파이썬 프로그램에서 처리되지 않은 예외 분석: 디지털 포렌식 소프트웨어 중점으로

Analyzing Uncaught Exceptions in Python program: Focusing on Digital Forensic software

이서우

전공소개서

➢ 논문 요약

› 디지털 포렌식 소프트웨어

- 디지털 증거: 어떠한 사실을 증명할 수 있는 디지털 데이터
- 디지털 증거의 **누락 및 오염**이 없어야 함
- DB와의 입출력이 빈번
- **예외**가 처리되지 않을 시 **신뢰성 깨짐**

› 집합제약식 기반 분석

- 프로그램의 각 지점마다 발생할 수 있는 **예외들의 집합** 및 예외들의 집합을 구하기 위한 **제약식** 도출
- 제약식들의 **방정식의 해**를 구함 => **처리되지 않은 예외** 및 **예외의 발생 위치**

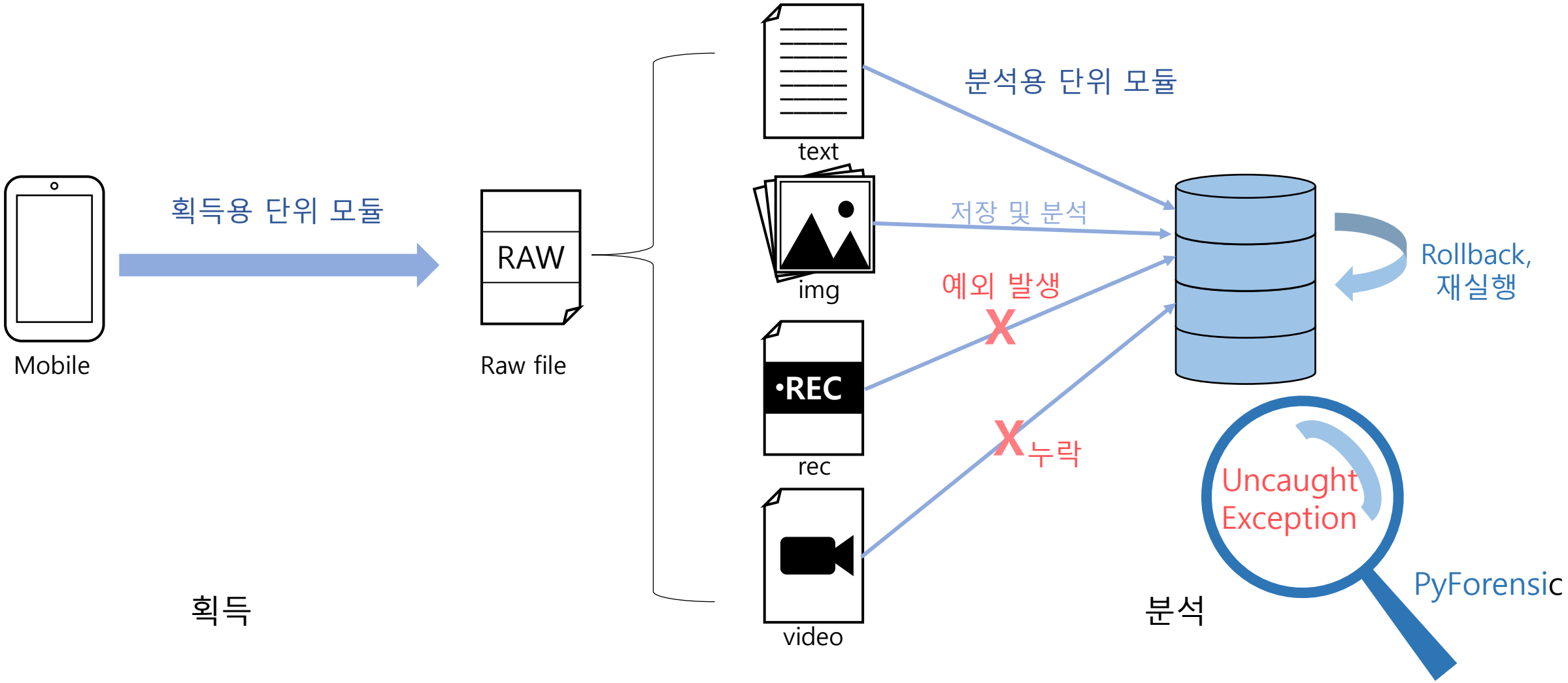
› 허위경보 및 중복된 경보 제거

- 허위 경보 제거: Pyright(마이크로소프트 사)의 타입 분석 결과 결합
- 중복된 경보 제거: 프로그램 지점들을 순회하며 중복된 경보 제거

