

# 개발자와 상호작용하는 쌍방향 정적 분석 시스템

허기홍

전산학부 / 정보보호대학원



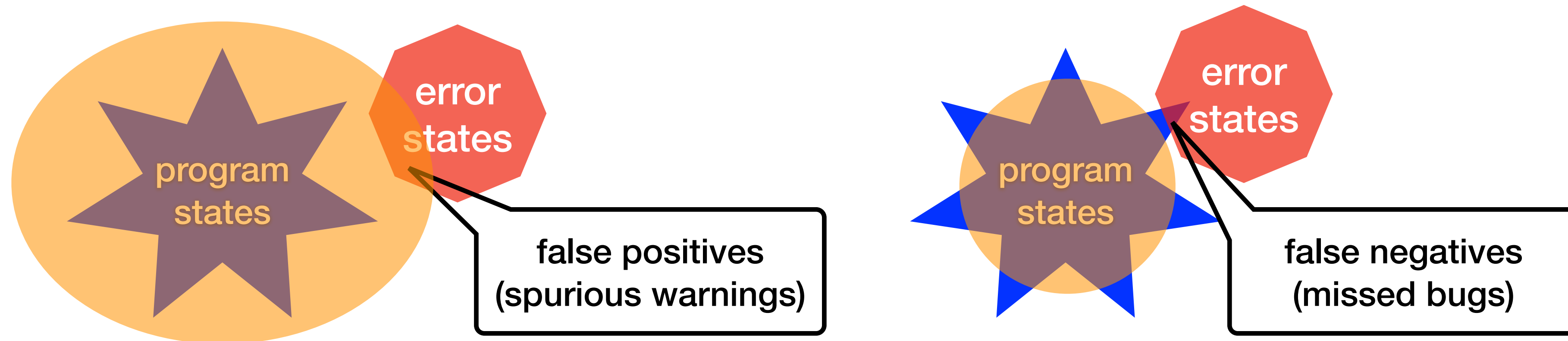
# 소프트웨어 정적 분석

- SW 의 동작을 자동으로 예측하는 일반적이고 체계적인 방법
  - 예측: 실행하기 전에 미리 (정적)
  - 자동: 소프트웨어를 분석하는 소프트웨어 (“분석기”)
  - 일반적: 언어와 성질에 국한되지 않음
  - 체계적 : 요약 해석 (abstract interpretation) 이라는 이론에 기반
- 응용: 오류 검출, 검증, 보안, 코드 유지보수, 코드 최적화 등



“SW MRI”

# 여전한 문제



“... can be difficult to do without introducing large numbers of **false positives**, or scaling **performance** exponentially poorly. In this case, **balancing** these ... caused us to **miss the defect**.”

– *On Detecting Heartbleed with Static Analysis, (Coverity, 2014)*

# 정적 분석기 길들이기

```
kihong@elvis01 ~$ clang -cc1 -analyzer-config-help | grep "default"
add-pop-up-notes (bool) Whether pop-up notes should be added to the final output. (default: true)
aggressive-binary-operation-simplification (bool) Whether SValBuilder should rearrange comparisons and additive operations of symbolic expressions to the maximum value of that type. A + n <OP> B + m becomes A - B <OP> m - n, where A and B symbolic, n and m are integers. <OP> is any of '=', '<!--
avoid-suppressing-null-argument-paths (bool) Whether a bug report should not be suppressed if its path includes a call with a null argument, even if the argument is a constant. (default: true)
c++-allocator-inlining (bool) Whether or not allocator call may be considered for inlining. (default: true)
c++-container-inlining (bool) Whether or not methods of C++ container objects may be considered for inlining. (default: false)
c++-inlining (string) Controls which C++ member functions will be considered for inlining. Value: "constructors", "destructors"
c++-shared_ptr-inlining (bool) Whether or not the destructor of C++ 'shared_ptr' may be considered for inlining. This covers std::shared_ptr
c++-stdlib-inlining (bool) Whether or not C++ standard library functions may be considered for inlining. (default: true)
c++-temp-dtor-inlining (bool) Whether C++ temporary destructors should be inlined during analysis. If temporary destructors are disabled, this option has no effect. (default: true)
c++-template-inlining (bool) Whether or not templated functions may be considered for inlining. (default: true)
cfg-conditional-static-initializers (bool) Whether 'static' initializers should be in conditional logic in the CFG. (default: true)
cfg-implicit-dtors (bool) Whether or not implicit destructors for C++ objects should be included in the CFG. (default: true)
cfg-lifetime (bool) Whether or not end-of-lifetime information should be included in the CFG. (default: false)
cfg-loopexit (bool) Whether or not the end of the loop information should be included in the CFG. (default: false)
cfg-rich-constructors
cfg-scopes
cfg-temporary-dtors
crosscheck-with-z3
ctu-dir
ctu-import-threshold
ctu-index-name
display-ctu-progress
eagerly-assume
downside is that it eagerly
elide-constructors
expand-macros
experimental-enable-naive
exploration_strategy
faux-bodies
graph-trim-interval
inline-lambdas
ipa
ipa-always-inline-size
max-inlinable-size
max-nodes
max-symbol-complexity
max-times-inline-large
min-cfg-size-treat-functions
mode
model-path
notes-as-events
objc-inlining
prune-paths
region-store-small-struct
report-in-main-source-file
serialize-stats
stable-report-filename
suppress-c++-stdlib
suppress-inlined-defensive
suppress-null-return-path
track-conditions
track-conditions-debug
unroll-loops
widen-loops
```

## [cfe-dev] Handling of loops in the Clang Static Analyzer

Sean Eveson via cfe-dev [cfe-dev at lists.lvm.org](mailto:cfe-dev@lists.lvm.org)  
Mon Feb 27 03:18:36 PST 2017

- Previous message: [\[cfe-dev\] Handling of loops in the Clang Static Analyzer](#)
- Next message: [\[cfe-dev\] Handling of loops in the Clang Static Analyzer](#)
- Messages sorted by: [\[ date \]](#) [\[ thread \]](#) [\[ subject \]](#) [\[ author \]](#)

Hi Venugopal,

> Sean Eveson (cc'd) did some initial work on loop widening to mitigate this problem.

I started to work on this, but have unfortunately not had time to take the next steps. There is a mode which does 'loop widening' which is off by

## FAQ and How to Deal with Common False Positives

1. [How do I tell the analyzer that I do not want the bug being reported here since my custom error handler will safely end the execution before the bug is reached?](#)
2. [The analyzer reports a null dereference, but I know that the pointer is never null. How can I tell the analyzer that a pointer can never be null?](#)
3. [How do I tell the static analyzer that I don't care about a specific dead store?](#)
4. [How do I tell the static analyzer that I don't care about a specific unused instance variable in Objective C?](#)
5. [How do I tell the static analyzer that I don't care about a specific unlocalized string?](#)
6. [How do I tell the analyzer that my instance variable does not need to be released in -dealloc under Manual Retain/Release?](#)
7. [How do I decide whether a method's return type should be \\_Nullable or \\_Nonnull?](#)
8. [How do I tell the analyzer that I am intentionally violating nullability?](#)
9. [The analyzer assumes that a loop body is never entered. How can I tell it that the loop body will be entered at least once?](#)
10. [How can I suppress a specific analyzer warning?](#)
11. [How can I selectively exclude code the analyzer examines?](#)

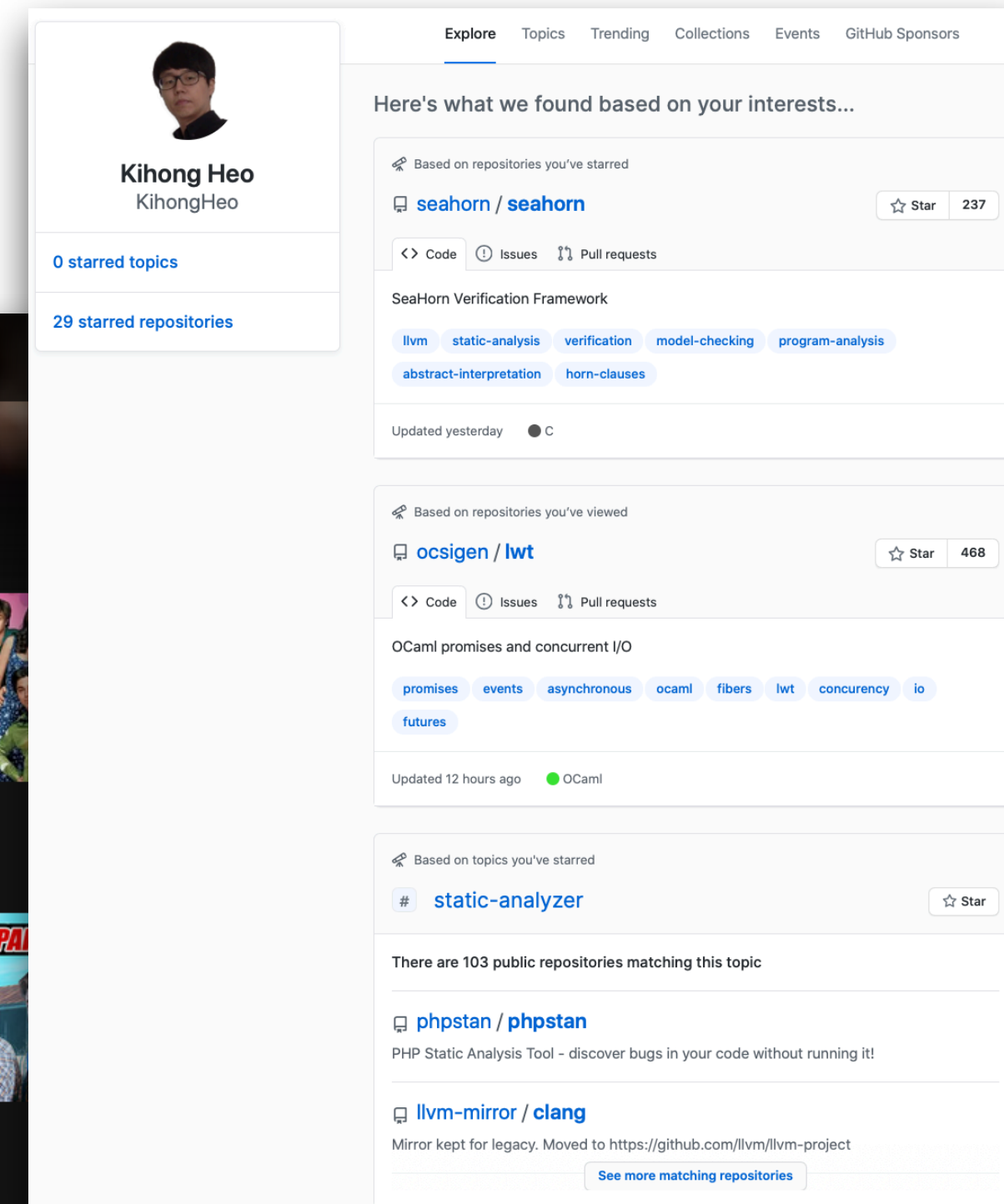
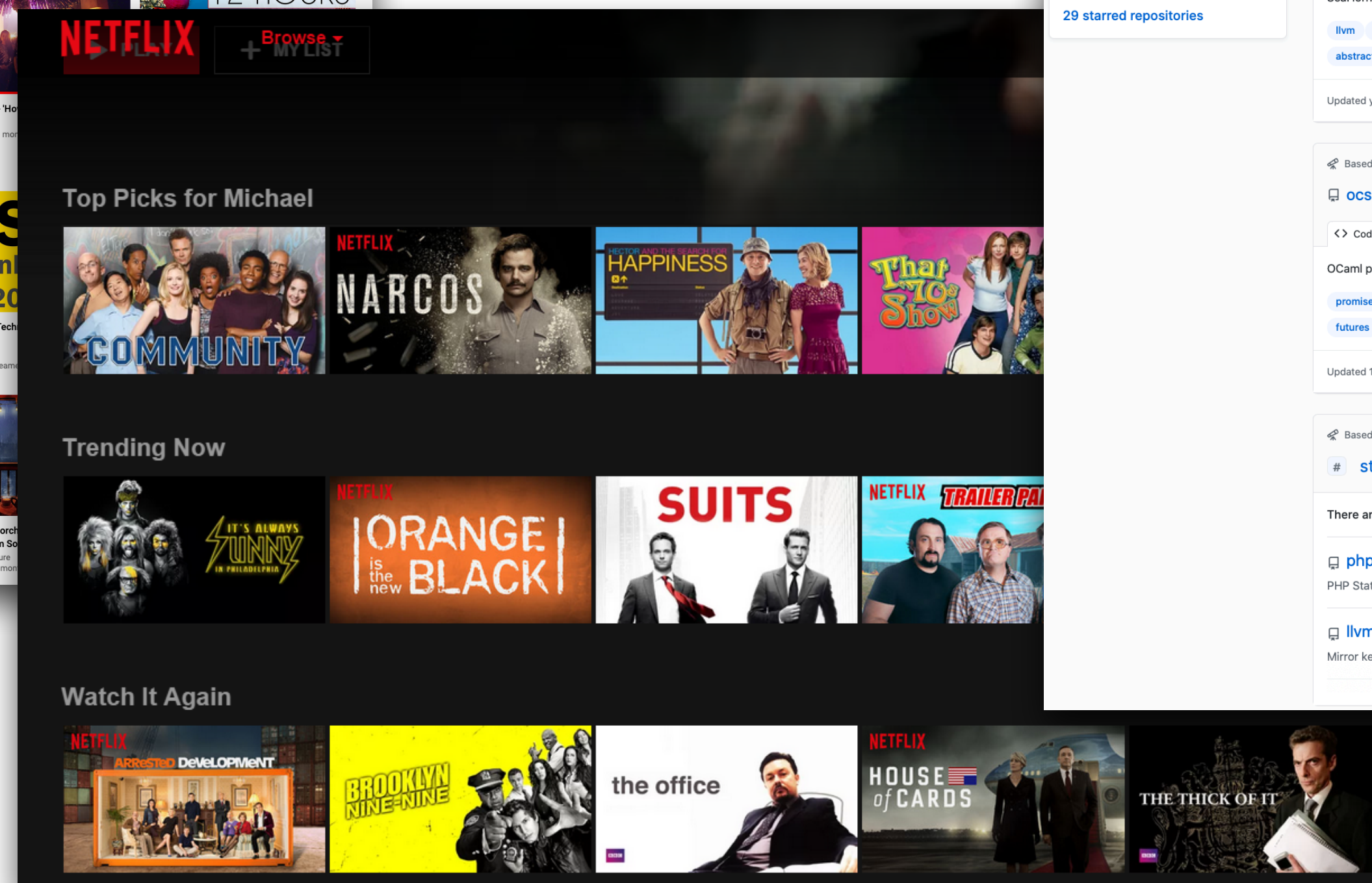
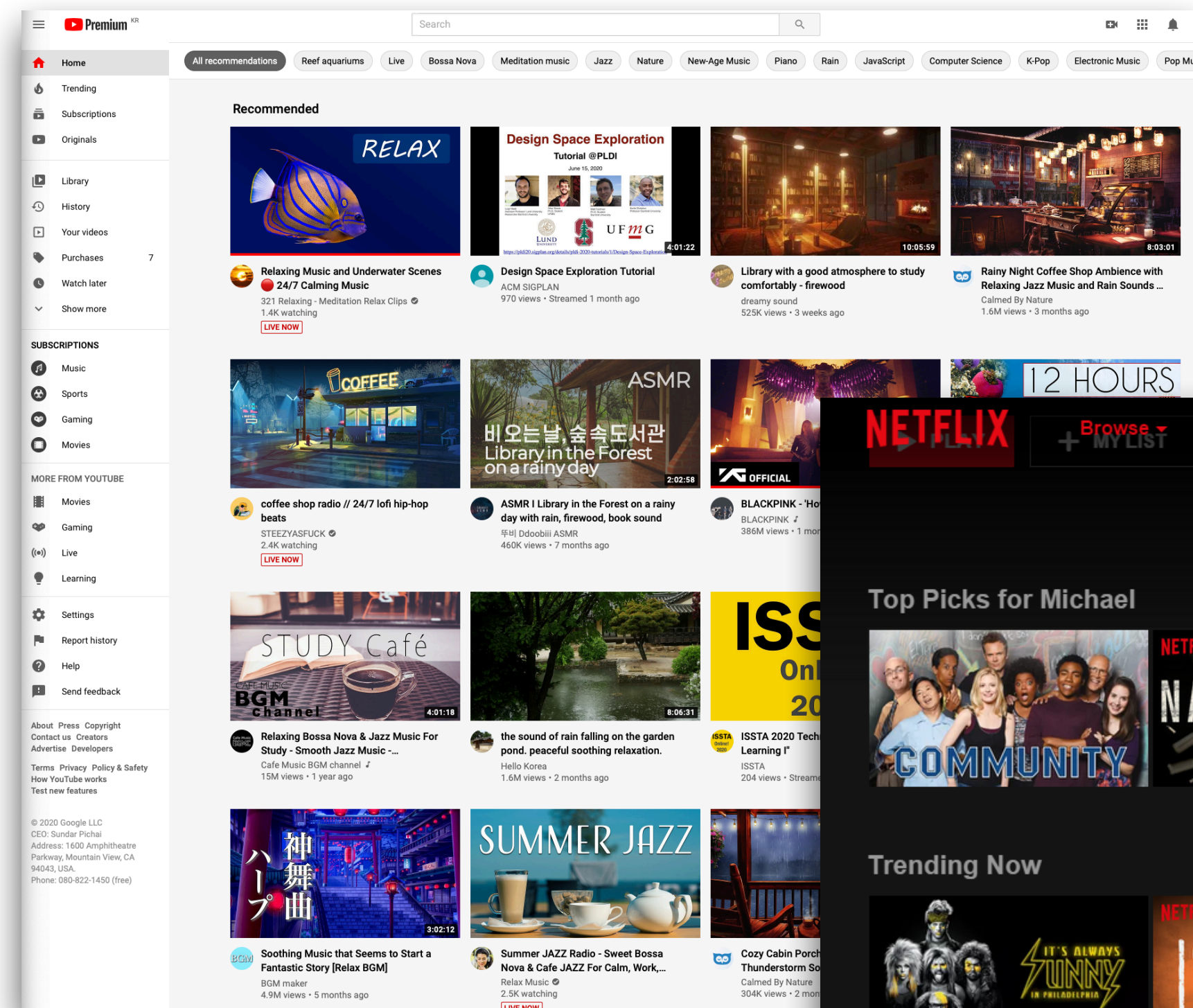
# 세상에 나쁜 정적 분석은 없다

