공학석사학위논문

파이썬 프로그램에서의 처리되지 않은 예외 분석: 디지털 포렌식 소프트웨어 중점으로

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

2023 년 2 월

서울대학교 대학원

컴퓨터공학부

이 서 우

공학석사학위논문

파이썬 프로그램에서의 처리되지 않은 예외 분석: 디지털 포렌식 소프트웨어 중점으로

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

2023 년 2 월

서울대학교 대학원

컴퓨터공학부

이 서 우

파이썬 프로그램에서의 처리되지 않은 예외 분석: 디지털 포렌식 소프트웨어 중점으로

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

지도교수 이 광 근

이 논문을 공학석사학위논문으로 제출함

2022 년 12 월

서울대학교 대학원

컴퓨터 공학부

이 서 우

이서우의 석사학위논문을 인준함

2023 년 1 월

위 원 장	권 태 경	(인)
부위원장	이 광 근	(인)
위 원	허 충 길	(인)

요약

본 논문에서는 파이썬 프로그램 중 대검찰청의 디지털 포렌식 소프트웨어의 신뢰성을 높이기 위하여 해당 소프트웨어에서 처리되지 않은 예외를 사전에 검출하는 방안을 설계하고 성능을 측정한다. 대검찰청의 디지털 포렌식 소프트웨어는 파이썬으로 개발되고 있으며 파이썬 라이브러리 자체 내에서 발생할 수 있는 예외 상황과 포렌식 프로그램 자체의 예외 상황이 발생할 수 있는데 본 연구에서는 포렌식 프로그램 내의 오류에 집중한다.

기본적으로 집합 제약식 기반 분석 방법을 사용하여 expression과 statement으로 구분된 프로그램의 지점마다 발생할 수 있는 예외들의 집합을 구하기 위한 제약 조건을 정의하여 이들간 방정식의 해를 구한다. 이 과정에서 발생할 수 있는 허위 경보를 줄이기 위하여 Pyright의 타입 분석 결과를 결합하여 리스트나 딕셔너리 등 타입별로 발생할 수 없는 KeyError나 IndexError 등의 예외들을 제거하여 허위 경보를 줄인다. 또한, 프로그램의 구성을 프로그램 지점들의 포함 관계에 따라 트리 형태로 나타내고 전위 순회를 하면서 중복되는 허위 경보를 제거한다.

개발한 정적 분석기를 대검에서 제공한 9 개의 벤치마크에서 수행시킨 결과 대검찰청에서 찾지 못한 KeyError, IndexError, ZeroDivisionError 등 3 가지 패턴에 속하는 예외 발생 지점 10여 개를 찾을 수 있었다. 또한, 9 개의 벤치마크에 대하여 평균 84%, 최대 89%의 허위 경보를 제거하였다. 이와 아울러 반복되는 허위 경보를 부모 자식 관계를 활용하여 더욱 허위 경보를 줄일 수 있는 여지도 발견하였다.

주요어: 처리되지 않은 예외, 정적 분석, 디지털 포렌식 소프트웨어, 파이썬 프로그램

학번: 2021-26902

목차

그림 목차

표 목차

제 1 장 서론

본 논문에서는 대검찰정에서 파이썬으로 개발한 디지털 포렌식 소프트웨어 단위 모듈을 중점으로하여 파이썬 프로그램 내 처리가 되지 않은 예외를 분석하는 방법 및 분석 결과의 허위 경보를 줄이는 방법을 설계한다. 최근 지속적인 디지털 매체들의 출현은 PC가 대부분이었던 시대에서 벗어나 PC, 스마트폰, 사물 인터넷(IoT) 등 다양한 디지털 매체들이 활용되도록 한다. 디지털 매체의 다양화에 의해 각 매체에서의 디지털 증가 확보 능력이 중요해졌고 이를 담당하는 디지털 포렌식 소프트웨어 또한 다양해지고 있다.

여기에서 디지털 증거는 법정에서 증거로 채택 가능한 PC 등 디지털 매체에 저장 또는 전송되는 데이터를 의미한다. 즉, 컴퓨터의 하드 디스크나 USB나 SSD에 저장되어 있는 사진이나 녹음 데이터, 네트워크 상에서 주고 받은 문자나 파일들 모두 디지털 증거가 될 수 있다. 이와 같이 일상 생활에서 디지털 매체를 이용하며 남긴 모든 기록들이 디지털 증거로 사용될 수 있고 디지털 매체에서 얻을 수 있는 정보의 양도 증가함에 따라 보다 많은 디지털 증거를 확보할 수 있다.

일상 생활 속의 데이터들이 증거로 채택이 되면서 범죄의 사실 여부나 어떠한 사건의 발생 여부를 밝혀내기가 용이해졌다. 이에 따라, 검찰청은 2008년 디지털포렌식센터(DFC; Digital Forensic Center)와 2012년 국가디지털포렌식센터(NDFC; National Digital Forensic Center)를 설립하였고 경찰이 디지털 포렌식을 수사에 이용한 횟수는 2017년 34,541건에서 2020년 63,034건으로 매년 증가하여 왔으며 앞으로 증가할 것으로 전망되고 있다[11]. 그러나 디지털 매체마다 수집할 수 있는 디지털 증거들의 속성이 다르기 때문에 디지털 증거를 확보하는 방법 또한 달라질 수 밖에 없다. 따라서 새로운 디지털 매체에서 디지털 증거를 확보하기 위해서는 그 디지털 매체에 맞는 새로운 디지털 포렌식 소프트웨어를 개발해야 한다[13].

디지털 증거의 확보의 필요성은 포렌식 소프트웨어 자체의 오류를 정확하게 분석하는 메커니즘을 필수적인 요소로 만들고 있다. 우선적으로, 디지털 증거는 진정성을 갖고 있어야하는데, 여기에서 진정성이란 무결성, 동일성, 신뢰성을 모두 포괄하는 개념이다. 먼저, 디지털 증거의 무결성이란, 디지털 매체로부터 데이터를 수집하여 법원에 증거로 제출되기까지 데이터의 훼손 및 조작이 없어야 함을 의미한다. 둘째, 동일성은 디지털 매체로부터 수집된 데이터의 내용과 법원에 제출된 디지털 증거의 내용이 다르지 않아야 함을 의미한다. 셋째, 신뢰성이란 디지털 증거를 획득하고 분석하는 소프트웨어의 신뢰성 및 정확성이 보장되어야하고 데이터 수집 및 분석, 출력을 하는 전 과정이 전문적인 기술을 가진 사람에 의해 수행되어야 함을 의미한다[10, 12].