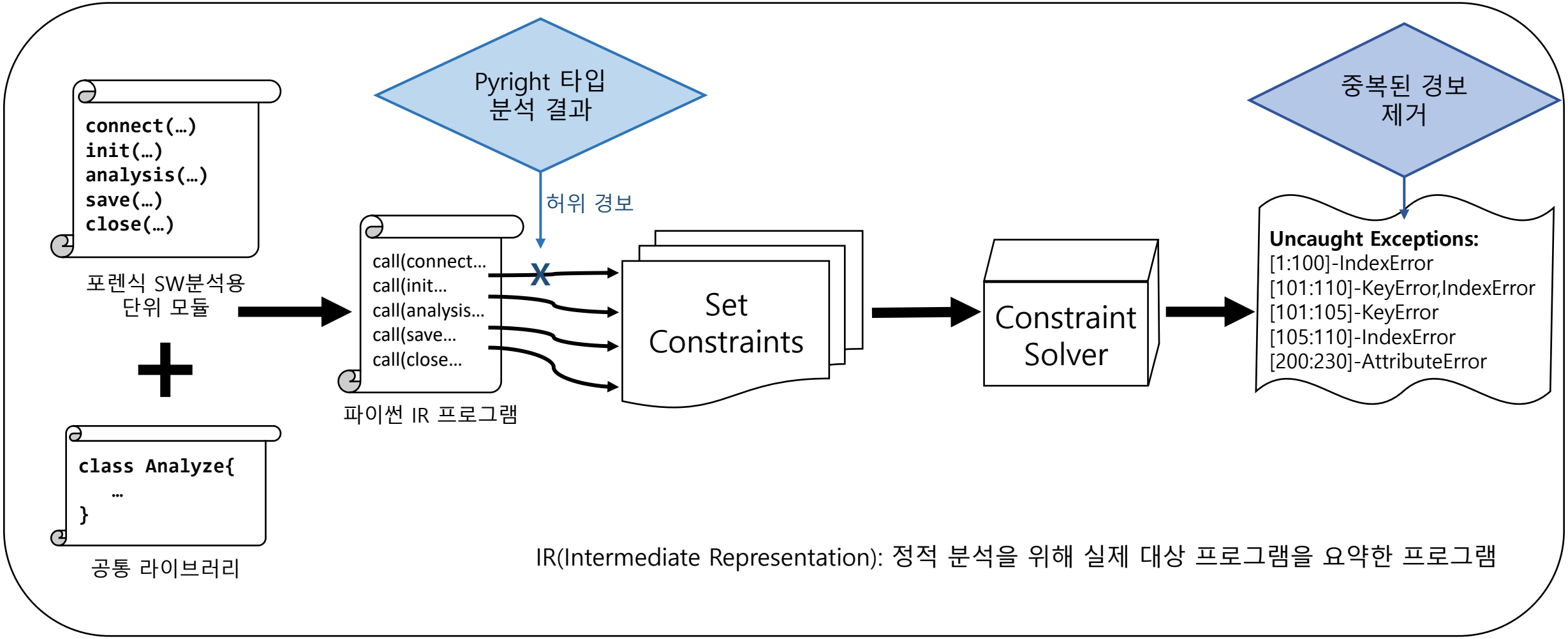
› 결과 및 의의

- 대검찰청 디지털 포렌식 소프트웨어에서 예외 발생 가능 패턴 3가지 발견
- 평균 84%의 허위 경보 및 중복된 경보 제거

➢ 연구 배경


Mobile
획득용 단위 모듈
RAW
Raw file
text
img
•REC
rec
video
분석용 단위 모듈
저장 및 분석
예외 발생
X
X
누락
Rollback,
재실행
Uncaught
Exception
PyForensic
획득
분석