정적 분석 훈련사

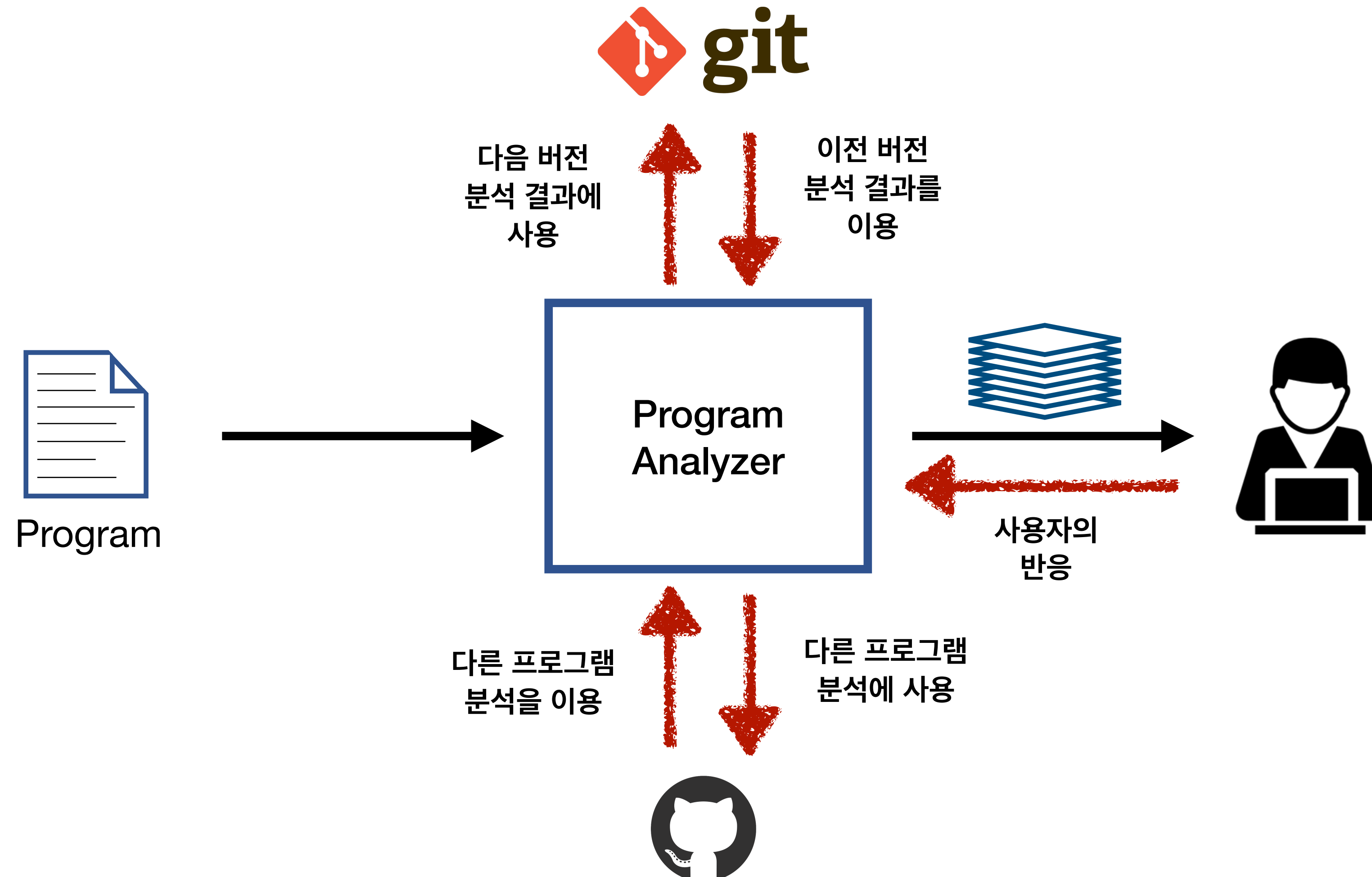
의뢰인 개발자



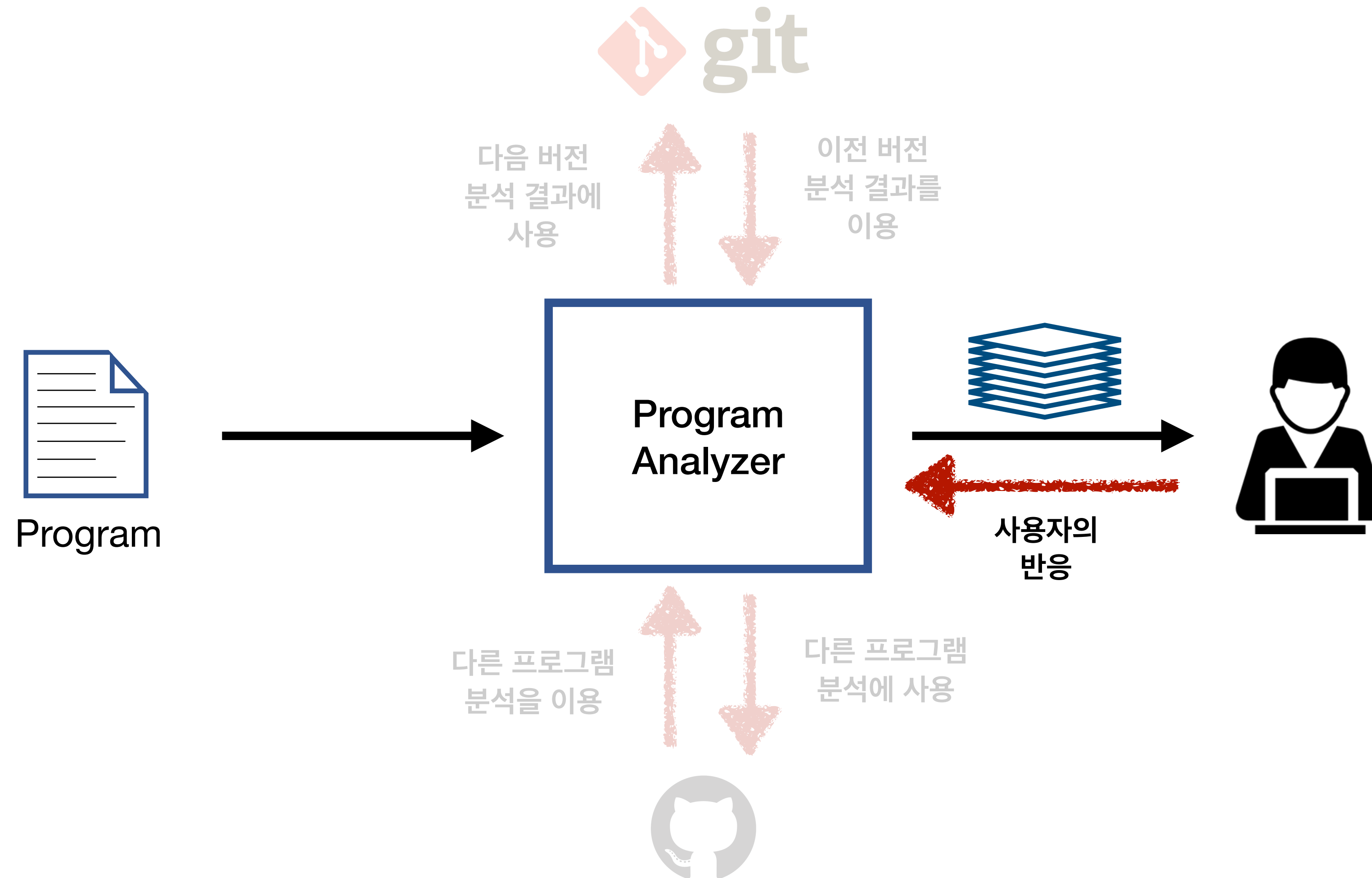
# 상호작용 시스템 (다른 분야)