결국 디지털 포렌식은 수많은 디지털 데이터들 속에서 수사에 필요하고 증거로 채택 가능한 디지털 증거들을 수집한다. 이때 디지털 포렌식 소프트웨어의 오류로 인해 발견하지 못하는 디지털 증거가 없어야 하고 발견된 데이터 증거들에는 오류가 없어야 한다. 이러한 특성들 때문에 신뢰성이 있는 디지털 증거를 확보하는 디지털 포렌식 소프트웨어 또한 신뢰성을 갖고 있어야 한다[3]. 이에 따라 디지털 포렌식 소프트웨어의 정확성을 위한 요구사항들을 정의하고 디지털 포렌식 소프트웨어를 실행한 결과를 통해 정확성을 확인하는 연구가 이루어졌다[14]. 그러나, 디지털 포렌식 소프트웨어의 정확성을 사람이 일일히 각각의 포렌식 소프트웨어의 오류를 분석을 하기는 어렵다. 결국 자동으로 디지털 포렌식 소프트웨어의 오류를 검출해주는 분석기가 필요하다.

디지털 포렌식 소프트웨어들은 구조적으로 공통으로 포함하여야 하는 필수 부분이 있어 동일한 성격의 오류가 발생할 수 있고 이러한 일반적인 오류들에 대해서는 자동으로 분석하는 기술을 개발할 수 있다. 대부분의 디지털 포렌식 소프트웨어들은 공통적으로 크게 세 부분으로 구성되어 있다. 첫째, 디지털 매체에서 데이터들을 추출하는 부분, 둘째, 추출된 데이터를 DB등의 데이터 저장 매체에 저장하는 부분, 셋째, 저장된 데이터들을 분석하면서 디지털 증거들을 확보하는 부분 등이다. 먼저, 데이터를 추출하는 과정에서는 NULL 값이나 쓰레기 값이 수

집되어 디지털 증거를 오염시키는 오류가 발생할 수 있다. 또, 수집된 데이터를 저장 매체에 저장하거나 저장된 데이터를 가져올 때에도 데이터 매체와의 연결에 문제가 생겨 데이터 일부가 유실될 수도 있다. 그리고, 데이터를 분석하는 과정에서도 예외가 정확히 처리되지 못해 프로그램의 수행이 멈추거나 데이터들이 누락 및 훼손될 위험이 있다. 이상의 디지털 포렌식 소프트웨어들에서 공통적으로 발생 가능한 오류들에 대해서는 분석 기술들을 개발할 수 있다.

본 논문에서는 디지털 포렌식 소프트웨어에서 획득한 데이터들을 분석하는 과정에서 처리되지 않은 예외를 분석하는 기술에 대한 연구를 진행한다. 대검찰청에서 개발한 디지털 포렌식 소프트웨어는 모바일폰 상에 저장된 디지털 증거들을 수집하는 소프트웨어로서 디지털 매체로부터 데이터를 수집하고 DB에 저장하는 획득 단계와 DB에 저장된 데이터를 가져와 분석하는 분석 단계 등 2 단계로 구성되어 있다. 본 논문에서는 두 단계 중 분석 단계의 프로그램을 대상으로 분석을 진행한다.

대검찰청의 디지털 포렌식 소프트웨어의 분석 단계에서는 DB로부터 데이터의 조회, 데이터의 분석, 분석 결과의 DB 저장 등의 작업이 반복적으로 수행되는데 이 과정에서 처리되지 않은 예외를 분석하는 기술이 필요하다. 이 3 단계를 거치면서 하나의 어플리케이션에 대한 디지털 증거를 확보하는 프로그램 하나하나를 분석용 단위 모듈이라고 한다. 이 단위 모듈들에 처리되지 않은 예외가 있으면 디지털 포렌식 작업 중에 프로그램의 수행이 중단되어 데이터의 누락이 발생할 수도 있고 DB와의 연결 관리에 문제가 생겨 데이터가 훼손 및 조작되면서 디지털 증거의 진정성을 해칠 수 있다. 또한, 롤백에 의해 DB에 다시 연결하여 연결 문제를 해결하거나 소프트웨어를 다시 실행할지라도 디지털 증거 채집 중 발생하는 예외가 여전히 남아있기 때문에 프로그램이 비정상 중단되고 디지털 증거가 누락 및 오염될 위험이 여전히 남아있다. 따라서 처리되지 않은 예외가 발생할 가능성을 실행하기 전에 미리 확인할 필요가 있다.

본 연구에서는 처리되지 않은 예외를 분석하기 위해 집합 제약식 기반 분석 기법(set constraint-based analysis)을 이용한다. 집합 제약식 분석 기법은 프로그

램의 각 지점마다 분석하고자 하는 객체의 집합을 구한 다음 각 프로그램 지점마다 만든 집합들 사이의 제약식을 세우고 이 제약식들의 방정식을 해의 집합이 변하지 않을 때까지 풀면서 프로그램을 분석하는 방법이다 [2, 4]. 이 분석 기법을 처리되지 않는 예외 분석에 적용하면 프로그램의 각 부분마다 발생할 수 있는 예외들의 집합을 구하고 이 예외 집합들의 포함 관계에 대한 제약식들을 세울 수 있게 되며 이 제약식들의 방정식의 해를 구하면 프로그램 내에서 처리되지 않은 예외를 분석할 수 있다.

연구 진행 중 처리되지 않은 예외 분석 결과에 허위 경보 및 반복된 경보가 다수 포함된다는 문제와 외부 라이브러리에 포함된 함수를 호출할 때 이 함수에서 발생 가능한 예외를 놓치는 문제를 파악하였다. 또, 대검찰청의 디지털 포렌식 소프트웨어의 구현 언어인 파이썬의 특성으로 인해 분석 결과의 정확도가 떨어지고 분석 결과에 허위 경보가 다수 포함된다는 문제도 발견된다. SML 프로그램에서 처리되지 않은 예외를 효율적으로 분석하는 연구[8, 9]와 Java 프로그램에서의 예외를 분석 하는 연구[6]와 대조적으로 파이썬은 타입 안전성이 보장되지 않으며 프로그램의 구조만으로는 함수를 호출하는 객체의 타입을 알기 어렵다. 또한, 대검찰청에서 자체로 작성한 외부 라이브러리의 경우 분석을 위한 코드를 분석 대상 코드에 추가하지 않고서는 외부 라이브러리에 정의된 변수 및 함수들에 접근이 불가능하다.

