

# LLM 질의를 통한 정적 오염분석 허위 경보 제거

## LLM-driven False Alarm Classification for Static Taint Analysis

주강대 조한결 이우석

Programming system Lab.  
Hanyang university



# 정적분석의 허위경보 문제 및 LLM 기반 접근의 한계

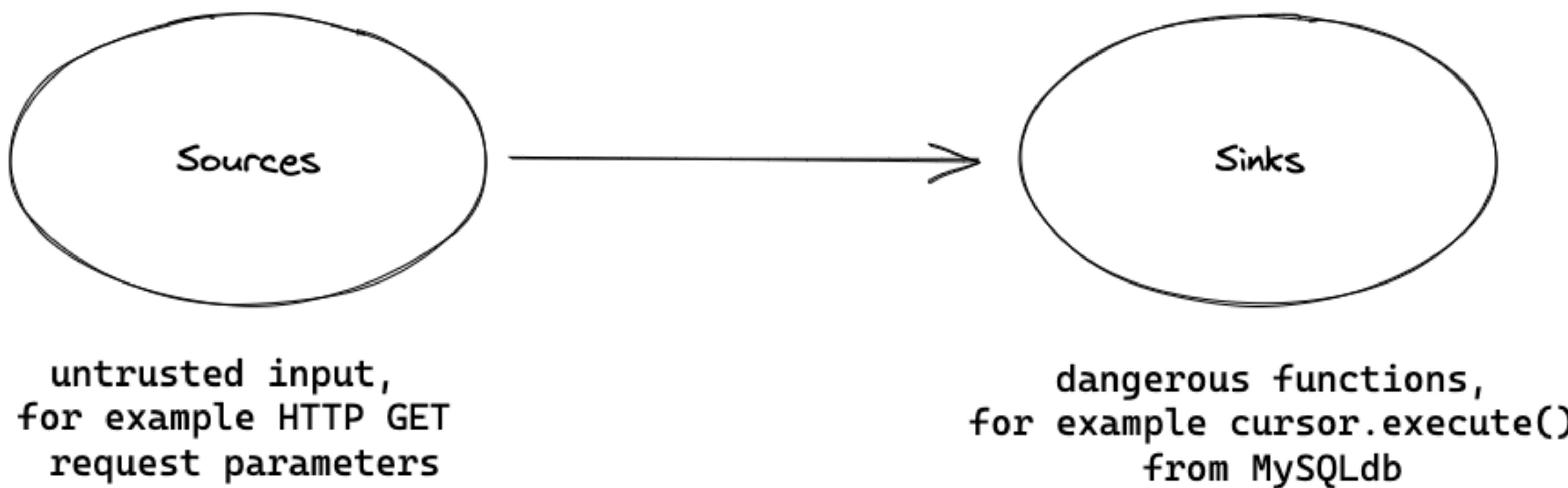
- 기존의 정적분석 도구들은 경우에 따라 수 많은 허위 경보를 만들어내며, 이를 식별하는 작업은 **많은 시간과 노력을 요구**
- 최근, 정적분석의 허위 경보 문제 개선을 위해 **LLM**을 적용하는 연구에 대한 관심 증가

## LLM을 사용한 접근에는 한계가 존재

- 환각(Hallucination), 답변 일관성, 컨텍스트 윈도우 초과 등 문제에 취약
- 경우에 따라, LLM이 만드는 결과를 완전히 신뢰하기 어려우며, 진짜 경보를 놓칠 가능성 존재

# 오염 분석 (Taint Analysis) 이란?

- **개요** : 특정 값이나 상태가 “문제가 되는 지점까지 도달할 수 있는지”를 추적하는 경로 중심 정적 분석
- **오염 분석의 핵심 구조**
  - ▶ **Source** : 외부 입력 또는 위험 상태가 생성되는 위치
  - ▶ **Sink** : Source에서 전파된 오염이 도달하면 문제가 발생하는 위치
  - ▶ **Sanitizer** : 오염 전파를 차단하는 조건 또는 연산



# 단일 실행 경로 단위 문제로 분해

“Key Idea : 오염분석 경보 분류 문제를 단일 실행 경로 단위 문제로 분해하여 접근”