# 정적 분석의 상호 작용



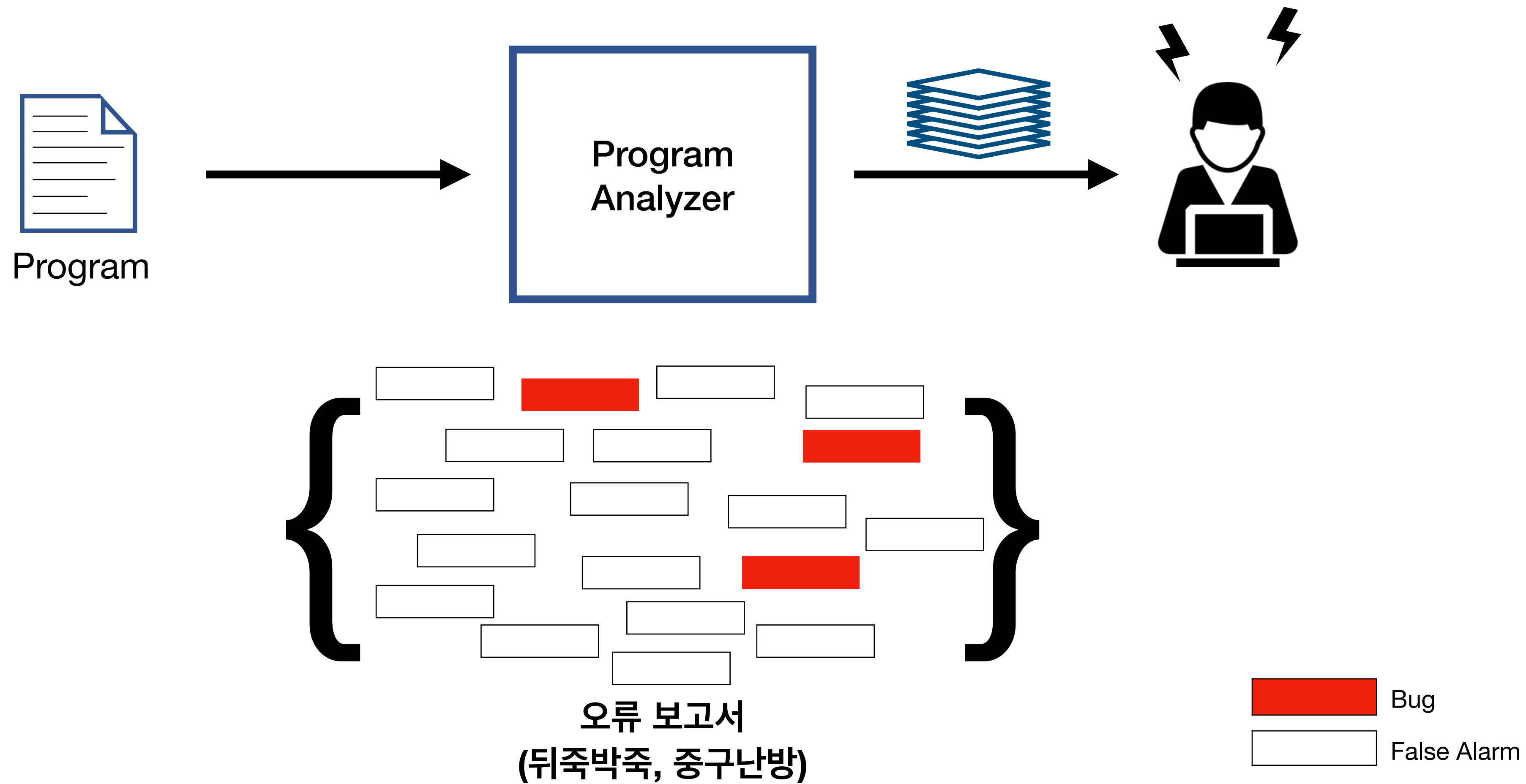
# 정적 분석의 상호 작용



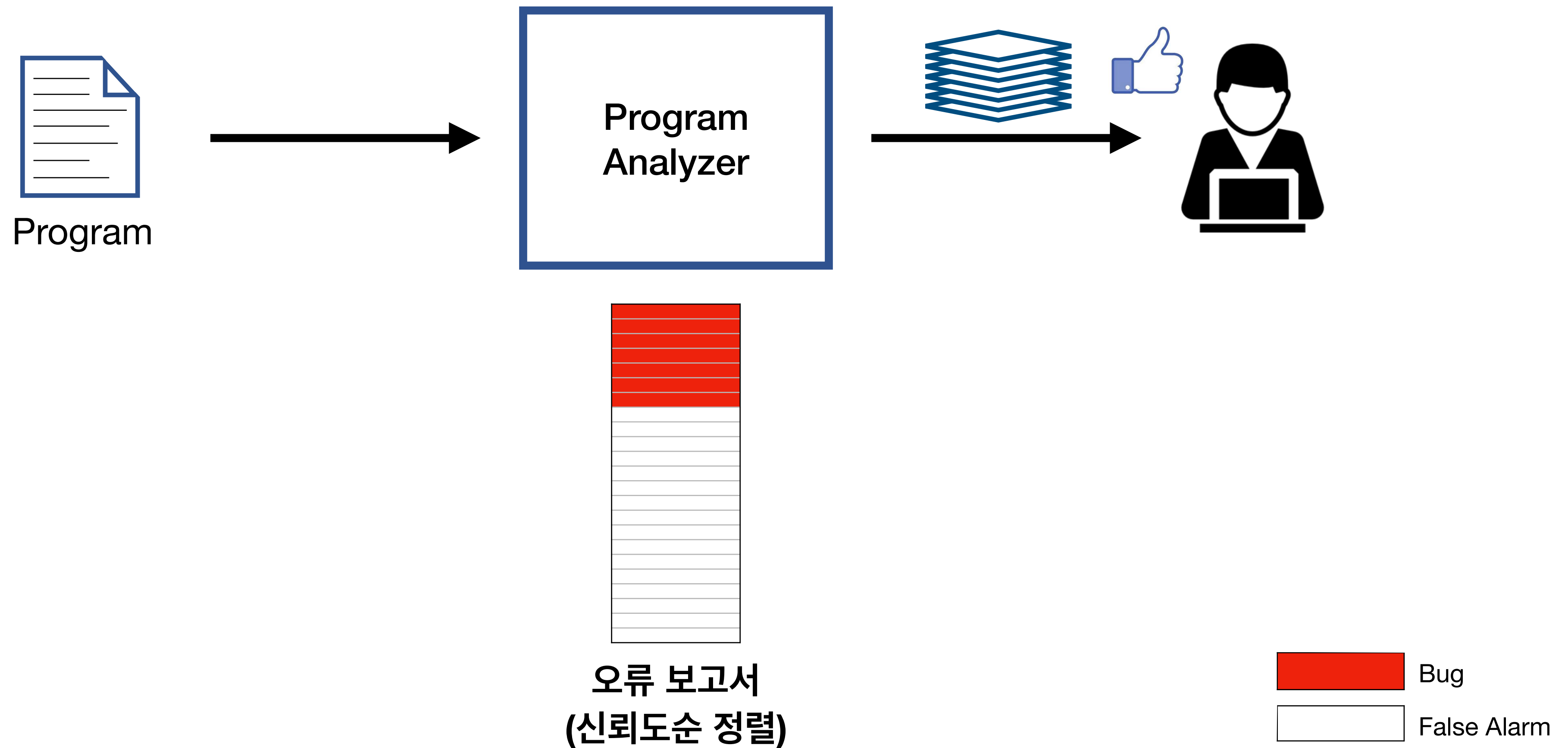


**Bingo: 사용자와 상호작용하는 정적 분석**  
**[PLDI'18]**

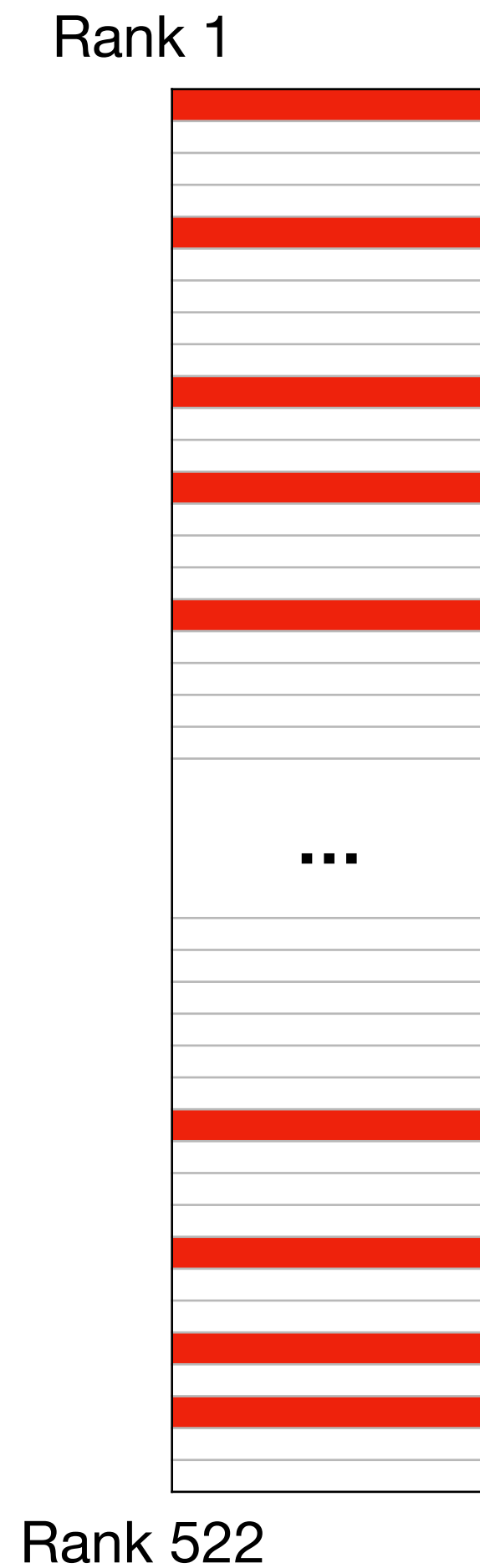
# 고질적인 문제점



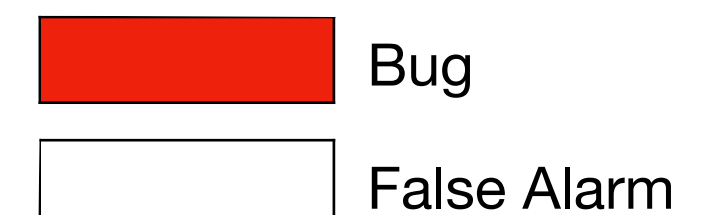
# 목표



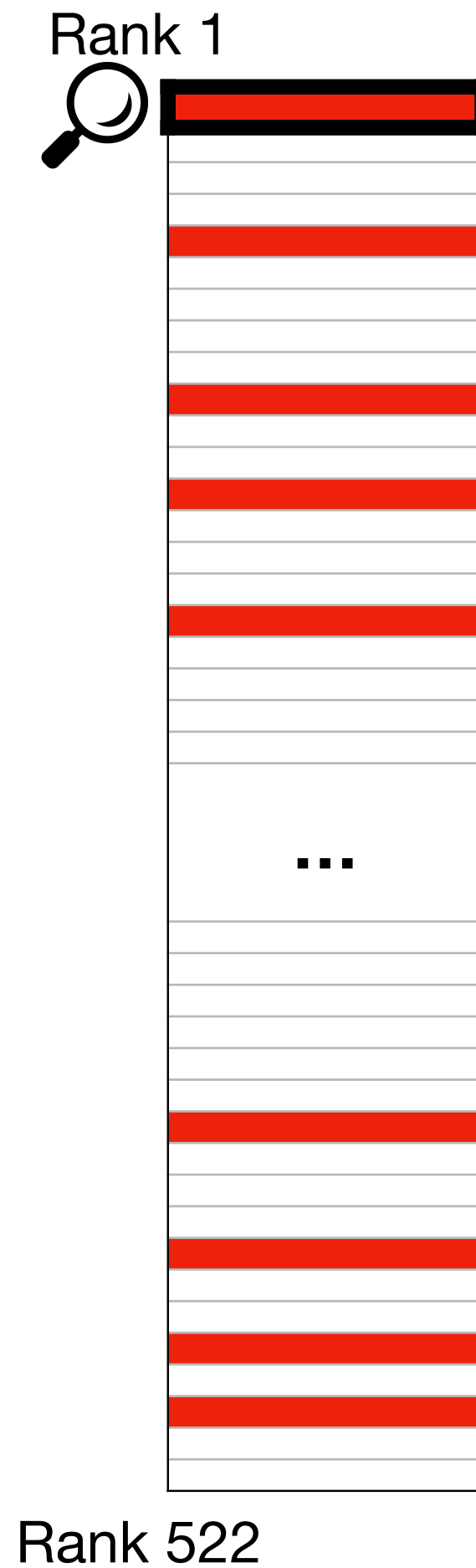
# 반응형 오류 보고



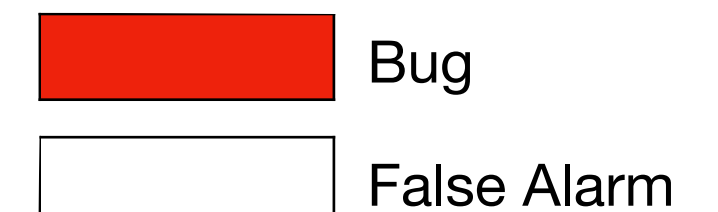
**152KLOC**  
**75 Datarace Bugs**  
**522 Total Alarms**



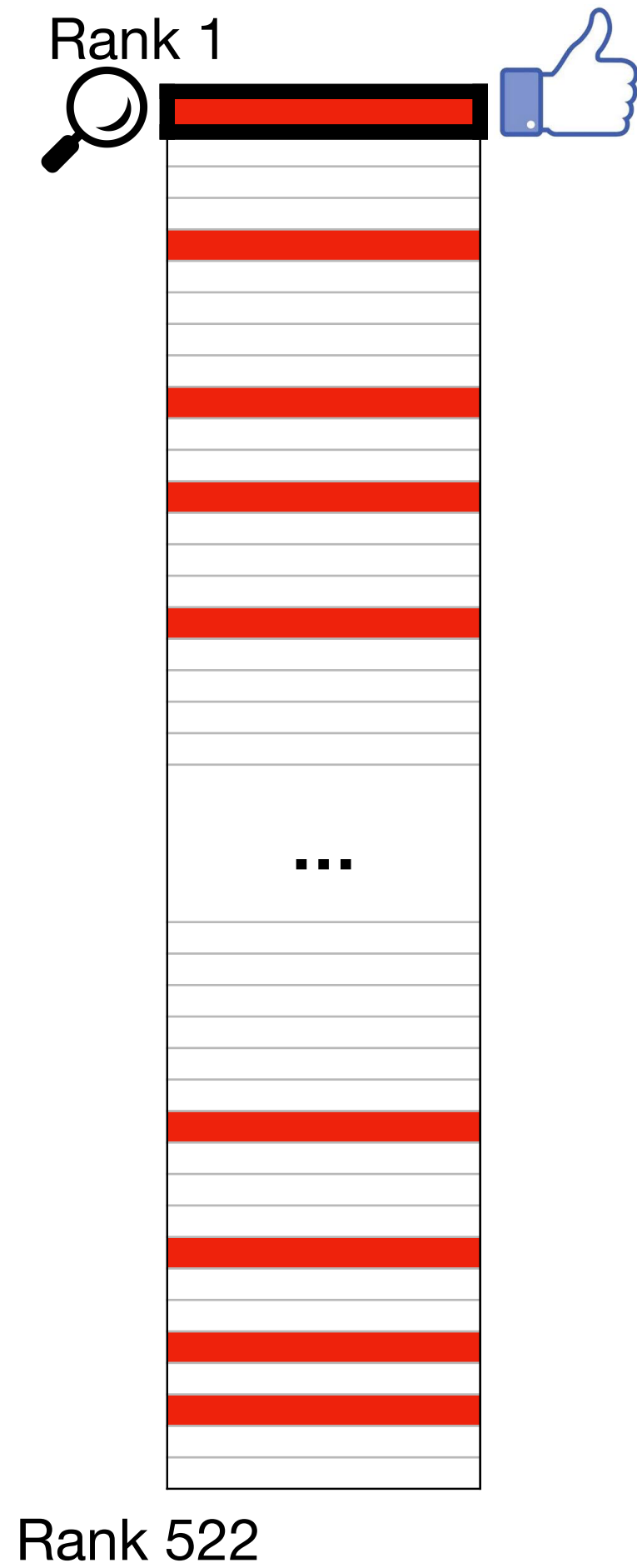
# 반응형 오류 보고



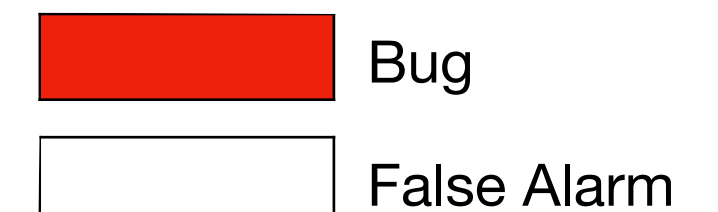
**152KLOC**  
**75 Datarace Bugs**  
**522 Total Alarms**