➢ PyForensic 구조



➢ 집합제약식 기반 분석

- 프로그램의 각 지점마다 발생 가능한 예외들의 집합 구함
  - 프로그램의 지점마다 발생 가능한 예외(AttributeError, IndexError, KeyError, ZeroDivisionError)들의 집합에 대한 제약식 도출
  - 제약식들의 방정식의 해를 구함

L_n: n번째 줄에서 발생 가능한 예외들의 집합
L_{try}: try문에서 발생 가능한 예외들의 집합
L_{prog}: 전체 프로그램에서 발생 가능한 예외들의 집합

```
1:denom = 0
2:li=[1,2,3]
3:try:
4:    a = li[5] + 1
5:except IndexError as e:
6:    a = 10/denom
```

L_1={NameError}

L_2={NameError}

L_4={NameError, IndexError, KeyError}

L_6={NameError, ZeroDivisionError}

$L_{try}=L_4 - \{IndexError\} \cup L_6$
={NameError, KeyError, ~~IndexError~~} ∪
{NameError, ZeroDivisionError}
={NameError, ZeroDIvisionError, KeyError}

$L_{prog}=L_1 \cup L_2 \cup L_{try}$
={NameError} ∪ {NameError} ∪{NameError, ZeroDivisionError}
={NameError, ZeroDIvisionError}

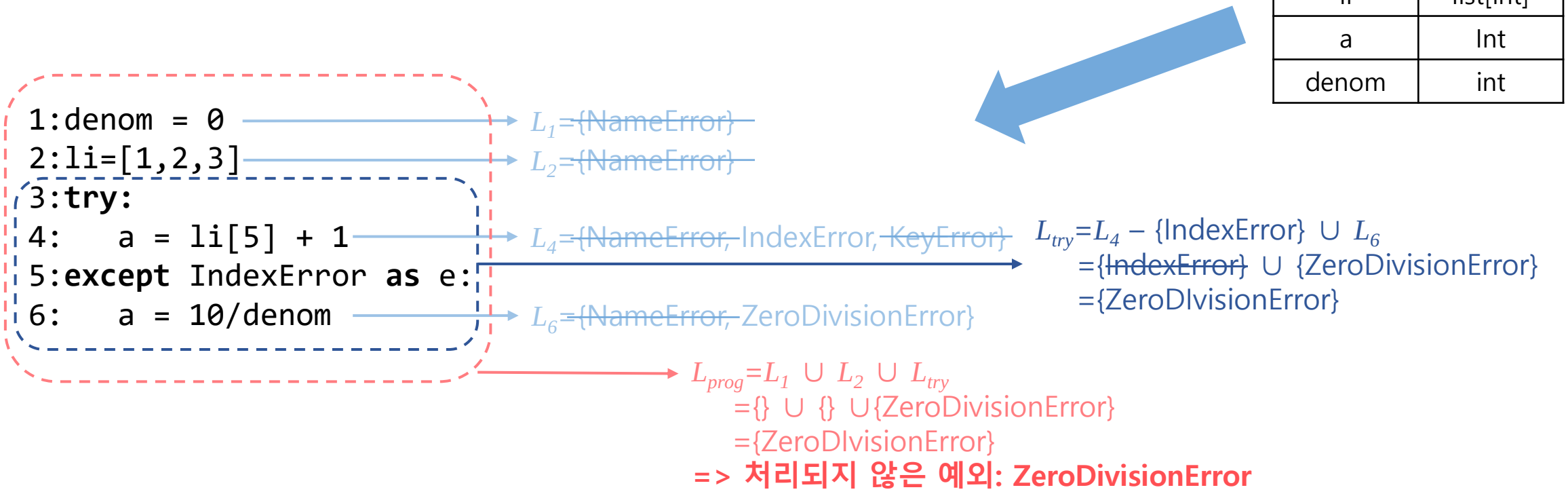
=> 처리되지 않은 예외: NameError, ZeroDivisionError