허위 경보 및 반복된 경보가 다수 발생하는 문제와 외부 라이브러리 함수에서 발생하는 예외를 놓치는 문제를 해결하기 위해 다음과 같은 메카니즘을 설계한다. 허위 경보를 줄이기 위해서는 타입 좁히기(type narrowing)을 활용하여 파이썬에서 타입을 근사해서 분석할 수 있는 정적 타입 분석기인 Pyright[1]의 분석 결과에 포함된 변수 정의와 객체 타입 정보를 활용한다. 또한, 외부 라이브러리의 함수 내에서 발생 가능한 예외를 놓치는 문제는 외부 라이브러리까지 분석도 가능하도록 예외 분석기의 기능을 확장한다.

본 논문의 구성은 다음과 같다. 1장에서 해결하고자 하는 문제를 정의한 후 2장에서는 분석 대상인 디지털 포렌식 소프트웨어에 대한 설명 및 연구 배경에

대해 기술한다. 3장에서는 처리되지 않은 예외를 분석하기 위해 사용한 집합 제약식 기반 분석 기법을 제시하고 분석 과정에서 적용한 분석 규칙들에 대해 구체적으로 설명한다. 4장에서는 본 연구에서 허위 경보를 줄이기 위해 사용한 기법들을 설계하고 수행과정을 기술한다. 5장에서는 본 연구를 통해 처리되지 않은 예외를 발견한 결과와 허위 경보를 줄인 결과를 보인다. 6장에서는 본 연구의 한계 및 허위 경보를 줄이기 위해 추후 진행될 수 있는 연구를 소개하고 본 논문의 결론을 도출한다.

제 2 장 연구배경

2.1 디지털 포렌식 소프트웨어

이번 장에서는 본 연구의 배경을 소개한다. 2.1에서는 디지털 증거 및 디지털 포렌식에 대한 기본 지식을 설명하고, 2.2에서는 분석의 대상인 대검찰청에서 개발중인 디지털포렌식 소프트웨어에 대해 설명한다.

2.1.1 디지털 증거 및 디지털 포렌식

디지털 포렌식은 디지털 매체로부터 수집한 데이터들을 분석하여 법정에서 어떤 사실을 증명하는 증거가 되는 데이터의 모음, 즉 디지털 증거들을 확보하는 과정을 의미한다. 여기에서 법정에서 어떠한 사실을 증명하는 증거가 되기 위해서는 DNA, 은행입출금내역, 족적, 혈흔, 흉기 등의 물리적인 증거과 마찬가지로 어떠한 조작이 가해지지 않은 상태에서 위법 혹은 사실을 증명할 수 있는 근거가 될 수 있어야 한다. 또한, 전문 지식을 지닌 주체에 의해 증거들이 확보되고 분석되어야 하며 이 과정 중에 증거의 누락이나 오류로 인해 잘못된 사실을 증명하지 않아야 한다. 따라서 디지털 증거를 확보하는 과정은 엄격한 절차에 따라 이루어져야 하고 디지털 증거를 확보하는 프로그램인 디지털 포렌식 소프트웨어에는 오류가 없어야 한다.

2.1.2 분석 대상 소프트웨어

대검찰청에서 개발한 모바일용 디지털 포렌식 소프트웨어 MFA(Mobile Forensic Analyzer)는 그림 2.1과 같이 두 단계로 나누어 디지털 증거들을 확보한다. 첫 단계인 획득 단계에서는 디지털 매체 로부터 데이터를 수집하고 이 데이터들을 압축되거나 가공되지 않은 이미지 데이터 파일(raw file) 형태로 DB에 저장한다. 그 다음 분석 단계에서는 저장된 이미지 데이터 파일로부터 분석의 단위인 어플

리케이션마다의 데이터를 분석하여 디지털 증거들을 확보하고 이를 수사용 DB에 저장한다. 따라서 디지털 포렌식 전 과정에 서 DB와의 통신이 빈번하게 일어난다. 획득과 분석 두 단계 모두 처리되지 않은 예외가 있어 프로그램이 비정상적으로 종료되면 디지털 증거들이 누락되거나 쓰레기 값이나 NULL 값이 코드 중간에 나타나면서 디지털 증거들이 오염될 가능성이 있다. 특히, 분석 단계에서 수집한 데이터들을 분석하여 법정에 제출할 디지털 증거를 확보하는 과정 및 확보한 디지털 증거들을 분석하고 DB에 저장하는 도중에 프로그램이 비정상적으로 종료되면 디지털 증거들이 누락되거나 오염될 수 있다.

또한, 프로그램 수행 과정에서 DB와의 연결을 시도했을 때 연결과정에서 문제가 발생한다면 DB와의 연결이 실패하고 뒤이어 프로그램이 종료되면서 디지털 증거의 누락이 발생할 수 있다. DB와의 연결에 성공했다고 해도 데이터들을 분석하고 DB에 저장하는 과정 중에 예외가 발생하는데 DB와의 연결이 종료 절차를 거치지 못하면 DB에 저장된 디지털 증거들에 쓰레기 값이 추가됨에 따라 디지털 증거들이 오염되고 법적 증거로서의 가치가 떨어질 수 있다. 더욱이, 이후에 다시 DB에 연결하는데 장애가 생겨 DB와의 연결이 안되거나 연결에 많은 시간이 걸릴 수 있다. 이와 같은 이유로 분석 단계에서 처리되지 않은 예외를 분석하는 것은 중요하다. 본 연구에서는 하나의 어플리케이션에 대하여 데이터 증거를 분석하고 DB에 저장하는 작업이 일어나는 분석용 단위 모듈을 하나의 분석단위로 설정하고 이 단위에서 처리되지 않는 예외를 파악한다.

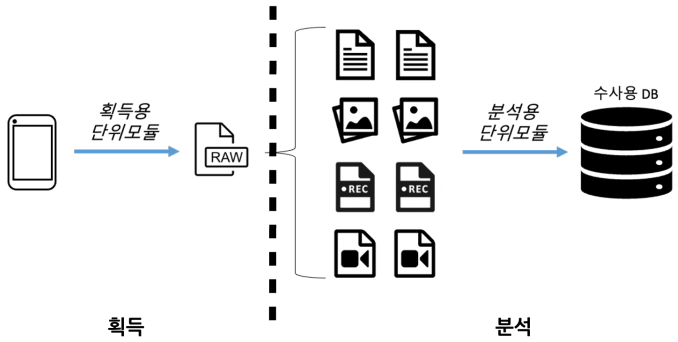


그림 2.1: 대검찰청 디지털 포렌식 소프트웨어 구조

제 3 장 집합 제약식 기반 분석

이번 장에서는 처리되지 않은 예외를 정의하고 본 연구에서 사용하고자 하는 집합 제약식 분석(set constraint-based analysis) 기법에 대해 설명한다. 3.1에서 처리되지 않은 예외의 기본적인 특성과 그 영향에 대해 기술하고 3.2에서는 대상 포렌식 소프트웨어의 구현 언어인 파이썬의 요약된 문법을 설명하며 처리되지 않은 예외 분석을 위한 집합 제약식 기반 분석에 대한 설명과 처리되지 않은 예외를 분석하기 위한 분석 규칙들을 소개한다.