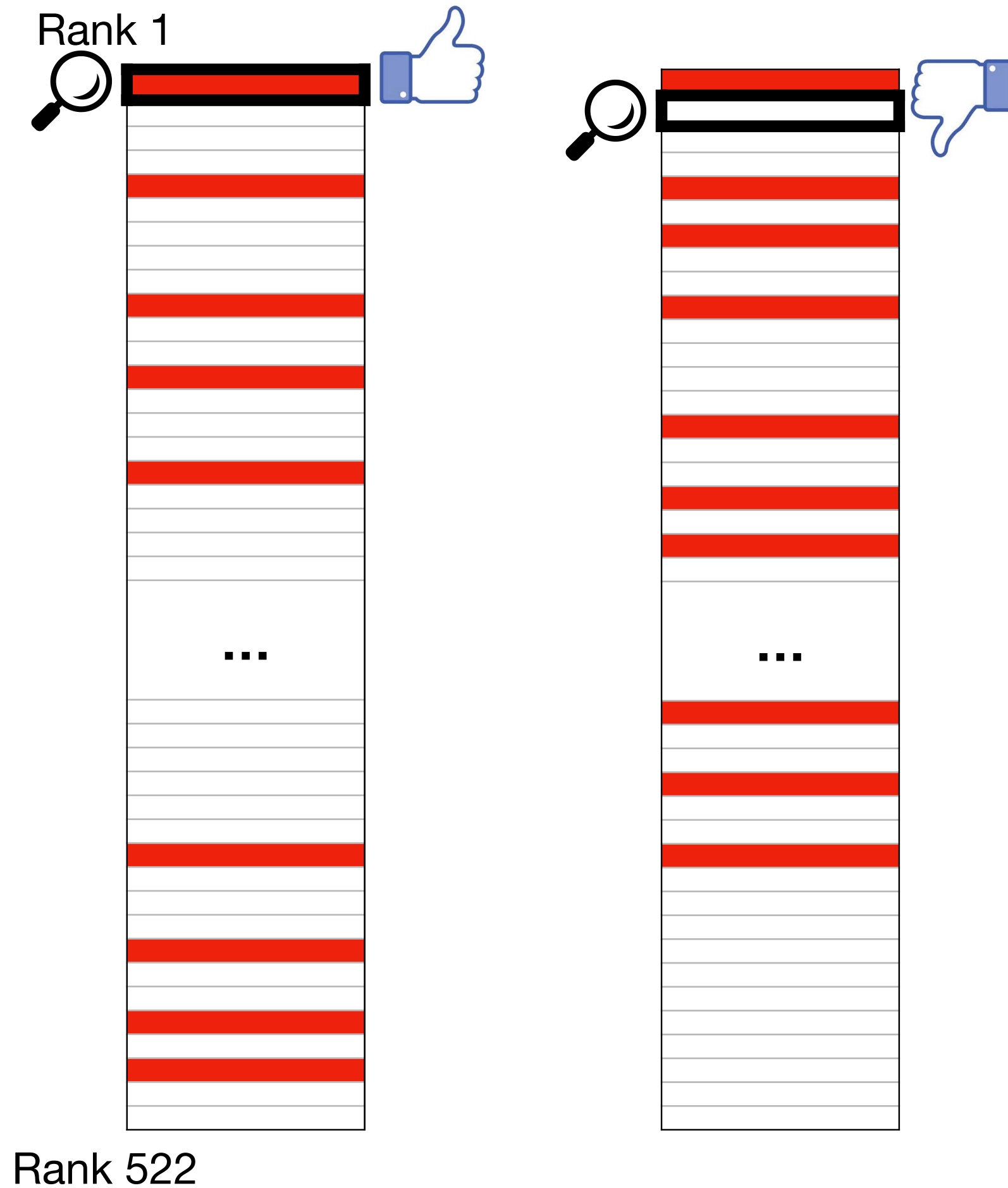
# 반응형 오류 보고



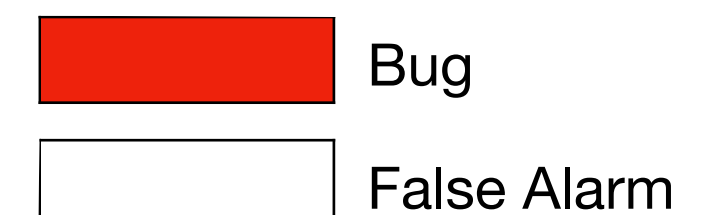
**152KLOC**  
**75 Datarace Bugs**  
**522 Total Alarms**