- 오염분석 경보를 분류하는 문제는 **복잡하고 어려움**

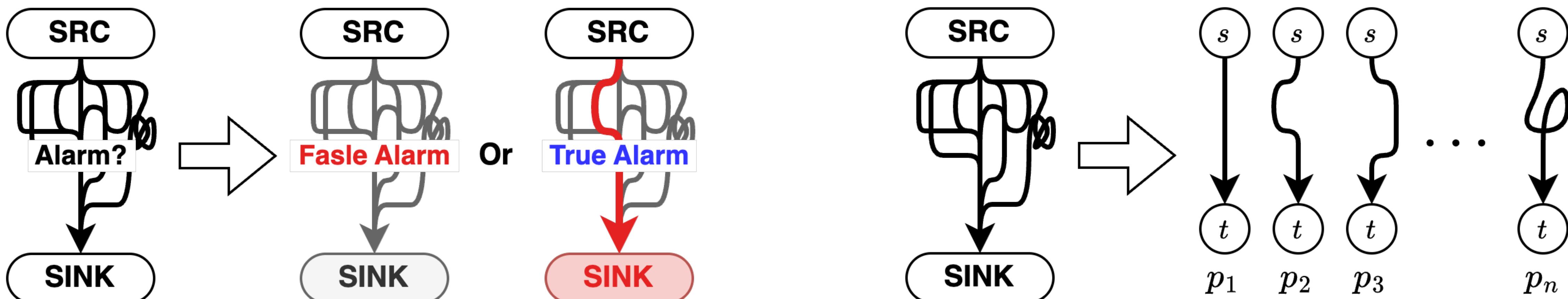
- ▶ 실제 가능한 오염 도달 경로가 하나도 없음이 확인됨 → 허위 경보로 판단