3.1 처리되지 않은 예외

처리되지 않은 예외란 프로그램 내에서 어떠한 오류로 예외가 발생하여 프로그램이 비정상 종료가 되는 상황을 의미한다. 구체적으로 말하면, 프로그램 내에서 0으로 나누거나(divide by zero), 할당된 범위를 벗어난 지점에 접근하거나(out of bound), 정의되지 않은 변수나 객체에 접근하거나(undefined) 하는 등의 예외들이 try문 등으로 제대로 처리가 되지 않아 프로그램이 비정상 종료하는 상황 등이다. 3.1.1과 3.1.2에서는 각각 예외를 처리하지 않는 예시 코드와 try 문으로 예외가 제대로 처리되지 않은 예시 코드를 보인다.

3.1.1 예외 처리가 되지 않은 경우

그림 3.1은 DB에 저장된 데이터를 출력하는 예시 코드이다. 이 예시 코드에서는 DB에 있는 kakao 테이블로부터 name 열에 있는 자료들 중 7번째에 있는 데이터를 출력한다. 그러나 kakao 테이블의 name 열에 있는 데이터들의 수가 7개보다 작으면 7번째 줄에서 파이썬에서 배열이 할당된 범위에서 벗어난 영역을 접근하는 예외(out of bound)인 IndexError가 발생할 수 있다. 또한, 1번째 줄에서 DB와 연결을 시도할 때 연결에 실패하여 pymysql.err.OperationalError가

발생할 수 있다. 따라서 이 예시 코드에서는 DB와의 연결에 실패하고 프로그램이 비정상적으로 종료되거나 DB와의 연결이 끊기기 전에 프로그램이 비정상적으로 종료되어 DB와의 연결이 정확하게 종료되지 않을 위험이 있다.

```
1:  db = pymysql.connect(...)
2:  cur = db.cursor()
3:  sql = "select name from kakao"
4:  vals = (name,)
5:  cur.execute(sql,vals)
6:  row = cur.fetchone()
7:  print(row[6])
8:  db.close()
```

그림 3.1: 예외가 처리되지 않은 프로그램 예시

3.1.2 예외 처리를 포함하지만 정확히 처리하지 못한 경우

그림 3.2은 그림 3.1의 예시 코드에서 try문으로 예외 처리를 해준 경우에 대한 예시 코드이다. 그림 3.2에서는 1번째 줄과 9에서 10번째 줄에 try와 except문을 추가하여 DB와의 연결에 실패할 경우에 대한 예외 처리를 정의하였다. 따라서 이 예시 코드에서는 DB와의 연결에 실패하여도 프로그램이 비정상적으로 종료되지 않는다. 그러나 여전히 예외가 제대로 처리되지 않아 비정상적으로 프로그램이 종료되고 DB와의 연결이 끊어지지 않을 위험이 있다. 여전히 8번째 줄에서 발생 가능한 IndexError에 대한 예외 처리가 이루어지지 않아 프로그램이 비정상적으로 종료될 수 있다.

```
1:  try:
2:        db = pymysql.connect(...)
3:        cur = db.cursor()
4:        sql = "select name from kakao"
5:        vals = (name,)
6:        cur.execute(sql,vals)
7:        row = cur.fetchone()
8:        print(row[6])
9:  except pymysql.err.OperationalError:
10:       print("Connection␣failed: Kakao [name]")
11: db.close
```

그림 3.2: 예외가 처리된 프로그램 예시

3.2 처리되지 않은 예외 분석

3.2.1 분석 대상 언어

본 연구의 분석 대상인 대검찰청의 디지털 포렌식 소프트웨어의 분석용 단위 모듈은 파이썬 언어로 작성되어 있다. 그러나 본 연구에서는 분석용 모듈의 처리되지 않은 예외를 분석하기 위해 분석용 단위 모듈을 기반으로 요약된 문법(abstract syntax)을 가지는 파이썬 IR(Intermediate Representation)을 정의하였다. 이 요약 문법은 PyTea[5]의 IR을 확장하여 정의하였으며 파이썬의 기본적인 문법의 틀을 지키면서 코드 작성을 쉽게 해주면서 분석에 방해가 되는 문법인 설탕구조(syntactic sugar)를 제거하고 분석의 주 대상인 예외 처리가 잘 드러나도록 한다.

하나의 프로그램은 실행할 수 있는 가장 작은 코드 조각인 statement로 구성되어 있으며 statement에는 pass문, expression 실행문, 할당문, if문, for문, return문, 정의문, 함수 정의문, try문, raise문이 있다. 값으로 표현될 수 있는 코드인 expression에는 정수, 실수, 문자열, Boolean, None, list, dictionary, tuple등의

기본적인 타입의 값과 예외를 포함한 객체(object), 객체의 속성(attribute), subscript, Boolean 연산 및 사칙연산, 비교 연산, 비트 연산 등의 연산들의 계산 값으로 구성되어 있다. 정의된 요약 문법의 전체적인 구조는 부록의 A.1 문법 항목에 첨부하였다. 기술된 문법에서 *는 해당 문법 부품 (expr, stmt, Undef 등)이 하나 이상 올 수 있다는 의미이다.

3.2.2 집합 제약식 기반 분석

본 연구에서는 집합 제약식 기반 분석 기법을 활용하여 분석용 단위 모듈에서 처리되지 않은 예외를 분석한다. 집합 제약식 기반 분석 기볍은 각 statement 혹은 expression마다 프로그램의 지점마다 분석하고자 하는 특성들의 집합과 이 집합들 사이의 관계를 나타내는 제약식들을 세우고 이 제약식들의 방정식의 해를 구하여 분석을 하는 기법이다. 이를 처리되지 않은 예외 분석에 적용하려면 프로그램의 각 지점마다 발생 가능한 예외들의 집합들을 구하기 위해 이 집합들 사이의 포함관계를 나타내는 제약식들을 정의해야 한다. 그 다음 이 제약식들의 방정식의 해를 구하여 처리되지 않은 예외 및 처리되지 않은 예외가 발생한 프로그램의 지점을 구한다.