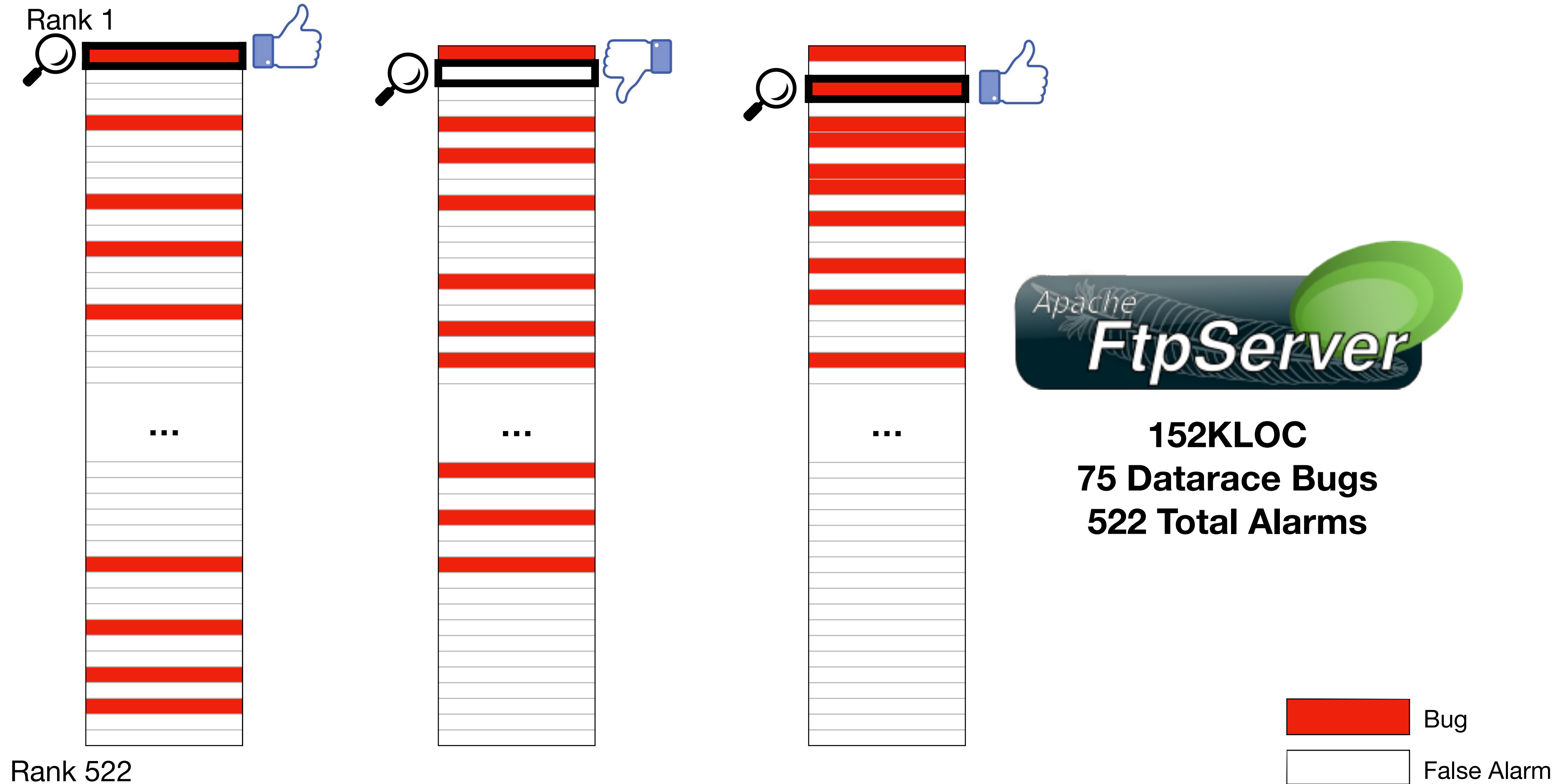
# 반응형 오류 보고



**152KLOC**  
**75 Datarace Bugs**  
**522 Total Alarms**

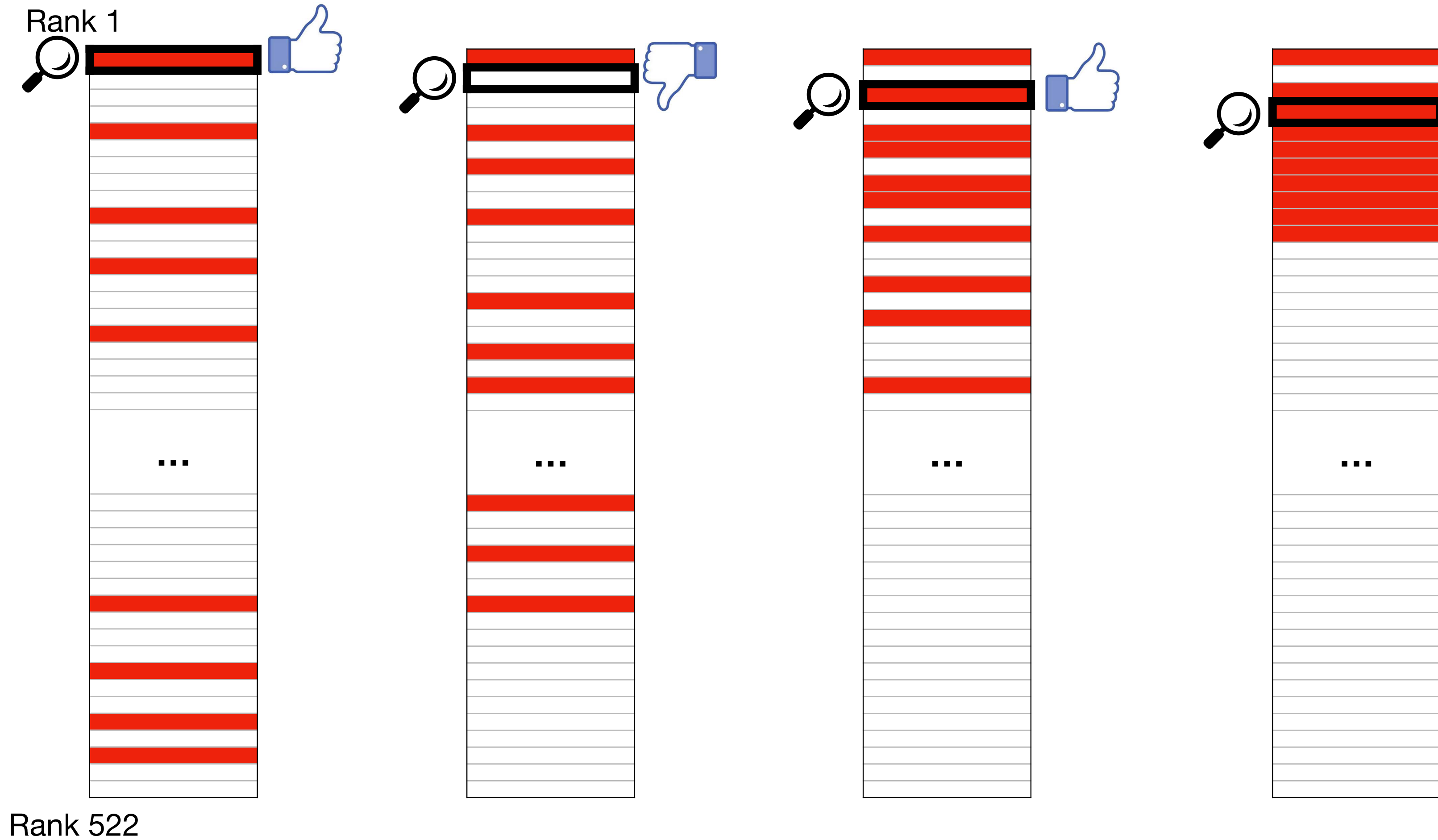


# 반응형 오류 보고





# 반응형 오류 보고

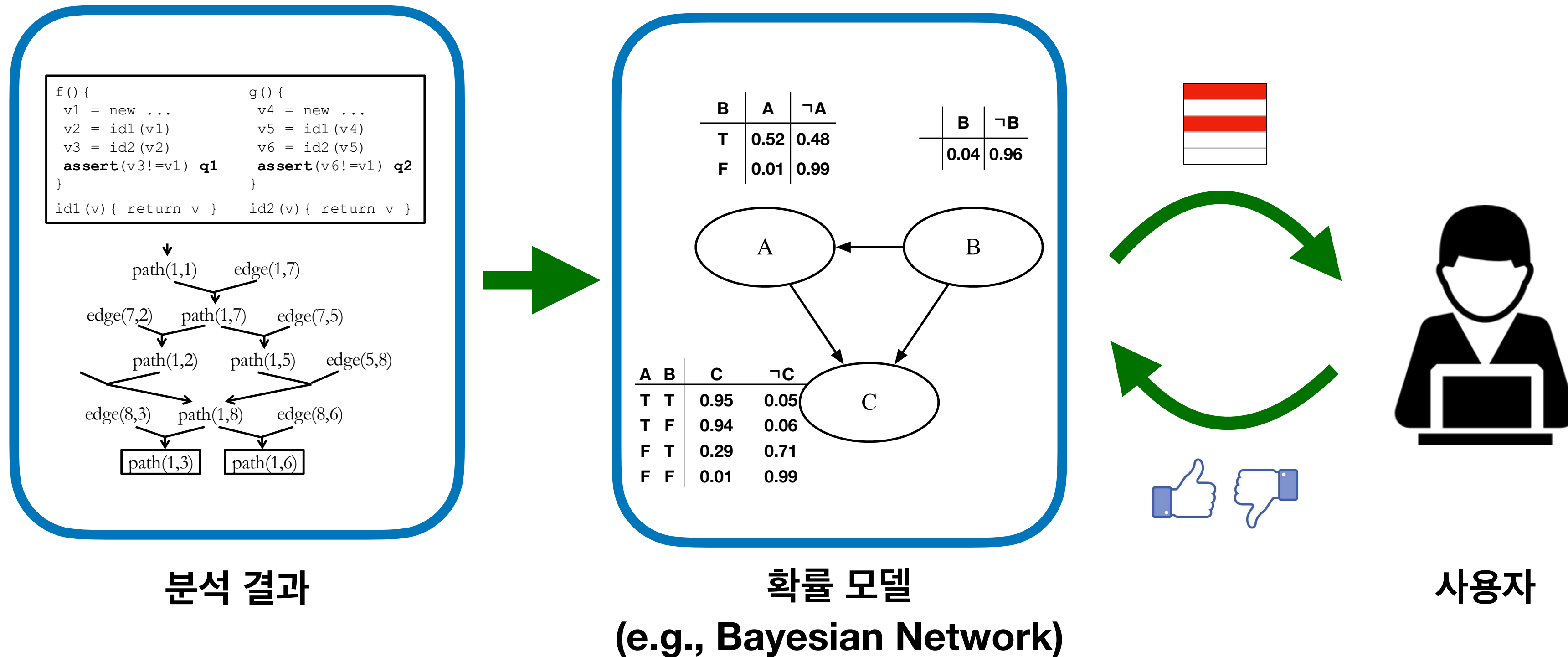


# 반응형 오류 보고





# 핵심 기술

## 인간-분석 상호 작용 + 베이지안 추론 (Bayesian inference)



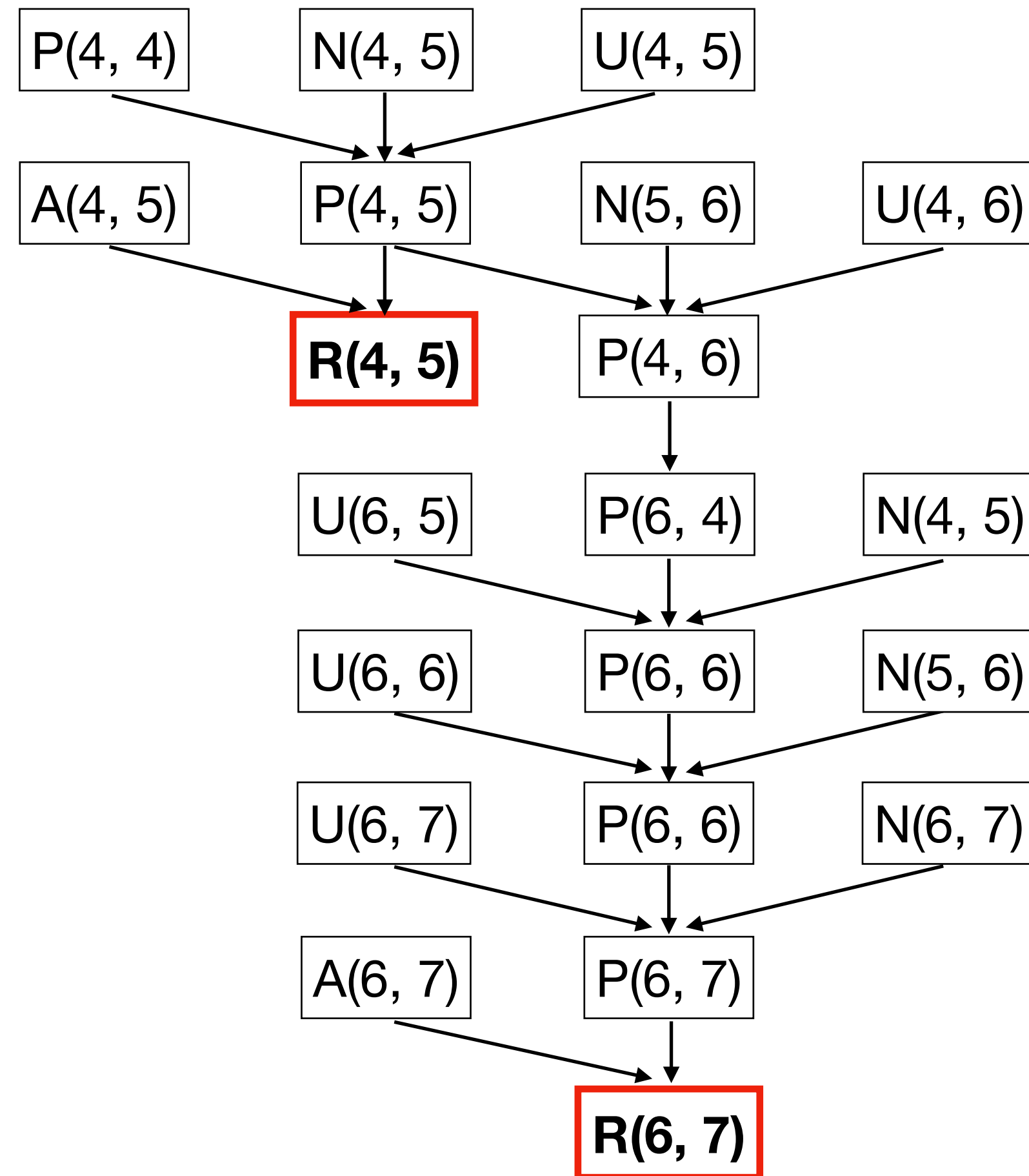
# 순위 기반 오류 보고서

```
public class RequestHandler {  
    private FtpRequest request;  
  
    public FtpRequest getRequest() {  
        return request; //L0  
    }  
  
    public void close() {  
        synchronized (this) { //L1  
            if (isClosed) return; //L2  
            isClosed = true; //L3  
        }  
        controlSocket.close(); //L4  
        controlSocket = null; //L5  
        request.clear(); //L6  
        request = null; //L7  
    }  
}
```

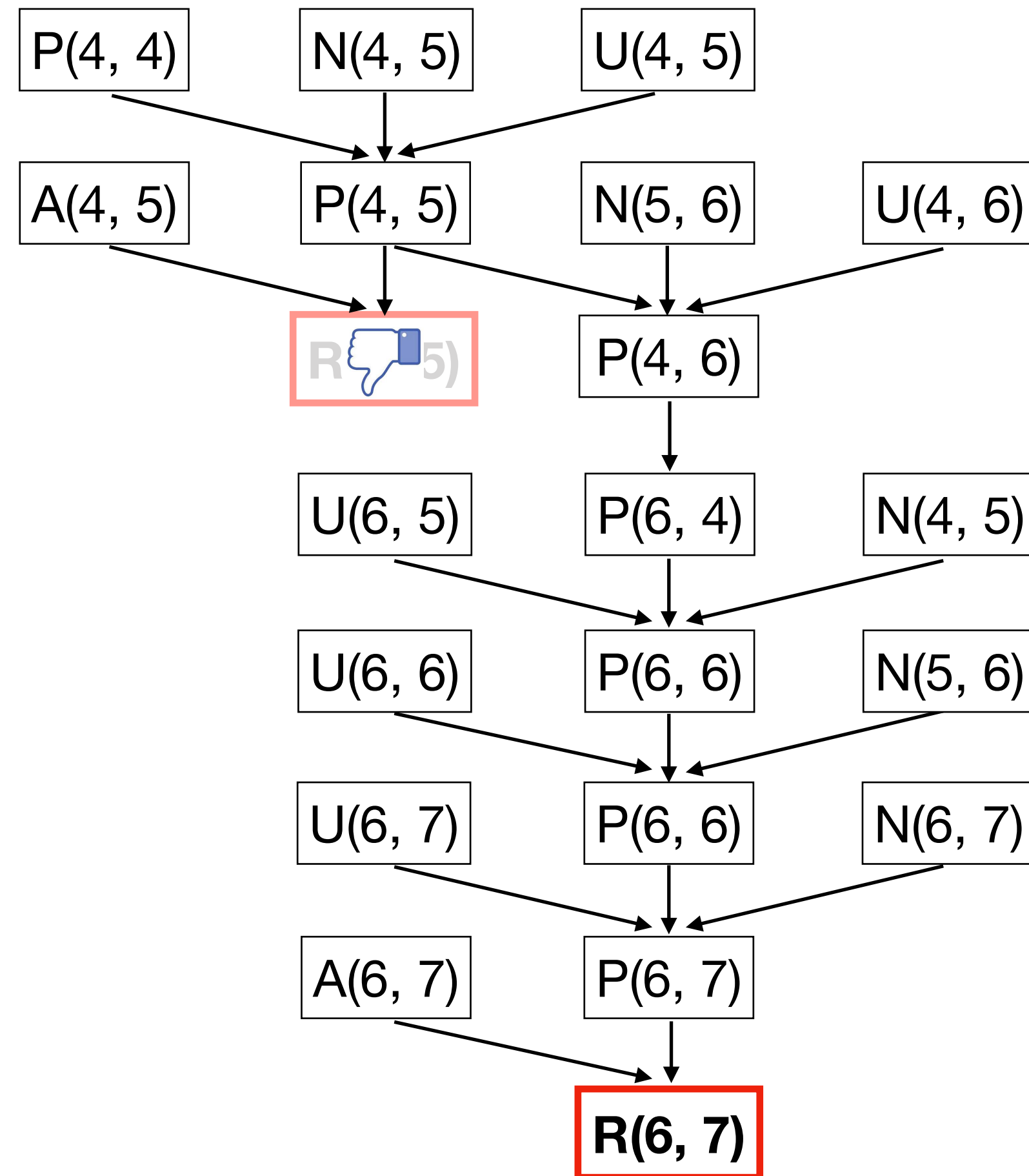
Ranking	Alarm	Confidence	
1	R(4, 5)	0.398	
2	R(5, 5)	0.378	
3	R(6, 7)	0.324	
4	R(7, 7)	0.308	
5	R(0, 7)	0.279	

**Q: What are the probabilities of the other alarms when R(4,5) is false?**

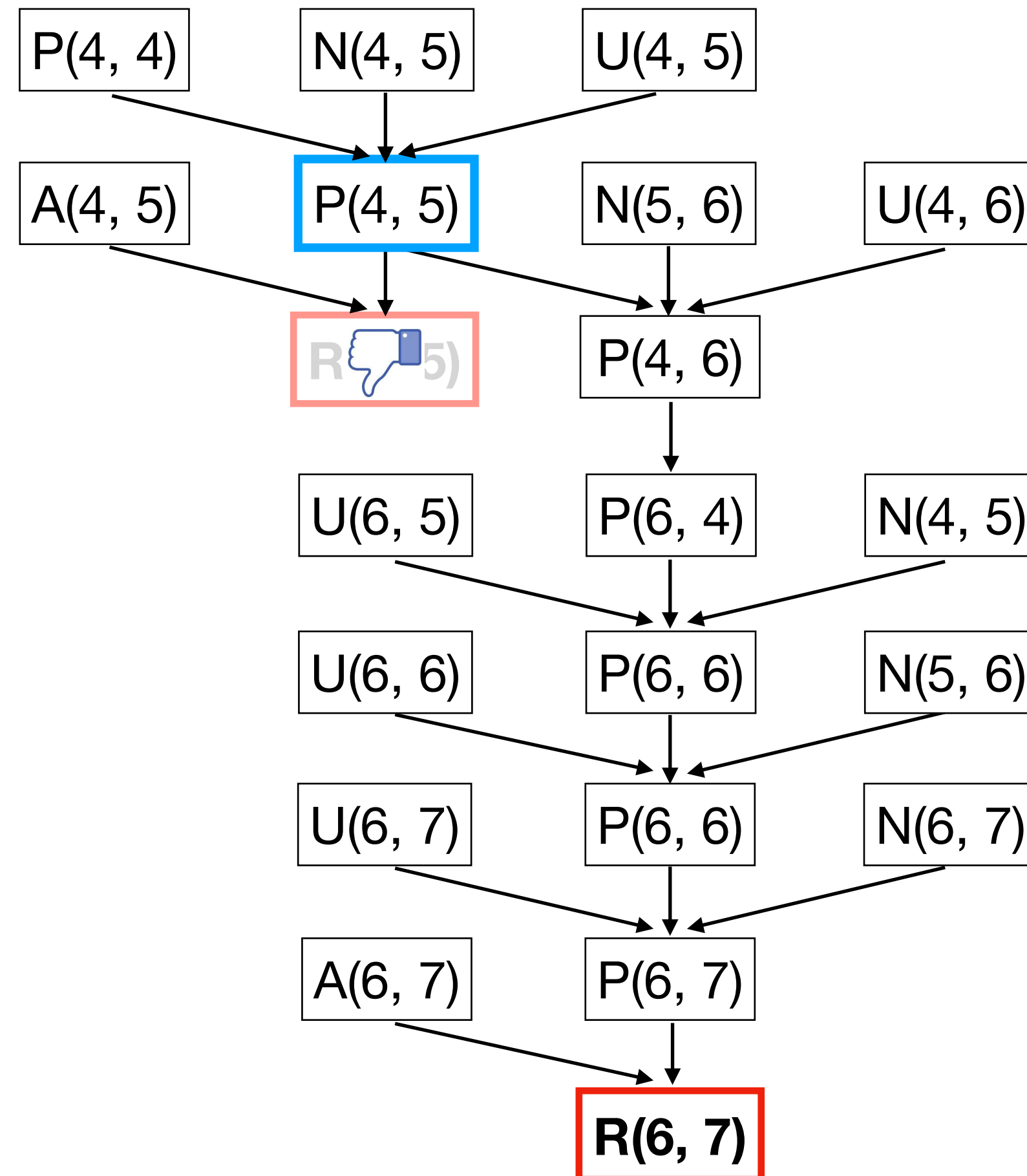
# 사용자의 반응에 따른 변화



# 사용자의 반응에 따른 변화

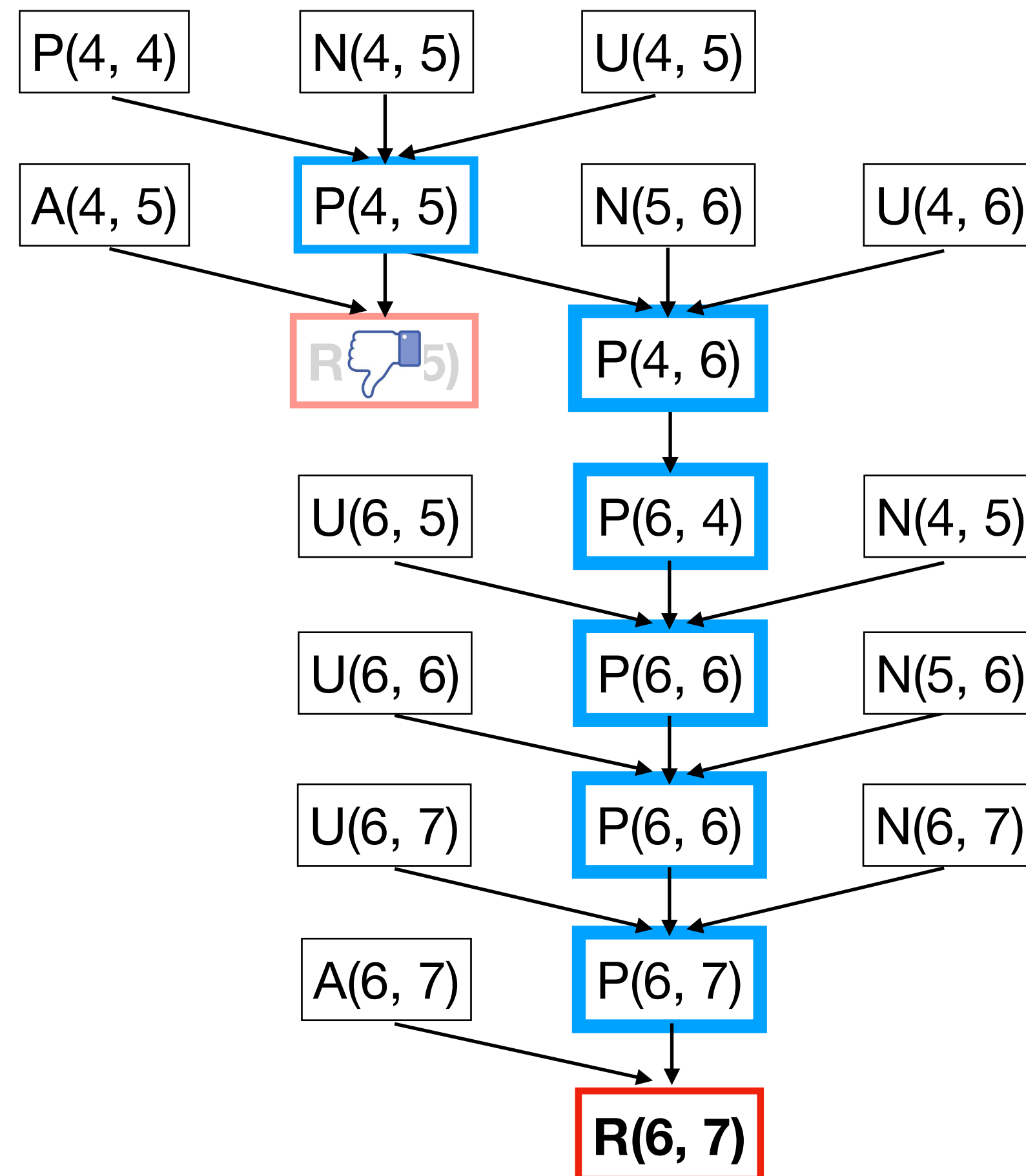


# 사용자의 반응에 따른 변화



$$\begin{aligned}
 &Pr(P(4,5) \mid \neg R(4,5)) \\
 &= Pr(\neg R(4,5) \mid P(4,5)) * \\
 &\quad Pr(P(4,5)) / Pr(\neg R(4,5)) \\
 &= 0.03
 \end{aligned}$$