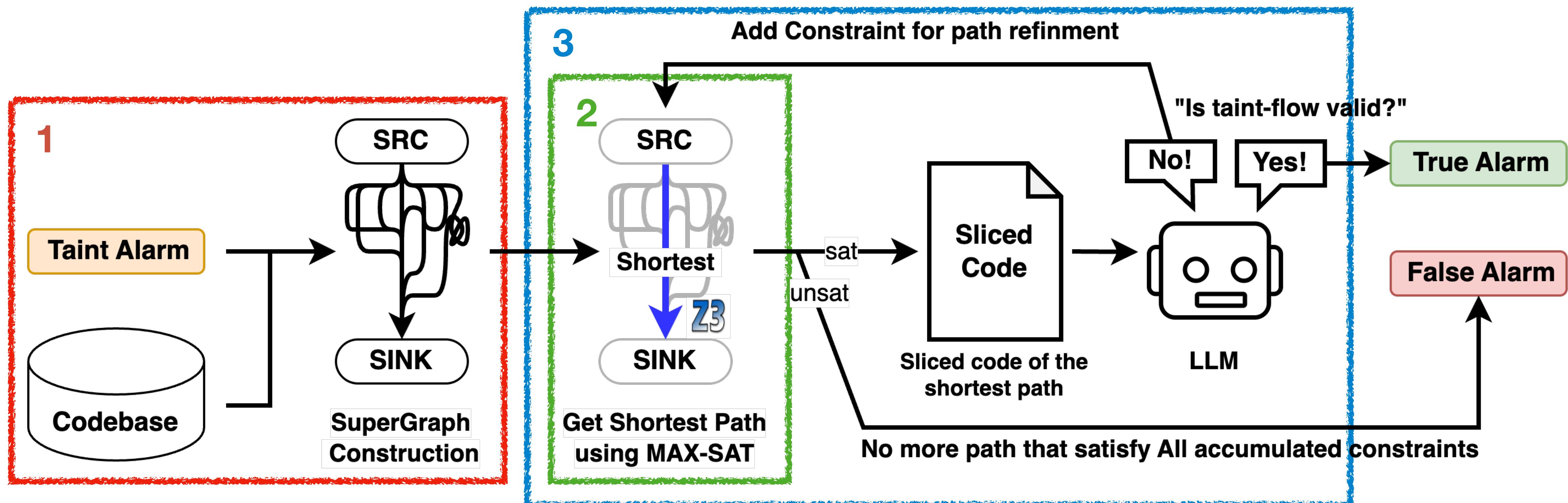
- 전역 경로를 동시에 고려하는 접근 X

- ▶ 가능한 모든 경로를 고려하므로 전역적 추론 요구 / 각 경로 별 문맥 기억을 장기적으로 유지 해야함

- 문제를 단일 실행 경로 단위로 분해하는 접근 ✓

- ▶ 해당 경로의 타당성만 판단하는 국소적 추론만 요구 / 불필요한 문맥 고려 없이, 경로 상 코드 의미에만 집중



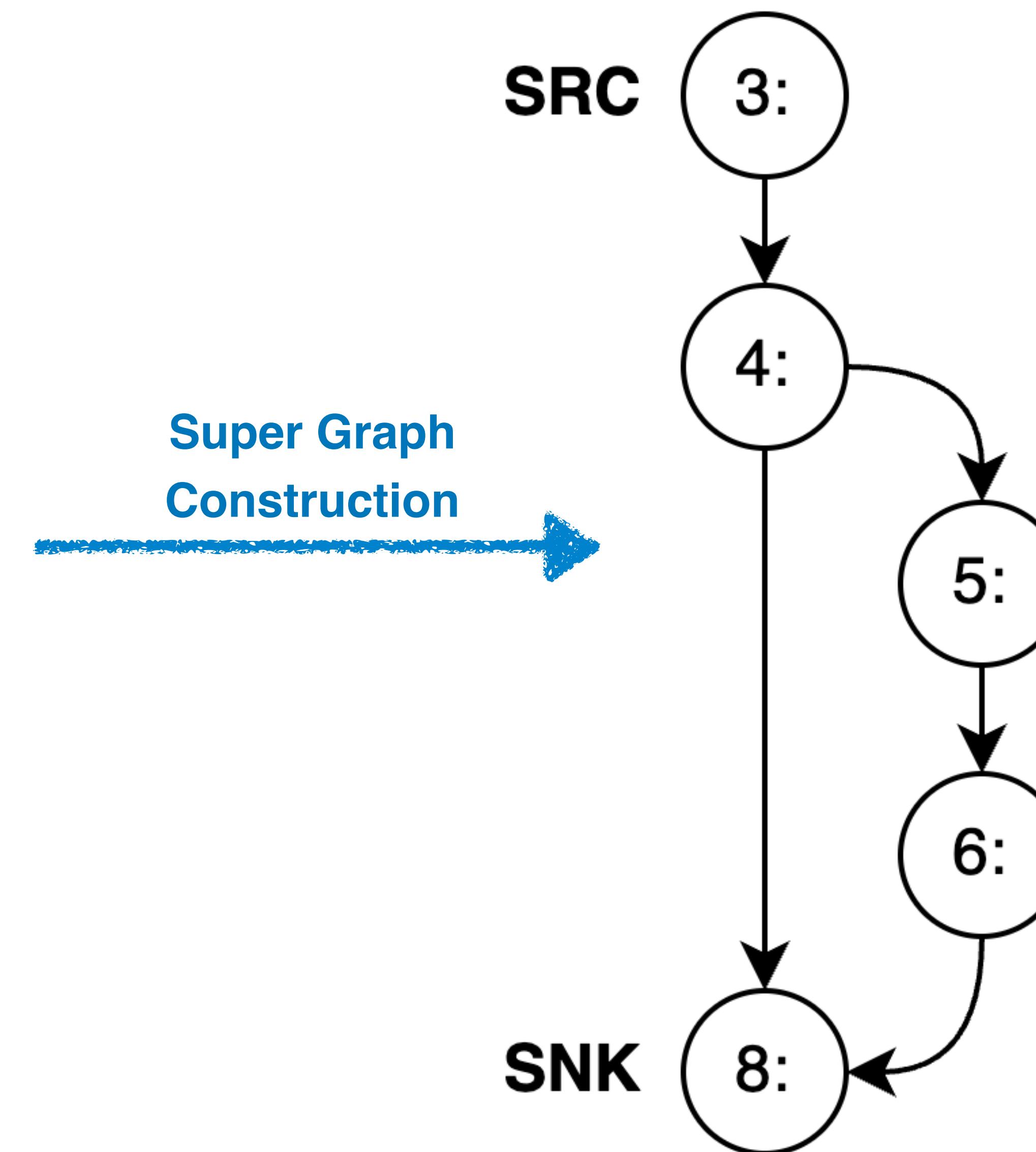


1. Super Graph 생성 (모든 경로를 포함하는 하나의 통합된 Graph 형태로 표현하기 위함)
2. MAX-SAT 기반의 누적 제약을 고려한 최단 경로 계산
3. LLM 질의를 통한 제약 누적 및 경로 정제 Loop

# LLM 질의를 통한 제약 누적 및 경로 정제 Loop 예시

```
1 bool foo(){
2     ret = false;      Source
3     Mem* p = new Mem(100);
4     if(p != null){
5         sp.reset(p);
6         ret = bar(p);
7     }                  Sink
8     return ret;
9 }
```