그림 3.3의 예시코드에서 집합 제약식 기반 분석 기법을 활용하여 처리되지 않는 예외를 분석하는 과정은 다음과 같다. n번째 줄에서 발생 가능한 예외들의 집합을 L_n이라고 하고 try문에서 발생 가능한 예외들의 집합을 L_{try}, 전체 프로그램에서 발생 가능한 예외를 L_{prog}라고 나타낼 때 1번째와 2번째 줄에서는 예외가 발생하지 않아 발생 가능한 예외들의 집합 L_1과 L_2는 $\emptyset$이다. 4번째 줄에서 발생 가능한 예외들의 집합 L_4는 {ZeroDivision}이다. 6번째 줄에서는 예외가 발생하지 않아 L_6은 $emptyset$이고 7번째 줄에서 발생 가능한 예외의 집합 L_7은 {NameError}이다. 이 발생 가능한 예외들 사이의 포함관계를 기반으로 제약식으로 나타내면 try문(3번째 7번째 줄)에서 발생 가능한 예외들의 집합을 구하는 제약식은 $L_{try} = L_4 - \{\text{ZeroDivisionError}\} \cup L_6 \cup L_7$ 이다. 그리고 전체 프로그램에서 발생 가능한 예외 즉, 처리되지 않은 예외를 계산하는 제약식은

$L_{prog} = L_1 \cup L_2 \cup L_{try}$이다. 따라서 다음과 같은 제약식들의 방정식을 세울 수 있다. $L_1 = \emptyset$, $L_2 = \emptyset$, L_4 ={ZeroDivisionError}, $L_6 = \emptyset$, L_7 ={NameError}, $L_{try} = L_4 -$ {ZeroDivisionError} $\cup L_6 \cup L_7$, $L_{prog} = L_1 \cup L_2 \cup L_{try}$ 이들의 방정식을 풀어 전체 프로그램에서 발생 가능한 예외들의 집합을 구하면 {NameError}이다.

```
1:  zero = 0
2:  li = [1,2,3,4,5]
3:  try:
4:      tmp = li[7] / zero
5:  except ZeroDivisionError as e:
6:      n = e.args
7:      raise NameError
```

그림 3.3: 처리되지 않은 예외 프로그램 예시

3.2.3 집합 제약식 생성 규칙

집합 제약식 기반 분석을 이용해 처리되지 않는 예외를 분석하기 위해서는 각 프로그램의 지점마다 발생 가능한 예외들의 집합을 구할 수 있는 제약식의 생성 규칙을 정의하여야 한다. 또한, 본 연구의 분석 예외이자 집합 제약식 생성 규칙에 나타나는 예외들은 그림 3.4과 같다. 본 연구에서는 분석 대상에서 TypeError가 발생하지 않는 타입 안전성이 보장된다고(type safe) 가정한다. 제약식 생성 규칙을 표현하기 위해서 아래와 같이 기호를 정의한다.

X_e : 프로그램 식 e가 가질 수 있는 예외 집합 변수
X_s : 프로그램 구문 s가 가질 수 있는 예외 집합 변수
C : 제약식 집합
E_c : 예외 클래스(class) c

그림 3.5와 그림 3.6은 각각 expression에서 발생 가능한 예외들의 집합을 구하는 제약식들의 생성 규칙과 statement에서 발생 가능한 예외들의 집합을 구하는

제약식 생성 규칙이다. 이 집합 제약식 생성 규칙은 안전한(sound)한 분석을 위하여 규칙들을 느슨하게 정의하였다. 그림 3.5에서 $\triangleright expr{:}C$은 expression $expr$에서 발생 가능한 예외들의 집합들을 구하는 제약식들의 집합은 C이라는 의미이고 그림 3.6도 마찬가지로 $\triangleright stmt{:}C$의 의미는 statement $stmt$에서 발생 가능한 예외들의 집합들을 구하는 제약식들의 집합은 C이라는 것이다.

E_{name}	:	잘못된 변수 이름 예외
$E_{atrribute}$	:	잘못된 속성 이름 예외
E_{index}	:	잘못된 인덱스 예외
E_{key}	:	잘못된 키 예외
$E_{zeroDiv}$	:	0으로 나누기 예외

그림 3.4: 예외 클래스

expression에 대한 집합 제약식 생성 규칙은 다음과 같다. 정수 또는 실수, 문자열 및 불리안(Boolean) 등의 상수(constant) 식 자체에서는 예외가 발생하지 않고 상수에서 생성되는 제약식들의 집합은 공집합($\emptyset$)이다. 변수나 함수, 객체를 지칭하는 식별자(identifier) 식에서는 정의되지 않은 식별자를 사용할 때 나타나는 이름 예외(name exception)가 발생할 수 있다. 객체의 속성값(attribute)에 접근할 때에는 잘못된 속성 이름 예외(attribute exception) 및 객체의 프로그램 식(e_1)에서 발생할 수 있는 예외가 발생할 수 있다. 정의되지 않은 식별자를 사용할 때 나타나는 이름 예외가 발생할 수 있다. 그리고 속성값에 접근할 때의 제약식들의 집합은 객체의 프로그램 식(e_1)에서 생성되는 제약식 집합(C_1)의 원소들을 포함한다. 리스트(list), 딕셔너리(dictionary), 문자열(string), 튜플(tuple) 등 인덱스로 객체 내 원소에 접근 가능한 객체(subscriptable object)에 인덱스로 접근할 때에는 잘못된 인덱스 예외 (index exception) 또는 딕셔너리 일 때의 잘못된 키 예외 (key exception)및 객체와 인덱스의 프로그램 식(e_1, e_2)에서 발생할 수 있는 예외가 발생할 수 있다. 이때 생성되는 제약식들의 집합은 객체와 인덱스의 프로그램 식(e_1, e_2)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다.