➢ 허위 경보 및 반복된 경보 제거(1/2)

- 허위경보 제거
  - Pyright를 이용하여 각 변수 및 함수의 반환 값에 대한 타입 분석
  - 타입 분석 결과에 따라 허위 경보 제거

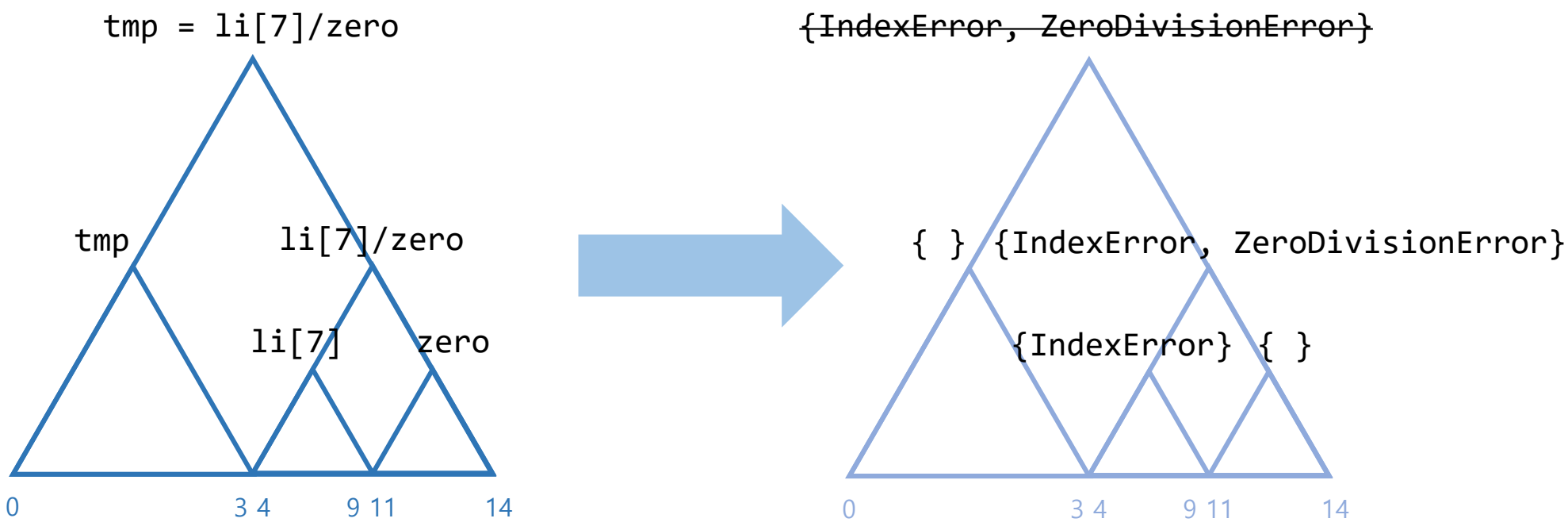
Pyright 결과

변수	타입
denom	int
li	list[int]
a	int
li	list[int]
a	Int
denom	int



➢ 허위 경보 및 반복된 경보 제거(2/2)

- 중복된 경보 제거
  - 프로그램 지점 정렬 후 순회
  - 중복되는 경보들 중 예외 발생 지점과 가장 가까운 프로그램 지점에 대한 경보만 남김



➢ 결과 및 결론

› 연구 결과

- 벤치마크
  - 대검찰청 제공 디지털 포렌식 소프트웨어 분석용 단위 모듈 9개
  - 외부 라이브러리 포함 벤치마크 1개
  - 일반 예외 발생 파이썬 프로그램 2개
- 예외 발생 가능 패턴 3개 발견
- 대검찰청 디지털 포렌식 소프트웨어 벤치마크 9개 및 외부 라이브러리 포함 벤치마크에 대하여 평균 84%의 불필요한 경보 제거

› 의의

- 파이썬 코드를 실행하기 전에 미리 발생하는 예외 및 예외의 발생 위치를 알 수 있음