간단한 메모리 누수 경보 예제 코드에서 Graph 생성

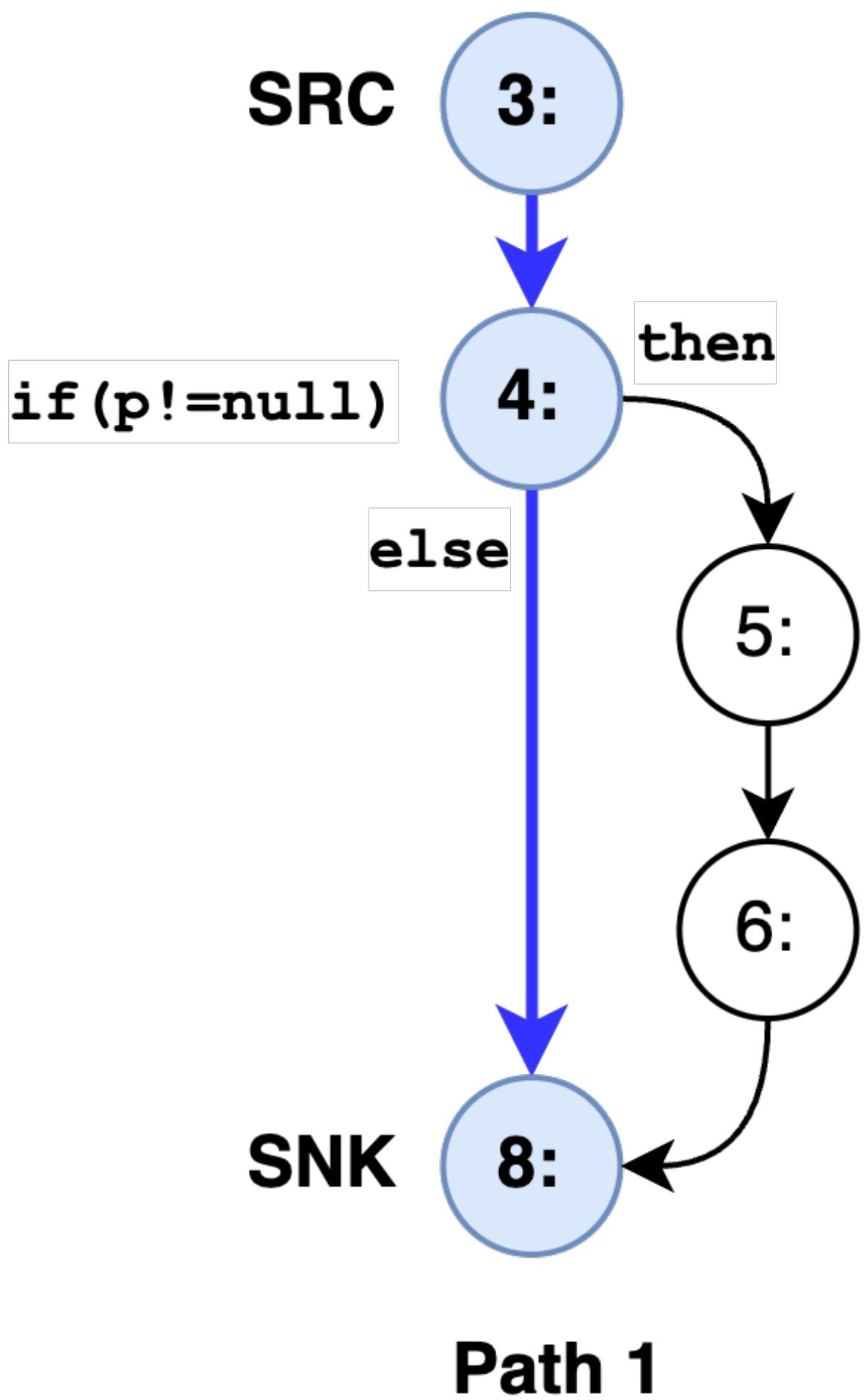


# LLM 질의를 통한 제약 누적 및 경로 정제 Loop 예시 ( 1 )

경로 제약:

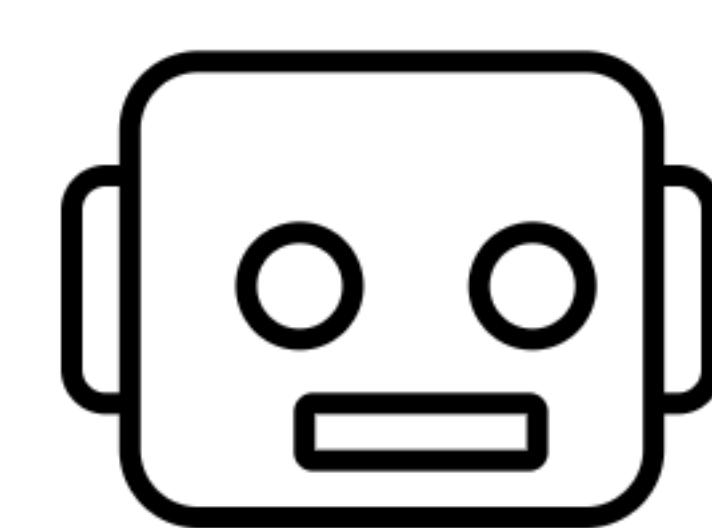
True (초기 경로 제약 없음)