리스트 및 튜플의 제약식 생성식에서는 각 원소들의 프로그램 식(e_1, e_2, e_3)에서 발생할 수 있는 예외가 발생할 수 있다. 이때 생성되는 집합 제약식들의 집합은 각 원소들의 프로그램 식(e_1, e_2, e_3)에서 생성되는 제약식 집합(C_1, C_2, C_3)의 원소들을 포함한다. 딕셔너리 생성식에서는 각 키 및 값들의 프로그램 식(e_1, e_2, e_3, e_4)에서 발생할 수 있는 예외가 발생할 수 있다. 딕셔너리에 대하여 생성되는 제약식 집합은 각 키, 값들의 프로그램 식(e_1, e_2, e_3, e_4)에서 생성되는 제약식 집합(C_1, C_2, C_3, C_4)의 원소들을 포함한다. 이항 연산에서는 피연산자들의 프로그램 식(e_1, e_2)에서 발생할 수 있는 예외와 나누기, 모듈로 연산일 때는 0으로 나누는 예외(zero division exception)가 발생할 수 있다. 이때의 제약식 집합은 피연산자들의 프로그램 식(e_1, e_2)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다. 단항 연산에서는 피연산자의 프로그램 식(e_1)에서 발생할 수 있는 예외가 발생할 수 있다. 단항 연산에서 생성되는 제약식들의 집합은 피연산자의 프로그램 식(e_1)에서 생성되는 제약식 집합의 원소들을(C_1) 포함한다. 함수 호출문에서는 함수의 객체 프로그램 식(e_1), 이름이 f인 모든 함수의 본문(function body, S_i), 그리고 함수 인자 프로그램 식(e_2)에서 발생할 수 있는 예외가 발생할 수 있다. 이때 생성되는 제약식들의 집합은 함수 객체 프로그램 식(e_1), 이름이 f인 모든 함수의 본문(S_i), 그리고 함수 인자 프로그램 식(e_2)에서 생성되는 제약식 집합(C_{e_1}, $\cup_{C_i}$, C_{e_2})의 원소들을 포함한다. 일단, 라이브러리 함수 호출에서는 예외가 발생하지 않는다고 가정한다. 여기에서 라이브러리 함수란 list()등 파이썬 빌트인 라이브러리 함수를 의미한다. 예외 클래스는 분석을 위해 따로 정의한다. 이 expression은 식별자가 가리키는 값이 예외 클래스인 경우이다.

각 statement에서 발생 가능한 예외들의 집합을 구할 수 있는 제약식들의 집합을 생성하는 규칙은 다음과 같다. 실행 흐름 제어문인 break, pass, continue 자체에서는 예외가 발생하지 않는다. 그리고 생성되는 제약식 집합은 공집합($\emptyset$)이다. expression 자체를 실행하는 expression 실행문에서는 프로그램 식(e_1)을 실행하는 구문에서는 해당 식에서 발생할 수 있는 예외가 그대로 발생할 수 있다. 그리고 이때 생성되는 제약식의 집합은 프로그램 식(e_1)에서 생성되는 제약식 집합

(C_1)의 원소들을 포함한다. 구문 연결문(sequence statement)에서는 연결되는 두 개의 구문(S_1, S_2)에서 발생할 수 있는 예외가 발생할 수 있다. 그리고 이 실행문에서의 집합 제약식들은 두 개의 구문(S_1, S_2)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다. 할당문에서는 할당되는 변수의 프로그램 식(e_1)과 할당되는 값의 프로그램 식(e_2)에서 발생할 수 있는 예외 모두가 포함된다. 할당문에서의 제약식들의 집합은 프로그램 식(e_1, e_2)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다.

if문에서는 조건식(e_1)과 조건이 만족할 때 실행되어야 할 구문(S_1), 그리고 만족하지 않을 때 실행되어야 할 구문(S_2)에서 발생할 수 있는 예외가 발생할 수 있다. if문에서의 제약식들의 집합은 조건식(e_1)과 각 구문(S_1, S_2)에서 생성되는 제약식 집합(C_1, C_2, C_3)의 원소들을 포함한다. for문에서는 각 원소에 대해 반복 실행될 객체의 프로그램 식(e_1)과 실행 구문(S_1)에서 발생할 수 있는 예외가 발생할 수 있다. 그리고 for문의 제약식들의 집합은 프로그램 식(e_1)과 구문(S_1)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다. 파이썬은 변수에 값을 할당하는 동시에 변수가 선언되지만 본 연구에서는 예외 분석을 위해 추가적인 선언 할당문(let문)을 정의한다.

선언 할당문에서는 할당되는 값의 프로그램 식(e_1)과 변수가 선언 할당된 후 실행되는 구문(S_1)에서 발생할 수 있는 예외가 발생할 수 있다. 실행 할당문에서의 제약식들의 집합은 프로그램 식(e_1)과 구문(S_1)에서 생성되는 제약식 집합(C_1, C_2)의 원소들을 포함한다. 할당값이 없는 선언 할당문에서는 변수가 선언 할당된 후 실행되는 구문(S_1)에서 발생할 수 있는 예외가 발생할 수 있다. 또한, 제약식들의 집합은 구문(S_1)에서 생성되는 제약식 집합(C_1)의 원소들을 포함한다. 함수 정의문에서는 함수 본문(function body)의 구문(S_1)에서 발생할 수 있는 예외가 발생할 수 있다.

제약식들의 집합은 구문(S_1)에서 생성되는 제약식 집합(C_1)의 원소들을 포함한다. 반환문(return문)에서는 반환되는 프로그램 식(e_1)에서 발생할 수 있는 예외가 발생할 수 있다. 그리고 제약식들의 집합은 프로그램 식(e_1)에서 생성되는 제약식

집합(C_1)의 원소들을 포함한다. 예외 처리문(try문)에서는 try 블록 안의 실행문(S_1)에서 발생할 수 있는 예외에서 예외(E_c)가 처리된다. 따라서 S_1에서 발생할 수 있지만 처리되지 않은 예외, 예외를 처리할 때 실행하는 구문(S_2), 그리고 예외 처리 후에 실행되는 구문(S_3, S_4)에서 발생할 수 있는 예외가 발생할 수 있다. 그리고 제약식 집합은 구문(S_1, S_2, S_3, S_4)에서 생성되는 제약식 집합(C_1, C_2, C_3, C_4)의 원소들을 포함한다. 처리되는 예외 클래스가 명시되지 않거나 Exception에 의해 예외가 처리되는 예외 처리문에서는 try 블록 안의 실행문(S_1)에서 발생할 수 있는 예외가 모두 처리된다. 따라서 처리 구문(S_2)과 그 후에 실행되는 구문(S_3, S_4)에서 발생할 수 있는 예외가 발생할 수 있다. 그리고 제약식들의 집합은 각 구문(S_1, S_2, S_3, S_4)에서 생성되는 제약식 집합(C_1, C_2, C_3, C_4)의 원소들을 포함한다. 예외 발생문(raise문)에서 예외 클래스의 인자가 없을 때에는 예외(E_c)가 발생한다. 그러나 예외 클래스에 인자로 프로그램 식(e_1)가 있을 때에는 발생시킨 예외(E_c)에 추가적으로 프로그램 식(e_1)에서 발생할 수 있는 예외 또한 발생할 수 있다. 그리고 제약식들의 집합은 프로그램 식(e_1)에서 생성되는 제약식 집합(C_1)의 원소들을 포함한다.