# 사용자의 반응에 따른 변화

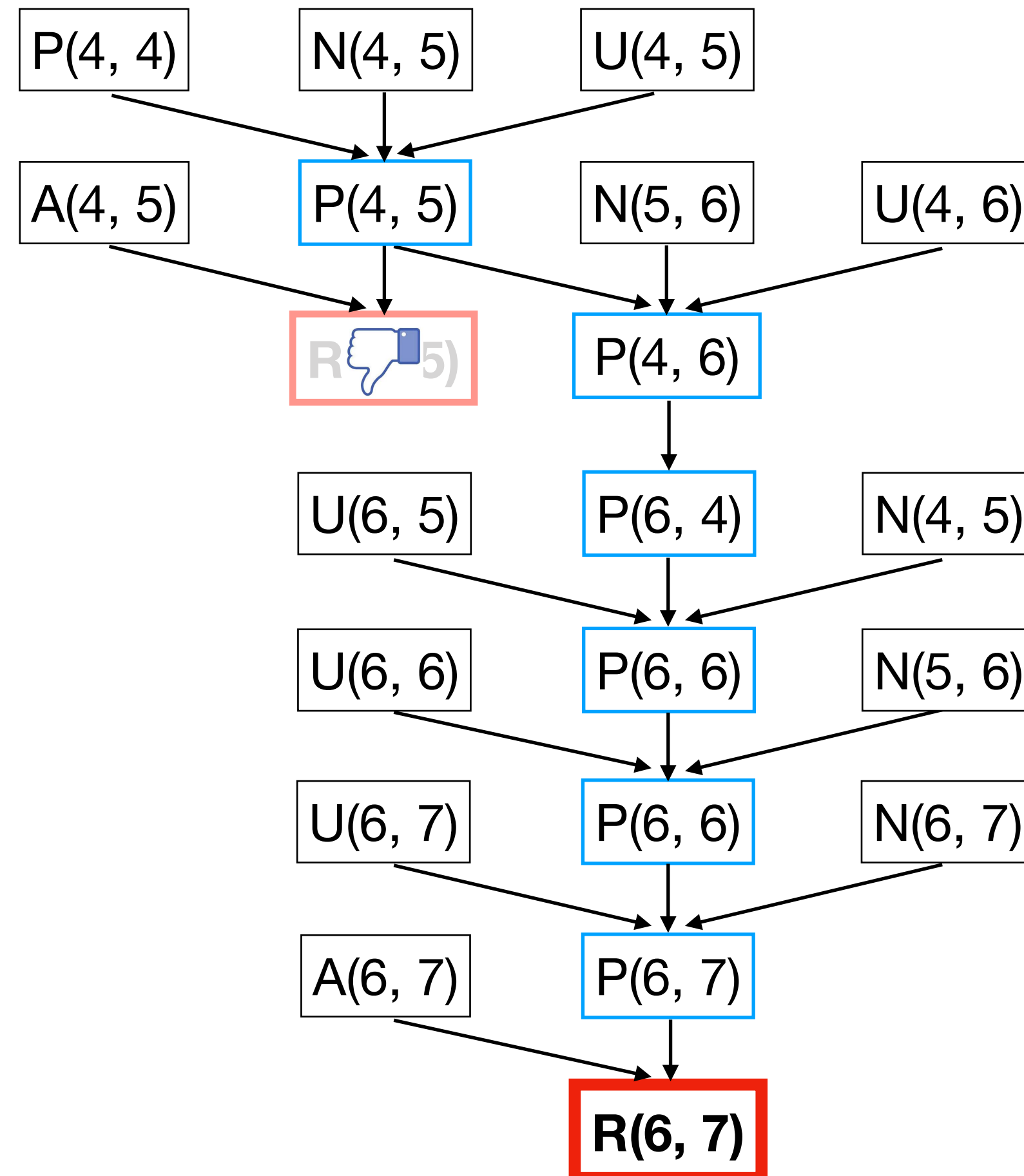


$$\begin{aligned}
 &Pr(P(4,5) \mid \neg R(4,5)) \\
 &= Pr(\neg R(4,5) \mid P(4,5)) * \\
 &\quad Pr(P(4,5)) / Pr(\neg R(4,5)) \\
 &= 0.03
 \end{aligned}$$

By Bayes's Rule:  
 $Pr(A|B) = Pr(B|A) * Pr(A) / Pr(B)$



# 사용자의 반응에 따른 변화




$$\begin{aligned}
 &Pr(P(4,5) \mid \neg R(4,5)) \\
 &= Pr(\neg R(4,5) \mid P(4,5)) * \\
 &\quad Pr(P(4,5)) / Pr(\neg R(4,5)) \\
 &= 0.03
 \end{aligned}$$

By Bayes's Rule:  
 $Pr(A|B) = Pr(B|A) * Pr(A) / Pr(B)$

$$\begin{aligned}
 &Pr(R(6,7) \mid \neg R(4,5)) \\
 &= Pr(R(6,7) \mid P(4,5)) * \\
 &\quad Pr(P(4,5) \mid \neg R(4,5)) \\
 &= 0.03
 \end{aligned}$$


# 기존 순위

Ranking	Alarm	Confidence
1	R(4, 5)	0.398
2	R(5, 5)	0.378
3	R(6, 7)	0.324
4	R(7, 7)	0.308
5	R(0, 7)	0.279

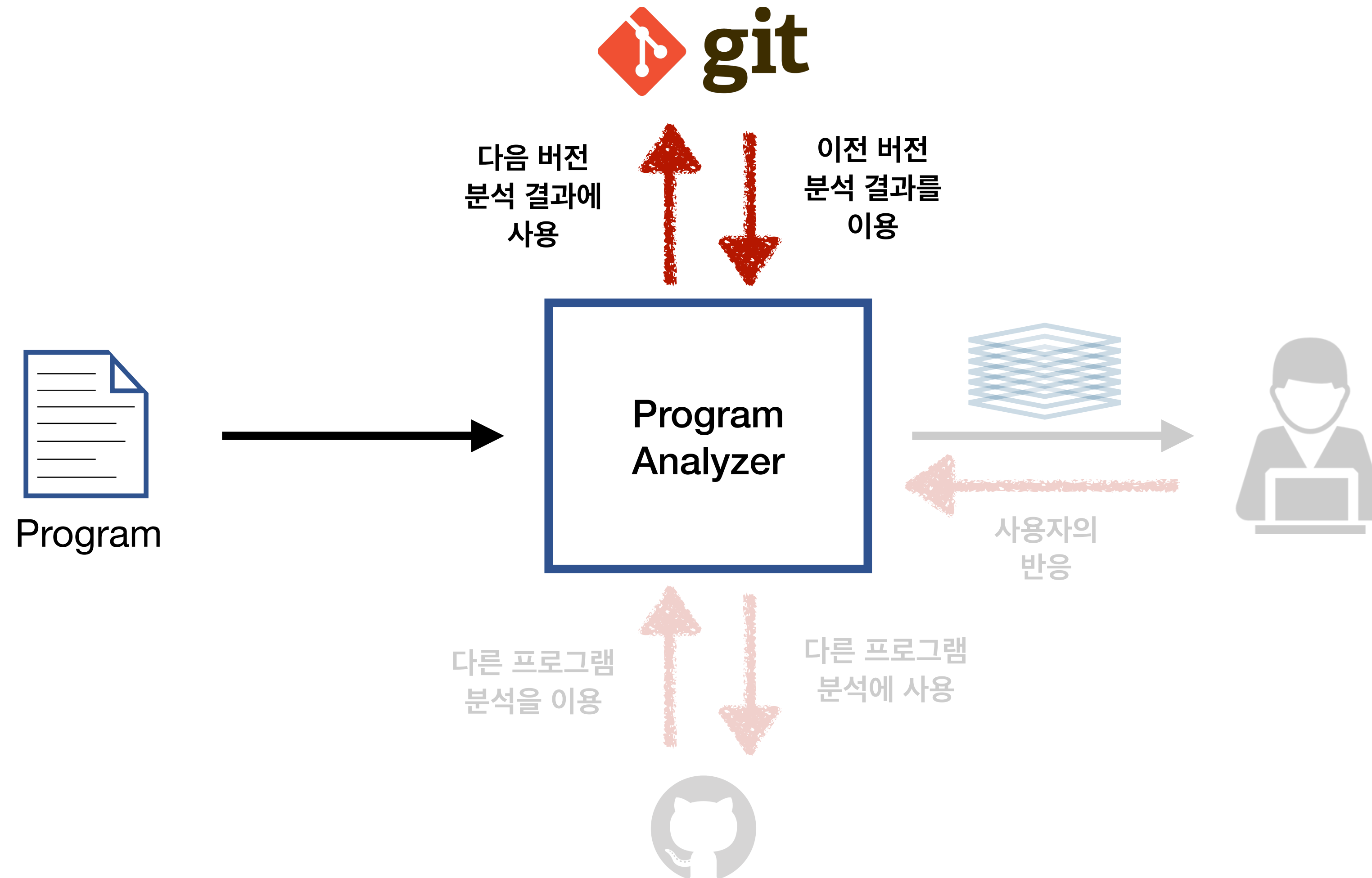


# 개선된 순위

Ranking	Alarm	Confidence
1	R(0, 7)	0.279
2	R(5, 5)	0.035
3	R(6, 7)	0.030
4	R(7, 7)	0.028
5	R(4, 5)	0

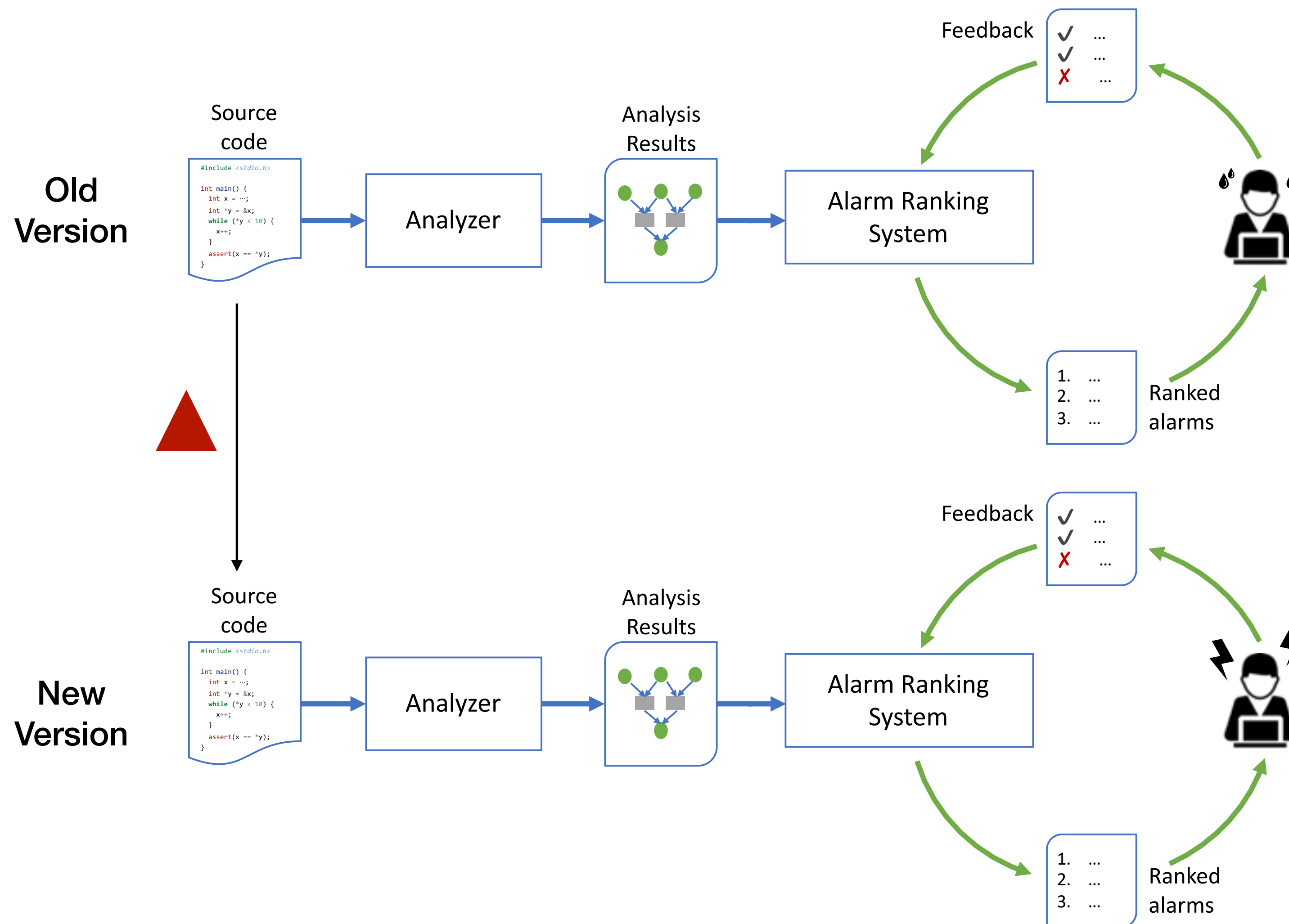


# 정적 분석의 상호 작용



**Drake: 프로그램의 변화를 감지하는 정적 분석**  
**[PLDI'19]**

# 일괄형 (batch-mode) 분석



# 연속적 (continuous) 분석

“We only display results for most analyses on **changed lines** by default; this keeps analysis results **relevant** to the code review at hand”, - Google, 2015