MAX-SAT 기반 최단 경로 계산:



SRC-SINK 간 오염 전파 가능?

```
1 bool foo() {  
3     Mem* p = new Mem(100);  
4     if(p != null) {  
7         }  
8     return ret;  
9 }
```



LLM

NO!

p에 메모리 할당,  
if 조건 항상 만족  
되어야함

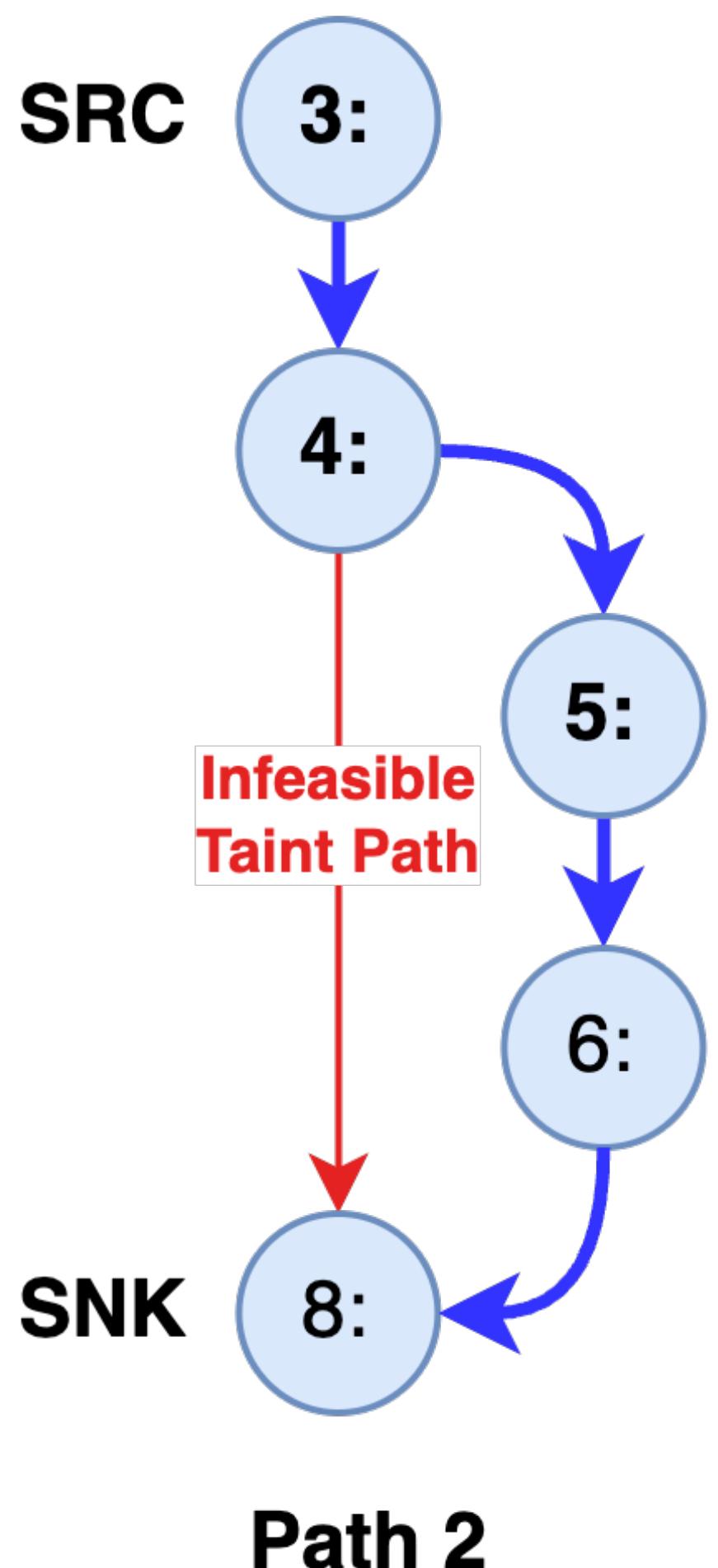
경로 제약 추가:  
 $\neg Edge_{4,8}$

# LLM 질의를 통한 제약 누적 및 경로 정제 Loop 예시 (2)

누적 경로 제약:

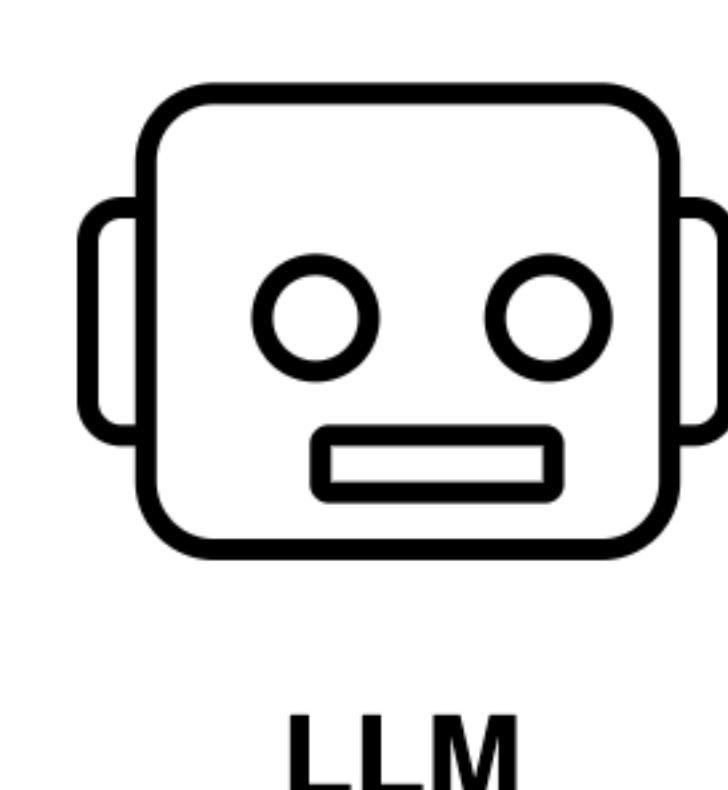
$$True \wedge \neg Edge_{4,8}$$

MAX-SAT 기반 최단 경로 계산:



SRC-SINK 간 오염 전파 가능?

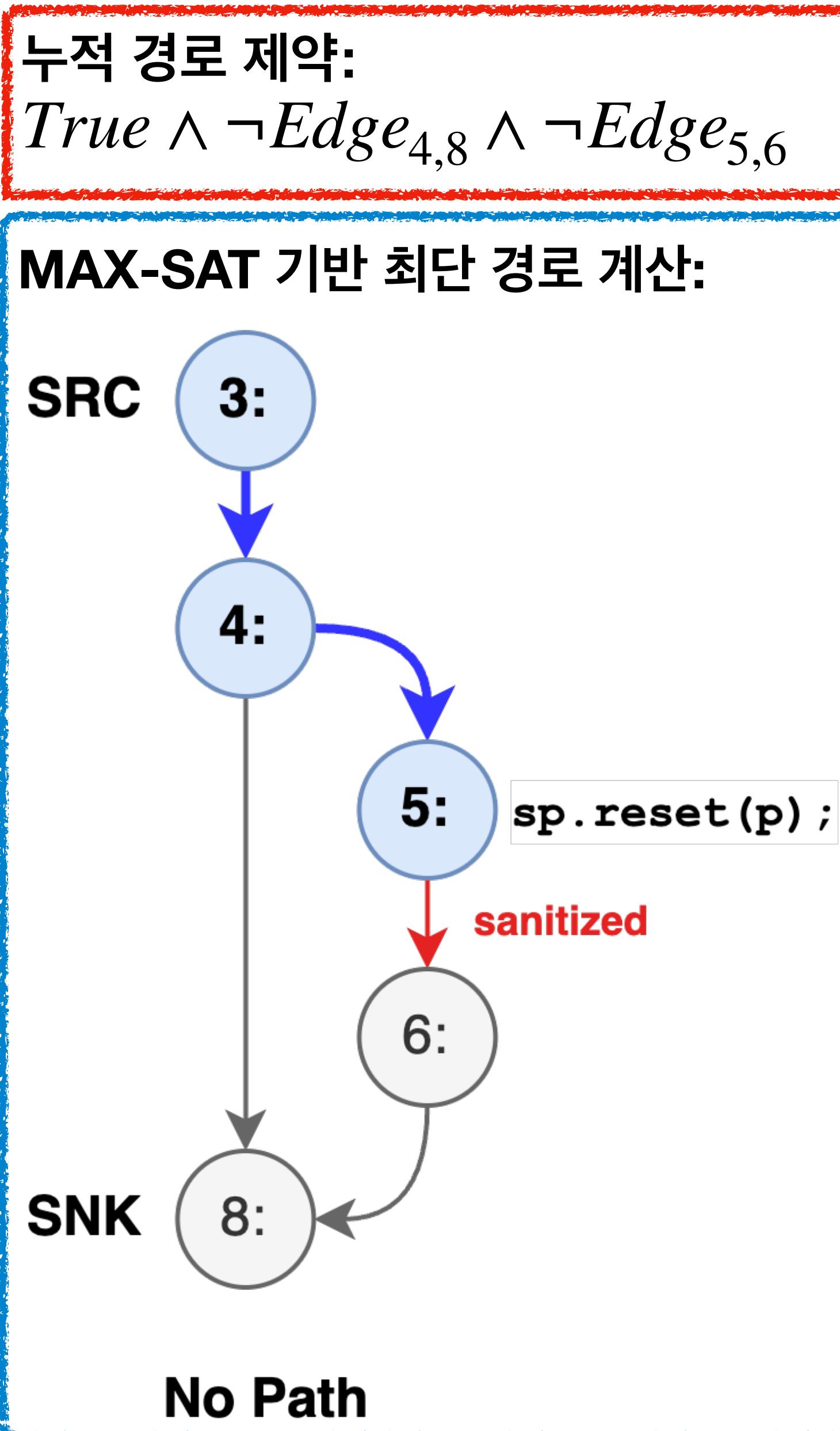
```
1 bool foo() {  
3     Mem* p = new Mem(100);  
4     if(p != null) {  
5         sp.reset(p);  
6         ret = bar(p);  
7     }  
8     return ret;  
9 }
```