$\triangleright expr : C$

[Constants] $$\frac{e \in \{n,r,s,\texttt{True},\texttt{False},\texttt{None}\}}{\triangleright\, e{:}\emptyset}$$

[Id] $$\frac{}{\triangleright\, \texttt{Name}(id)_e{:}\{X_e \supseteq E_{name}\}}$$

[Attribute] $$\frac{\triangleright\, e_1{:}C_1}{\triangleright\, \texttt{Attribute}(e_1,id)_e{:}\{X_e \supseteq X_{e_1} \cup E_{attribute}\} \cup C_1}$$

[Subscript] $$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, e_2{:}C_2}{\triangleright\, \texttt{Subscript}(e_1,e_2)_e{:}\{X_e \supseteq X_{e_1} \cup X_{e_2}\} \cup E_{index} \cup E_{key}\} \cup C_1 \cup C_2}$$

[New List] $$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, e_2{:}C_2 \quad \triangleright\, e_3{:}C_3}{\triangleright\, \texttt{List}(e_1,e_2,e_3)_e{:}\{X_e \supseteq X_{e_1} \cup X_{e_2} \cup X_{e_3}\} \cup C_1 \cup C_2 \cup C_3}$$

[New Tuple] $$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, e_2{:}C_2 \quad \triangleright\, e_3{:}C_3}{\triangleright\, \texttt{Tuple}(e_1,e_2,e_3)_e{:}\{X_e \supseteq X_{e_1} \cup X_{e_2} \cup X_{e_3}\} \cup C_1 \cup C_2 \cup C_3}$$

[New Dictionary] $$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, e_2{:}C_2 \quad \triangleright\, e_3{:}C_3 \quad \triangleright\, e_4{:}C_4}{\triangleright \texttt{Dictionary}((e_1,e_2),(e_3,e_4))_e{:}\{X_e \supseteq X_{e_1} \cup X_{e_2} \cup X_{e_3} \cup X_{e_4}\} \cup \{C_i \mid 1 \le i \le 4\}}$$

[Binary Operation] $$\frac{\triangleright\, e_1{:}\, C_1 \quad \triangleright\, e_2{:}\, C_2 \quad binOp \in \{/, //, \%\}}{\triangleright\, \texttt{BinOp}(binOp,e_1,e_2)_e{:}\, \{X_e \supseteq X_{e_1} \cup X_{e_2} \cup E_{zeroDiv}\} \cup C_1 \cup C_2}$$

$$\frac{\triangleright\, e_1{:}\, C_1 \quad \triangleright\, e_2{:}\, C_2 \quad binOp \notin \{/, //, \%\}}{\triangleright\, \texttt{BinOp}(binOp,e_1,e_2)_e{:}\, \{X_e \supseteq X_{e_1} \cup X_{e_2}\} \cup C_1 \cup C_2}$$

[Unary Operation] $$\frac{\triangleright\, e_1{:}C_1}{\triangleright\, \texttt{UnaryOp}(UnaryOp,e_1)_e{:}\{X_e \supseteq X_{e_1}\} \cup C_1}$$

[Call] $$\varphi(f) = \{o_i.f \mid o_i \in Object, f \in method(o_i)\}$$
$$o_i.f{=}(x_{f_i},x_i,S_i)$$
$$\frac{\triangleright\, S_i{:}C_i \qquad\qquad \triangleright\, e_2{:}C_{e_2}}{\triangleright \texttt{Call}(e_1.f,e_2)_e{:}\{X_e \supseteq X_{e_1.f} \cup X_{e_2} \cup (\cup_{X_{S_i}})\} \cup (\cup_{C_i}) \cup C_{e_2} \cup C_{e_1.f}}$$

[LibCall] $$\frac{}{\triangleright\, \texttt{LibCall}(e_1.f,e_2)_e{:}\,\emptyset}$$

[ExceptionClass] $$\frac{c \in \{\texttt{NameError},\texttt{KeyError},\texttt{IndexErrorAttributeError},\texttt{ZeroDivisionError}\}}{\triangleright\, \texttt{Except}(c)_e{:}\{X_e \supseteq E_c\}}$$

그림 3.5: expression 제약식 생성 규칙

$\boxed{\triangleright stmt : C}$

[Flow Control]
$$\frac{S \in \{\texttt{Pass},\texttt{Continue},\texttt{Break}\}}{\triangleright\, S{:}\emptyset}$$

[Excecute]
$$\frac{\triangleright\, e_1{:}C_1}{\triangleright\, \texttt{Expr}(e_1)_s{:}\{X_s \supseteq X_{e_1}\} \cup C_1}$$

[Sequence]
$$\frac{\triangleright\, S_1{:}C_1 \quad \triangleright\, S_2{:}C_2}{\triangleright\, \texttt{Seq}(S_1,S_2)_s{:}\{X_s \supseteq X_{S_1} \cup X_{S_2}\} \cup C_1 \cup C_2}$$

[Assign]
$$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, e_2{:}C_2}{\triangleright\, \texttt{Assign}(e_1,e_2)_s{:}\{X_s \supseteq X_{e_1} \cup X_{e_2}\} \cup C_1 \cup C_2}$$

[If]
$$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, S_1{:}C_2 \quad \triangleright\, S_2{:}C_3}{\triangleright\, \texttt{If}(e_1,S_1,S_2)_s{:}\{X_s \supseteq X_{e_1} \cup X_{S_1} \cup X_{S_2}\} \cup C_1 \cup C_2 \cup C_3}$$

[For In]
$$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, S_1{:}C_2}{\triangleright\, \texttt{ForIn}(id,e_1,S_1)_s{:}\{X_s \supseteq X_{e_1} \cup X_{S_1}\} \cup C_1 \cup C_2}$$

[Let]
$$\frac{\triangleright\, e_1{:}C_1 \quad \triangleright\, S_1{:}C_2}{\triangleright\, \texttt{Let}(id,e_1,S_1)_s{:}\{X_s \supseteq X_{e_1} \cup X_{S_1}\} \cup C_1 \cup C_2}$$

[Let-U]
$$\frac{\triangleright\, S_1{:}C_1}{\triangleright\, \texttt{Let}(id,\texttt{Undef},S_1)_s{:}\{X_s \supseteq X_{S_1}\} \cup C_1}$$

[Def Function]
$$\frac{\triangleright\, S_1{:}C_1}{\triangleright\, \texttt{FuncDef}(f,x,S_1)_s{:}\{X_s \supseteq X_{S_1}\} \cup C_1}$$

[Return]
$$\frac{\triangleright\, e_1{:}C_1}{\triangleright\, \texttt{Return}(e_1)_s{:}\{X_s \supseteq X_{e_1}\} \cup C_1}$$

