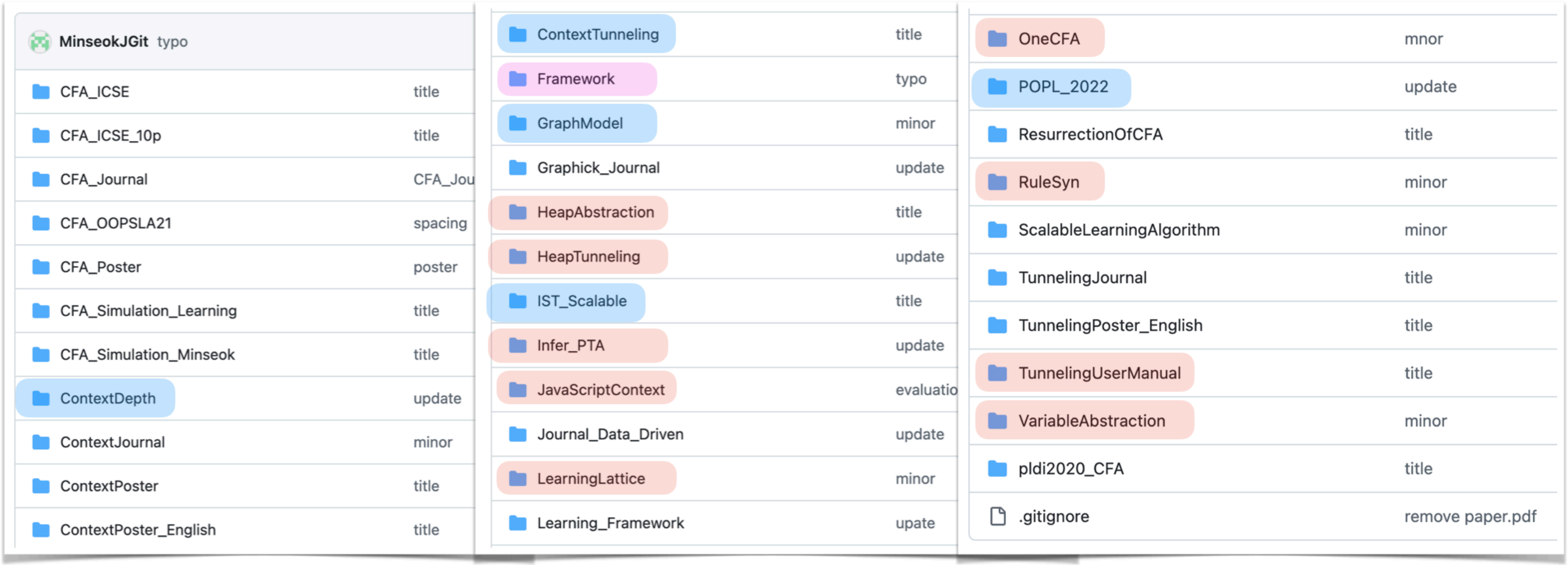


결론

2. (실패를 동반한) 많은 시도를 해야 한다.

🟦 : 성공한 연구 (논문 출판)

🟥 : 실패한 연구 (중도 포기)



MinseokJGit typo

	CFA_ICSE	title
	CFA_ICSE_10p	title
	CFA_Journal	CFA_Jou
	CFA_OOPSLA21	spacing
	CFA_Poster	poster
	CFA_Simulation_Learning	title
	CFA_Simulation_Minseok	title
🟦	ContextDepth	update
	ContextJournal	minor
	ContextPoster	title
	ContextPoster_English	title
🟦	ContextTunneling	title
🟪	Framework	typo
🟦	GraphModel	minor
	Graphick_Journal	update
🟥	HeapAbstraction	title
🟥	HeapTunneling	update
🟦	IST_Scalable	title
🟥	Infer_PTA	update
🟥	JavaScriptContext	evaluatio
	Journal_Data_Driven	update
🟥	LearningLattice	minor
	Learning_Framework	upate
🟥	OneCFA	mnor
🟦	POPL_2022	update
	ResurrectionOfCFA	title
🟥	RuleSyn	minor
	ScalableLearningAlgorithm	minor
	TunnelingJournal	title
	TunnelingPoster_English	title
🟥	TunnelingUserManual	title
🟥	VariableAbstraction	minor
	pldi2020_CFA	title
	.gitignore	remove paper.pdf

결론

3. 문제를 찾을 때 남의 생각에 너무 휘둘리지는 말자.

Comment 1:
별로다. 다른 주제를 연구 하는게 좋아보인다.

Comment 2:
Incremental한 연구다.

…

- 첫 발표 -

“The paper makes a strong case for reconsidering choice of baseline analyses for future investigations.”

- Reviewer B [Accept]

- 논문&완성 제출 후 -

결론

1. 좋은 문제는 작고 개인적인 질문에서 발견된다.

2. (실패를 동반한) 많은 시도를 해야 한다.

3. 문제를 찾을 때 남의 생각에 너무 휘둘리지는 말자.