NO!  
명백한  
Sanitizer 발견

경로 제약 추가:  
 $\neg Edge_{5,6}$

# LLM 질의를 통한 제약 누적 및 경로 정제 Loop 예시 (3)



UNSAT : No path  
Loop Terminate

This is False alarm

# 경보 분류 성능 평가 실험 결과

경보 분류 성능 평가 실험에서 Baseline 대비 Accuracy·Recall·Precision 및 분산 전반에서 일관된 개선을 보임

Metric	Baseline	Ours
Accuracy	(72.57%, 19.45)	(82.36%, 6.90)
Recall	(79.04%, 88.91)	(87.69%, 17.75)
Precision	(20.95%, 20.93)	(33.95%, 15.46)

Accuracy: 전체 경보를 올바르게 분류한 비율

Recall: 진짜경보를 놓치지 않고 식별한 비율

Precision: 진짜라고 분류한 결과 중 실제 진짜경보의 비율

# LLM 질의를 통한 정적 오염분석 허위 경보 제거

LLM-driven False Alarm Classification for Static Taint Analysis

주강대, 조한결, 이우석 (한양대학교 프로그래밍 시스템 연구실)



# Thank You!

## LLM 질의를 통한 정적 오염분석 허위 경보 제거

LLM-driven False Alarm Classification for Static Taint Analysis

주강대, 조한결, 이우석 (한양대학교 프로그래밍 시스템 연구실)

### 1. Motivation

#### 1.1. 정적분석의 허위경보 문제

- GitHub CodeQL<sup>[1]</sup>과 같은 기존 정적 분석 도구들은 많은 **허위 경보** 발생.
- 허위 경보를 식별 작업 → 노동 집약적, 많은 시간
- 정적 분석 허위 경보 문제 해결에 LLM을 적용하는 연구<sup>[2][3]</sup>에 대한 관심 증가.
  - ▶ LLM은 **답변 일관성** 및 **환각(hallucination)** 등의 문제로 완전히 신뢰하기 어려움

### 3. 방법 (2)

#### 3.2. 최단 경로 구하기 (MAX-SAT)

Hard Constraints ( $\Phi_{\text{start}} \wedge \Phi_{\text{end}} \wedge \Phi_{\text{flow}}$ )

- 시작 노드 제약 ( $\Phi_{\text{start}}$ ) : Source의 Out-edge 중 하나 포함
- 종료 노드 제약 ( $\Phi_{\text{end}}$ ) : Sink의 In-edge 중 하나 포함
- 노드 흐름 제약 ( $\Phi_{\text{flow}}$ ) : edge의 연속적 흐름 제약
  - ▶  $\Phi_{\text{flow}}(n) = \text{IN}(n) \Rightarrow \text{Out}(n)$  if  $n \neq \text{src} \wedge n \neq \text{sink}$
- 사용자 제약 ( $\phi$ ) : LLM에 의해 추가, 누적되는 경로 제약 (초기값 : True)

Soft Constraints

- 모든 Edge 변수의 가중치를 -1로 설정
- Graph 상 각 Edge들은 Boolean 변수로 취급
- Hard Constraints : 반드시 만족
- Soft Constraints : 반드시 만족 필요 x