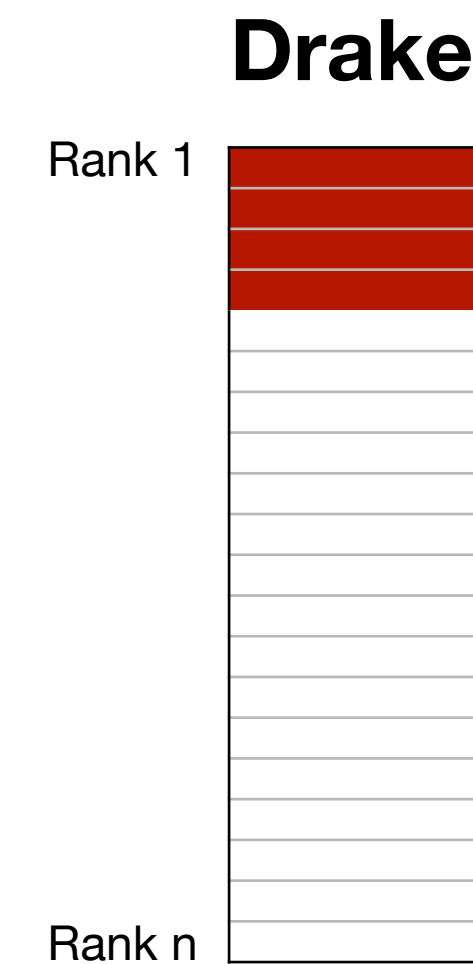
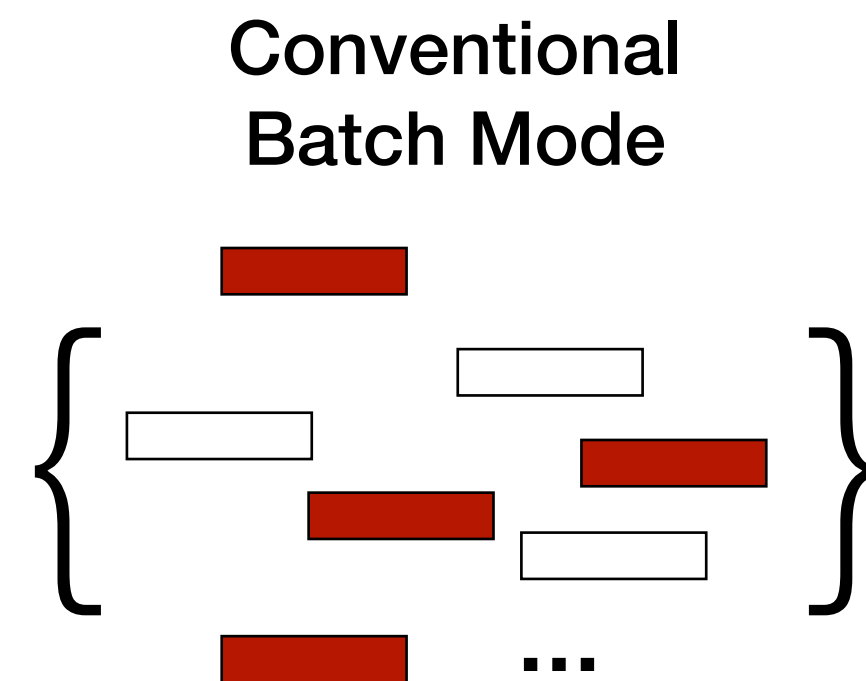
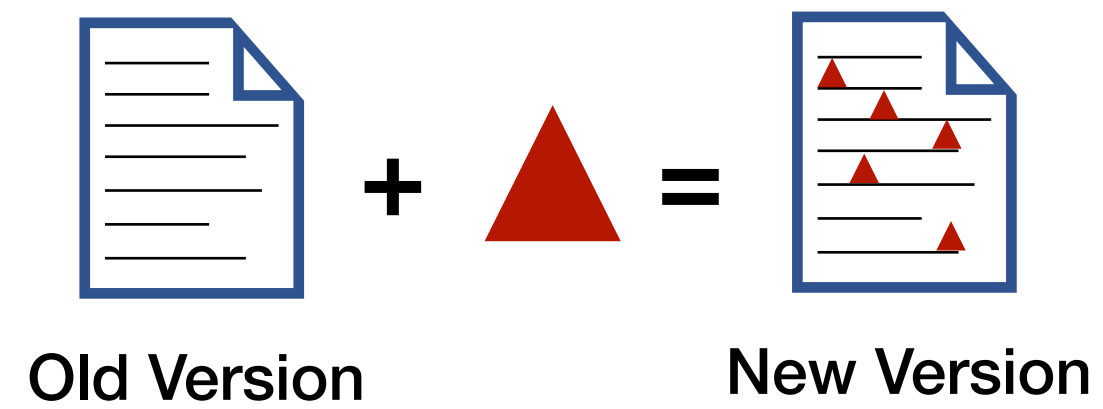
“The vast majority of Infer’s impact to this point is attributable to **continuous reasoning at diff time**”, - Facebook, 2016

“... is the ability to analyze a **changelist (a.k.a. a commit) rather than the entire codebase.** This functionality can help developers assess the quality and impact of a change ...”, - Microsoft, 2016

“In order to realize the goal, verification must continue to work with low effort **as developers change the code.** ... Neither of these approaches would work for Amazon as s2n is under **continuous development.**”, - Amazon, 2018

# 목표

- 코드 변화와 관련이 있는 순으로 정렬해주는 오류 보고 시스템



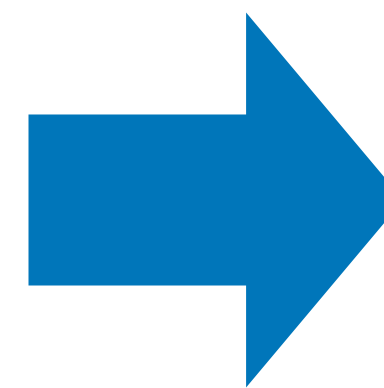
: Alarms relevant to the change  
: Alarms NOT relevant to the change



# 예제

## Old Version

```
1: x = input();  
2: y = input();  
3: x = opaque_dec(x);  
-4: y = opaque_dec(y);  
5: x++; // Alarm ✓  
6: y++; // Alarm ✓
```



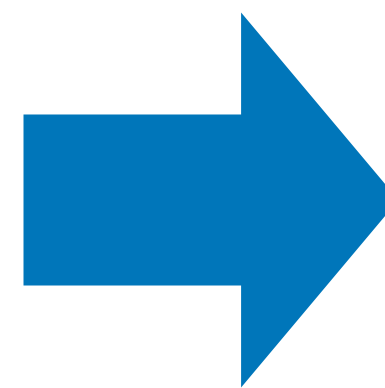
## New Version

```
1: x = input();  
2: y = input();  
3: x = opaque_dec(x);  
+4: y = identity(y);  
5: x++; // Alarm ✓  
6: y++; // Alarm 🐛
```

# 예제

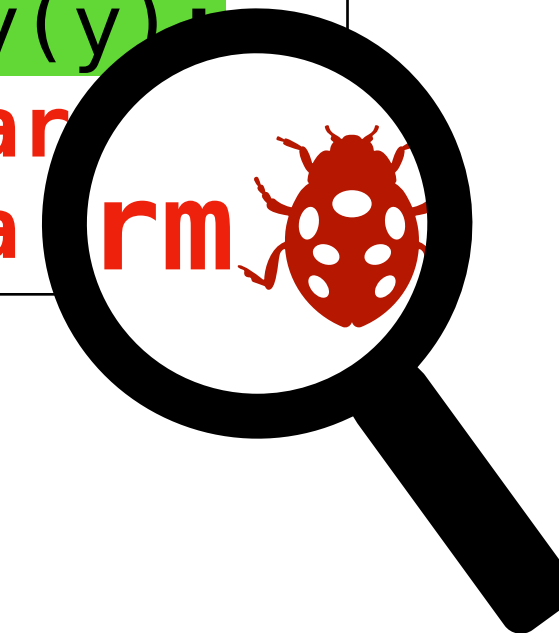
## Old Version

```
1: x = input();  
2: y = input();  
3: x = opaque_dec(x);  
-4: y = opaque_dec(y);  
5: x++; // Alarm ✓  
6: y++; // Alarm ✓
```



## New Version

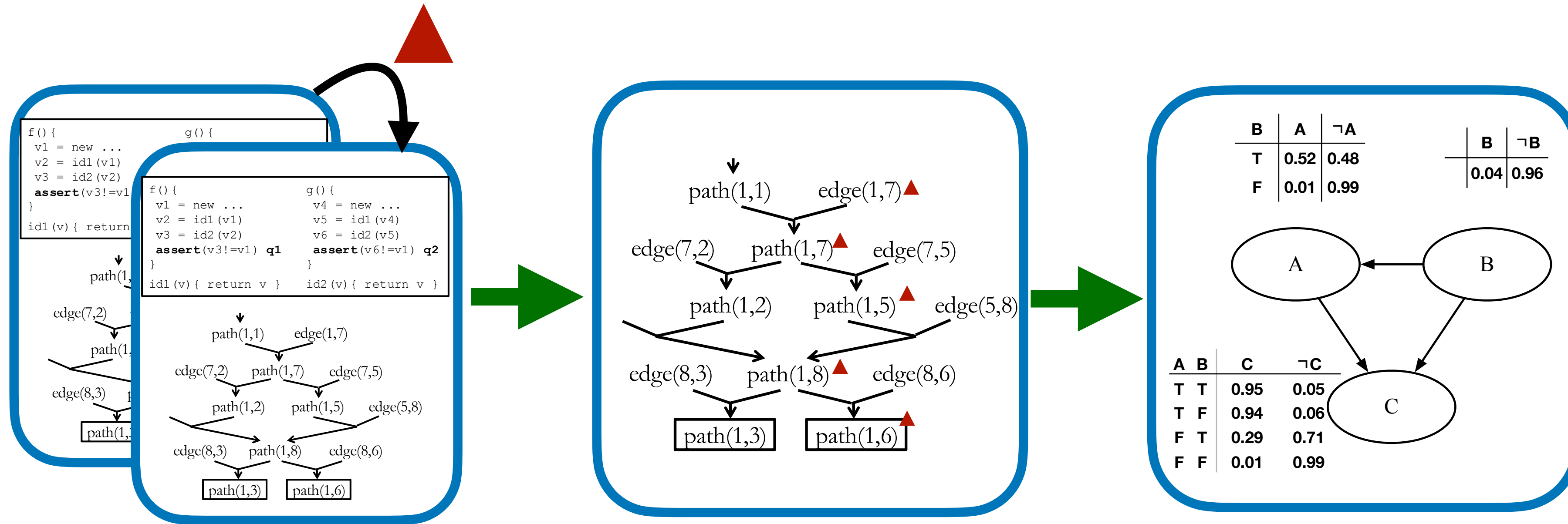
```
1: x = input();  
2: y = input();  
3: x = opaque_dec(x);  
+4: y = identity(y);  
5: x++; // Alarm  
6: y++; // Alarm
```



Q: How to emphasize the alarm at 6 that is relevant to this change?

# 핵심 기술

## 변화 감지 분석 결과 + 베이지안 추론 (Bayesian inference)



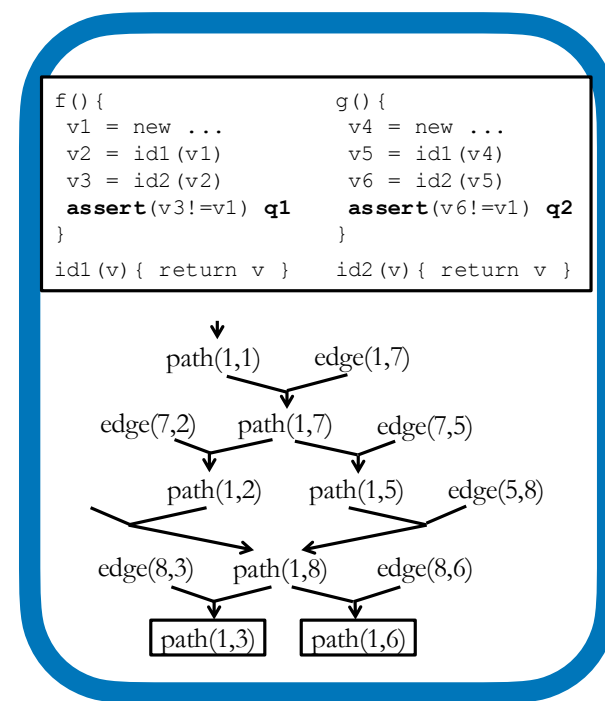
두 버전의 분석 결과

코드 변화를 감지하는 분석 결과

확률 모델  
(Bayesian Network)