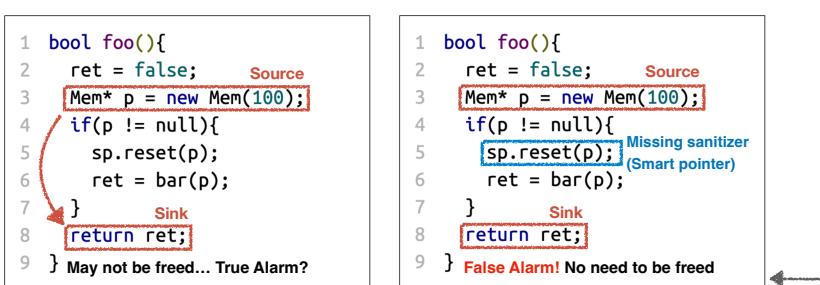
#### 1.2. 정적 오염분석 개요

Source에서 발생한 오염이 Sink에 도달 가능한지를 추적하는 경로 중심 분석

- Source : 최초 오염 발생 지점
- Sink : Source에서 진파된 오염 도달시 위험 지점
- Sanitizer : 오염 전파를 차단하는 조건 또는 연산

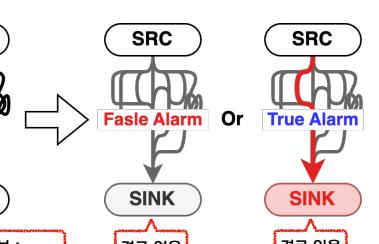
- 예1) Source = 메모리 할당 → Sink = 함수종료 Sanitizer = {메모리해제, 외부변수할당, ... }  
 예2) Source = 사용자 입력 → Sink = memcpy() Sanitizer = {입력 길이 검사, ... }

#### 1.3. 오염분석 허위경보 예시 - 메모리 누수

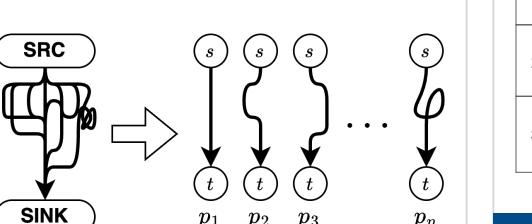


### 2. 문제

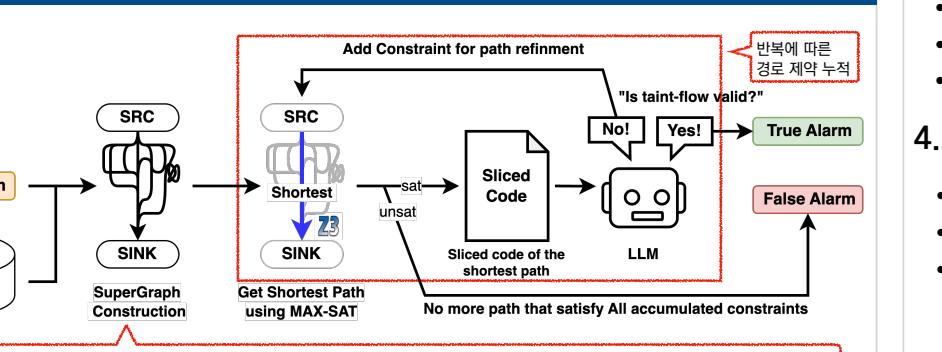
#### 2.1. 오염분석 경보 분류하기



#### 2.2. 단일 실행 경로 단위 문제로 분해

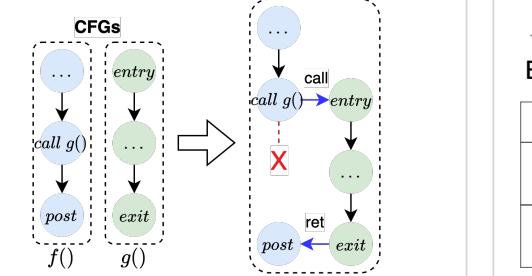


### 3. 방법(1)



#### 3.1. 슈퍼그래프 생성

- CFGs 직접사용 X
  - ▶ 각 함수마다 별도 CFG 존재
  - ▶ 함수 간 호출 관계 표현 불가
- Super Graph ✓
  - ▶ 하나의 단일 그래프에 모든 실행 경로 포함



**Key Advantage :** 허위경보 분류를 단일 실행 경로 단위의 실현 가능성 판단 문제로 분해하고, MAX-SAT + LLM 기반 점진적 경로 정제로 전역 추론 부담과 경로 오판 컨텍스트 윈도우 초과 문제를 동시에 완화