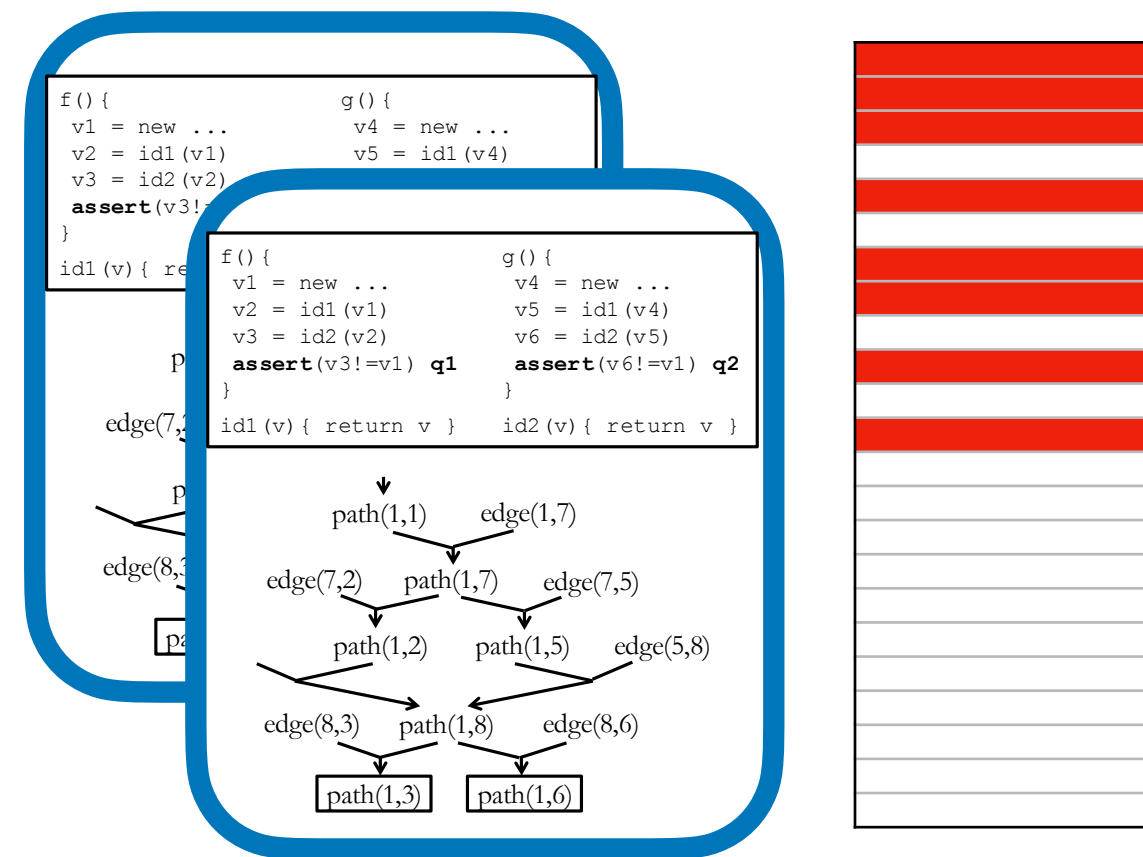
# 효과

평균 3개 버그를 찾기 위한 드는 노력의 평균을 측정



일괄형

563 개 경보 사이에  
모든 버그가 산재



관련성 기반 순위

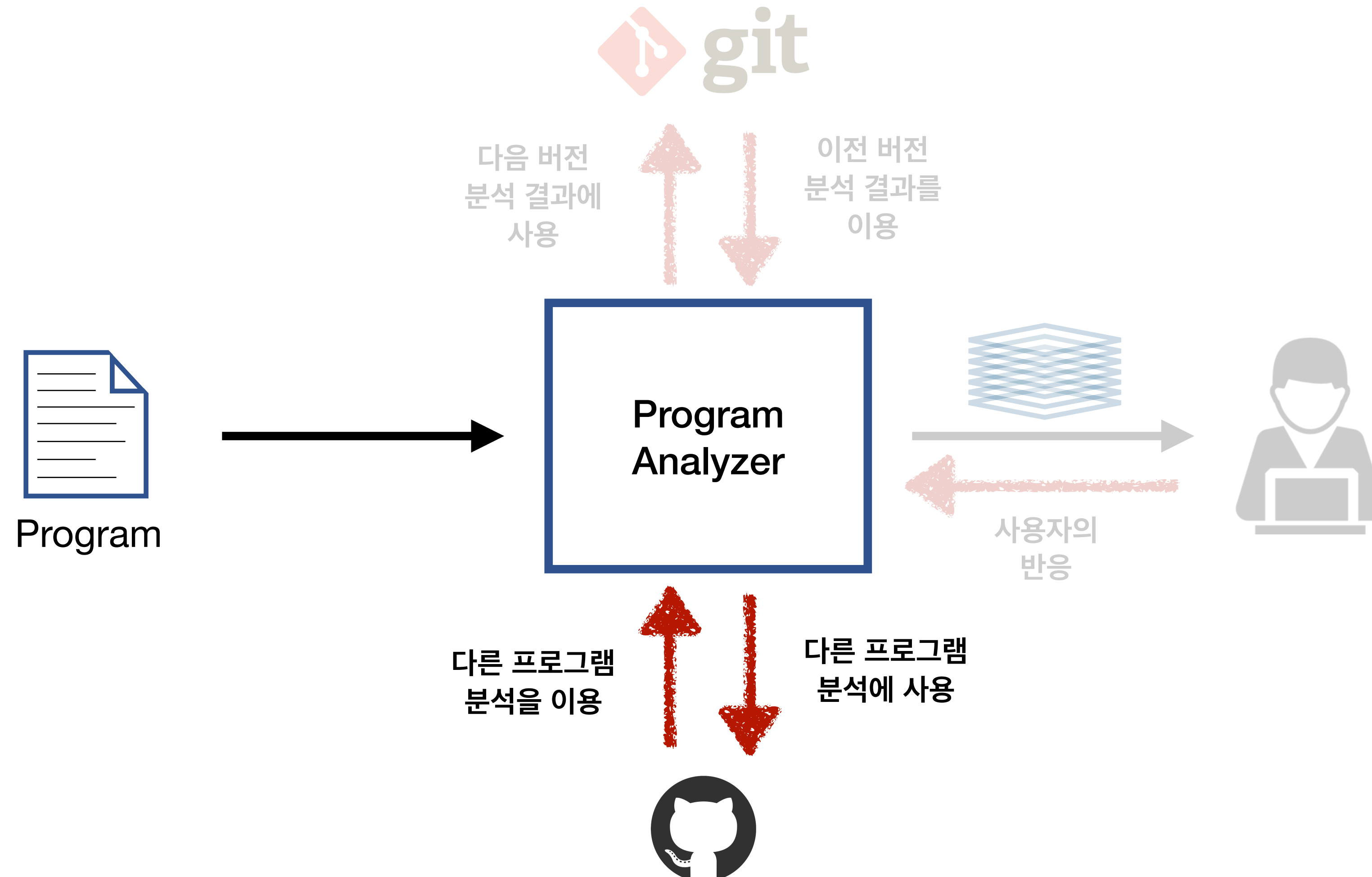
최대 순위 94 위 이내에  
모든 버그 위치



+ 상호 작용 기반 순위

상호 작용 30 회 이내에  
모든 버그 검출

# 정적 분석의 상호 작용

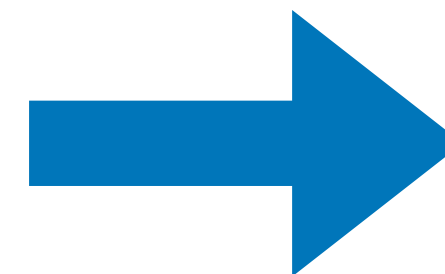


**BayeSmith: 다른 프로그램 분석에서 배우는 정적 분석**  
**[현재 진행]**

# 잘못된 일반화

```
/* GNU sort-7.2 */
1: size_t avoid_trashing_input (struct sortfile *files, size_t ntemps, ...) {
2:     for (i = ntemps; i < nfiles; i++) {
3:         if (/* file[i] is the same as outfile */) {
4:             while (i + num_merged < nfiles) {
5:                 num_merged += mergefiles (&files[i], 0, nfiles - i, tftp, temp);
6:                 files[i].name = temp; // false alarm
7:                 files[i].pid = pid; // false alarm
8:                 memmove(&files[i], &files[i + num_merged], // true alarm
9:                     num_merged * sizeof *files);
10:                 ntemps += 1;
11:                 nfiles -= num_merged - 1;;
12:                 i += num_merged;
13:             }
14:         }
15:     }
16: }
```

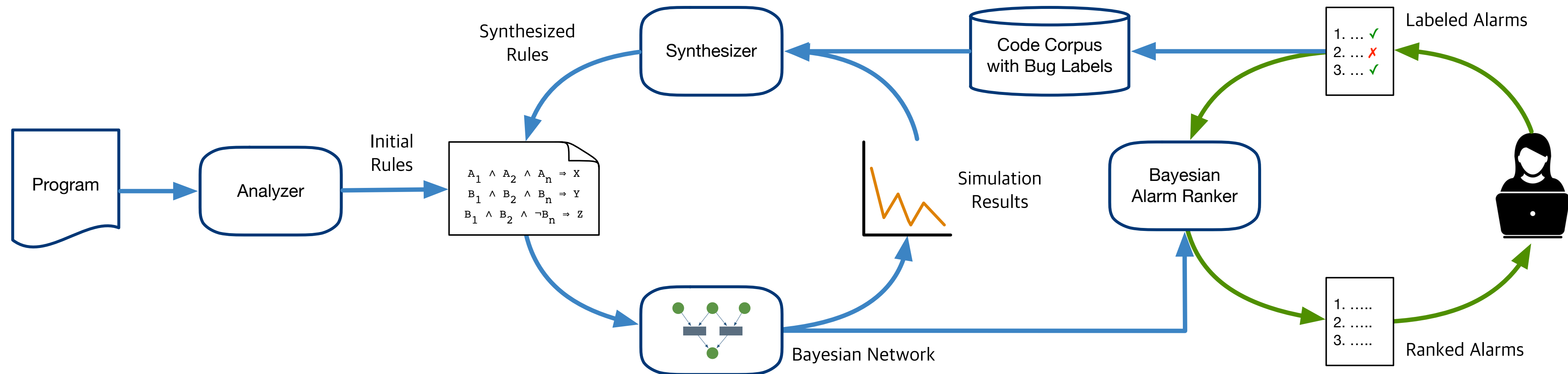
Ranking	Alarm	Confidence
1	Line 6	0.93
2	Line 7	0.92
3	Line 8	0.91
...		



Ranking	Alarm	Confidence
...		
132	Line 7	0.41
133	Line 8	0.40
...		

# 핵심 기술

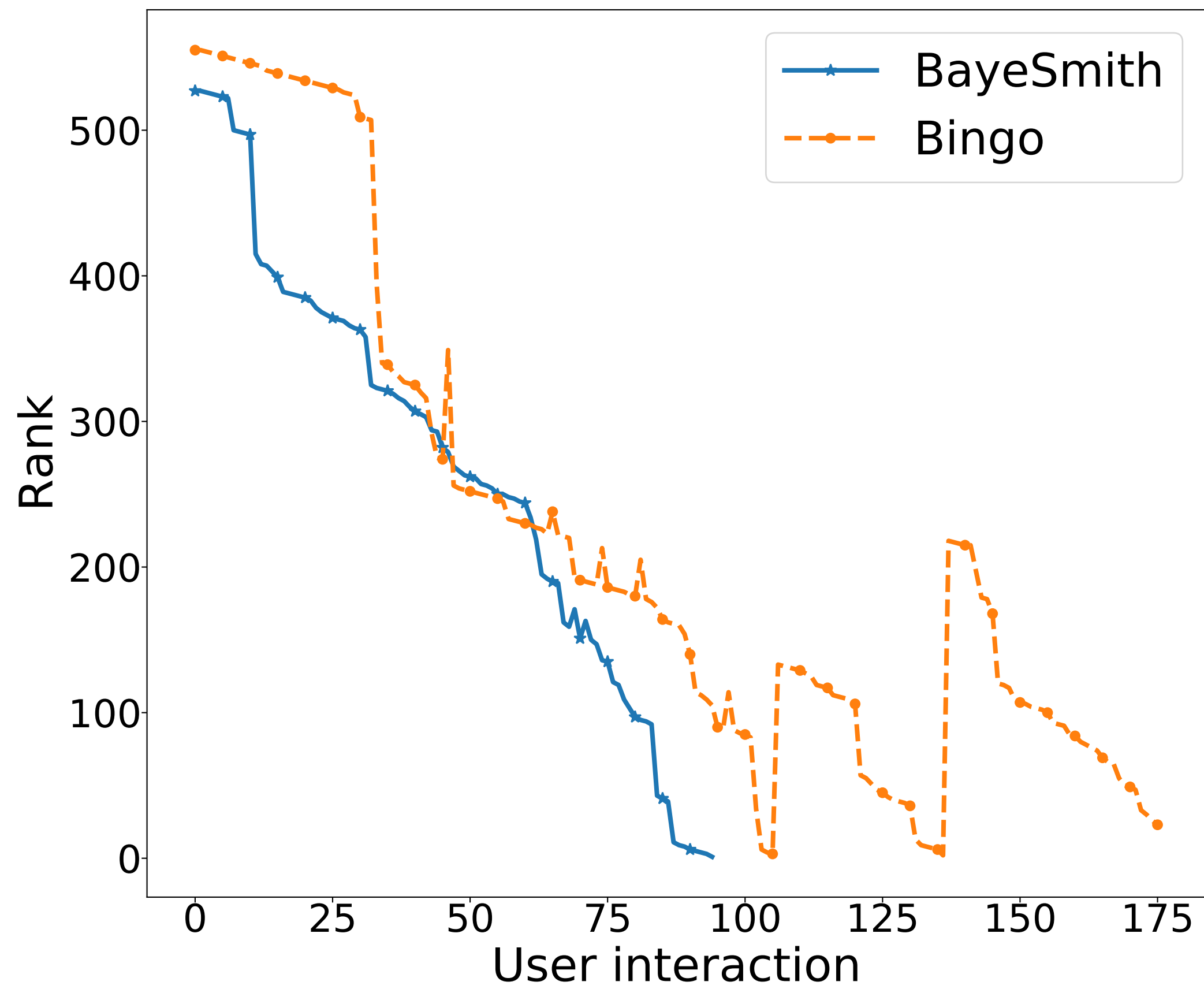
## 프로그램 합성 + 베이지안 추론 (Bayesian inference)



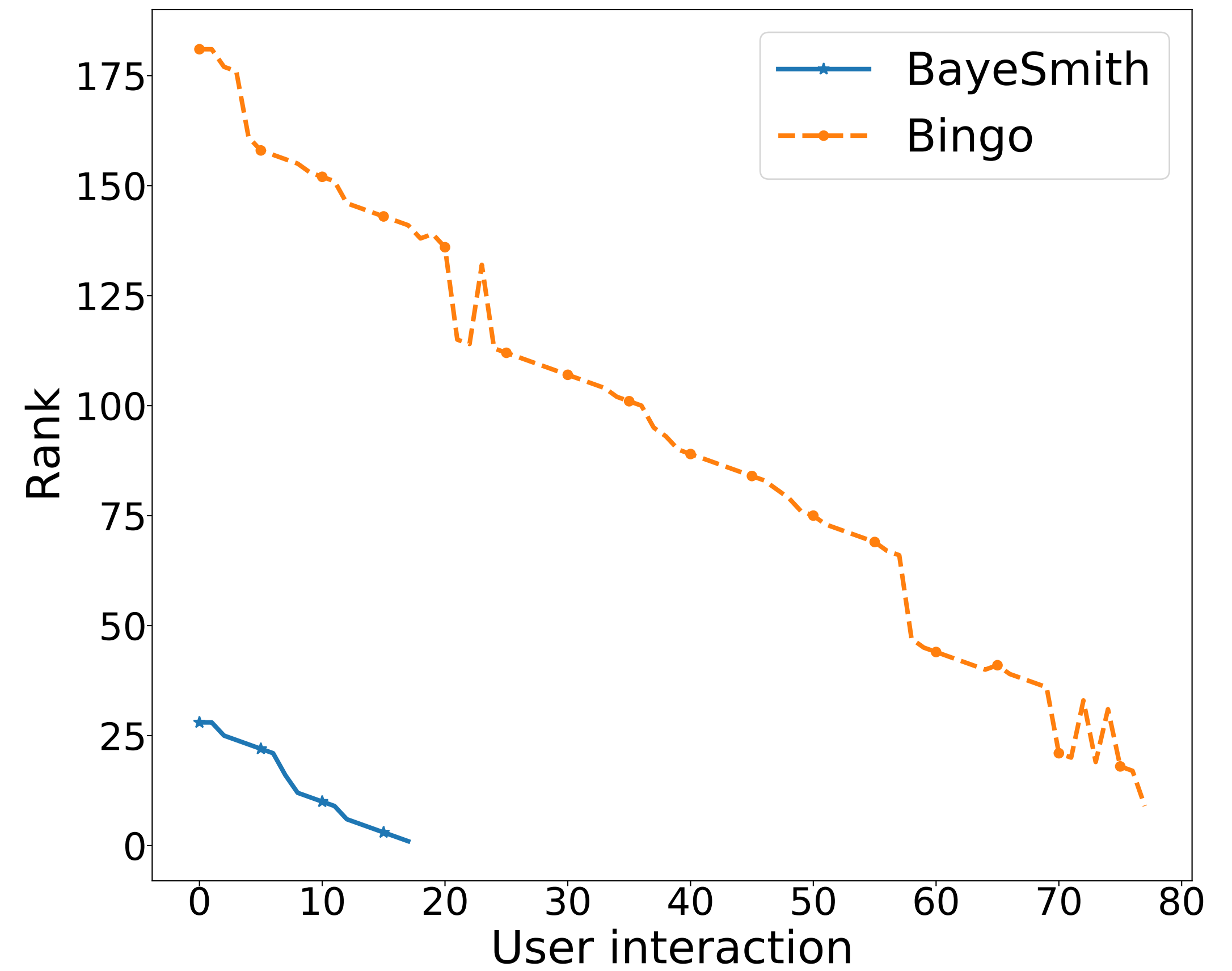


# 성능

sort



readelf



# 결론

- 사용자가 정적 분석을 이해해야 하는 시대 → 정적 분석이 사용자를 이해하는 시대
- 여러 가지 방식으로 상호작용
  - Bingo: 사용자와 직접 상호작용
  - Drake: 이전 버전과 상호작용
  - BayeSmith: 다른 프로그램의 분석 결과 상호작용
- 핵심 기술: 정적 분석 + 프로그램 합성 + 통계적 추론