[Try-U]
$$\frac{\triangleright\, S_1{:}C_1 \quad \triangleright\, S_2{:}C_2 \quad \triangleright\, S_3{:}C_3 \quad \triangleright\, S_4{:}C_4}{\triangleright\, \texttt{Try}(S_1,\texttt{Undef},S_2,S_3,S_4)_s{:}\{X_s \supseteq X_{S_2} \cup X_{S_3} \cup X_{S_4}\} \cup \{C_i \mid 1 \le i \le 4\}}$$

[Try-E]
$$\frac{\triangleright\, S_1{:}C_1 \quad \triangleright\, S_2{:}C_2 \quad \triangleright\, S_3{:}C_3 \quad \triangleright\, S_4{:}C_4}{\triangleright\, \texttt{Try}(S_1,\texttt{Except(Exception)},S_2,S_3,S_4)_s{:}\{X_s \supseteq X_{S_2} \cup X_{S_3} \cup X_{S_4}\} \cup \{C_i \mid 1 \le i \le 4\}}$$

[Try]
$$\frac{\triangleright\, S_1{:}C_1 \quad \triangleright\, S_2{:}C_2 \quad \triangleright\, S_3{:}C_3 \quad \triangleright\, S_4{:}C_4}{\triangleright\, \texttt{Try}(S_1,E_c,S_2,S_3,S_4)_s{:}\{X_s \supseteq X_{S_1} - E_c \ \cup \{X_{S_i} \mid 2 \le i \le 4\}\} \cup \{C_i \mid 1 \le i \le 4\}}$$

[Raise-U]
$$\frac{}{\triangleright\, \texttt{Raise}(E_c,\texttt{Undef})_s{:}\{X_s \supseteq E_c\}}$$

[Raise]
$$\frac{\triangleright\, e_1{:}C_1}{\triangleright\, \texttt{Raise}(E_c,e_1)_s{:}\{X_s \supseteq E_c \cup X_{e_1}\} \cup C_1}$$

그림 3.6: statement 제약식 생성 규칙

3.2.4 분석 결과

그림 3.3의 예시 코드를 3.2.3에서 정의한 집합 제약식 생성 규칙에 따라 분석하면 그림 3.7과 같은 결과가 산출된다. 그림 3.7에서 대괄호 사이에 있는 숫자는 에러가 발생한 지점의 처음과 끝의 위치를 나타내며 이는 코드의 처음에서부터 해당 지점까지의 문자의 수를 나타낸다. 중괄호 사이에 있는 Error들은 해당 지점에서 발생한 예외들이다. 예를 들어 [0:4] can raise NameError는 코드의 0번째 문자부터 4번째 문자까지의 지점인 zero 변수에서 NameError가 발생한다는 의미이다.

3.2.3에서 정의한 제약식 생성 규칙들은 안전한(sound)한 분석을 위해 느슨하게 정의한다. 따라서 그림 3.7과 같이 발생하지 않는 예외에도 발생한다고 분석하는 허위 경보가 다수 발생한다.

```
================unhandled error report start ================

[0:4] can raise {NameError, }
[0:8] can raise {NameError, }
[104:119] can raise {NameError, }
[26:119] can raise {NameError, AttributeError, KeyError, Ind
exError, }
[35:38] can raise {NameError, }
[35:53] can raise {NameError, KeyError, IndexError, ZeroDivi
sionError, }
[41:43] can raise {NameError, }
[41:46] can raise {NameError, KeyError, IndexError, }
[41:53] can raise {NameError, KeyError, IndexError, ZeroDivi
sionError, }
[49:53] can raise {NameError, }
[89:90] can raise {NameError, }
[89:99] can raise {NameError, AttributeError, }
[93:94] can raise {NameError, }
[93:99] can raise {NameError, AttributeError, }
[9:11] can raise {NameError, }
[9:25] can raise {NameError, }

unhandeled bugs: 16
```

그림 3.7: 처리되지 않은 예외 분석 결과

제 4 장 분석기 정확도 향상

본 장에서는 3장에서 집합 제약식 기반 분석 방법으로 분석한 결과에서 허위 정보를 줄이기 위한 방안과 외부 라이브러리 함수 호출 시 놓치는 예외를 줄이는 방안을 보인다. 4.1에서는 파이썬 정적 타입 분석기인 Pyright[1]의 분석 결과를 활용한 방법을 설명하고 4.2에서는 중복되는 허위 경보들을 줄이는 방법에 대해 설명한다. 4.3에서는 외부 라이브러리의 함수 호출 시 탐지 못하는 예외를 줄이는 방안을 제시한다.

4.1 파이썬 정적 타입 분석기(Pyright) 활용 방안

경보인 IndexError, KeyError, NameError, AttributeError를 줄이기 위해 파이썬 정적 타입 분석기의 분석 결과를 결합하는데 파이썬 정적 타입 분석기 Pyright는 파이썬의 타입을 분석해주는 분석기로 타입 좁히기(type narrowing)으로 expression들의 타입을 분석한다. 본 연구에서는 파이썬 정적 타입 분석기의 분석 결과에서 변수들의 타입과 정의된 변수 정보를 활용하여 허위 경보를 줄인다.

4.1.1 IndexError, KeyError 허위 경보 축소

3.2.3의 집합 제약식 생성 규칙에 따르면 배열이나 딕셔너리에 인덱스나 키로 접근하면(subscript) IndexError와 KeyError 모두 발생한다고 분석한다. 이는 집합 제약식 기반 분석만으로는 인덱스나 키로 접근하는 객체가 리스트인지 딕셔너리인지 알 수 있는 방법이 없기 때문에 안전한 분석을 위해 두 예외가 모두 발생한다고 분석했다.

KeyError와 IndexError가 모두 발생한다고 분석하여 어쩔 수 없이 발생하는 허위 경보는 파이썬 정적 타입 분석기의 수행 결과를 이용해 줄일 수 있다. 타입을 분석한 결과에서 인덱스나 키로 접근하는 객체의 타입 정보를 이용하면 해당 객

체가 리스트인 경우에는 KeyError를, 딕셔너리인 경우에는 IndexError를 제거할 수 있다.

4.1.2 NameError 허위 경보 축소

NameError 허위 경보를 줄이기 위해 파이썬 정적 타입 분석기의 수행 결과에서 정의된 변수의 정보를 이용하는데 파이썬의 정적 타입 분석기는 타입이 분석되면 타입이나 Any를 분석 결과로 출력하고 그외의 경우에는 Unknown 등의 다른 결과를 출력한다. 이를 이용하여 타입이 분석된 변수의 경우에는 NameError가 발생한다고 분석하지 않게 하여 NameError 허위 경보를 줄인다.

4.1.3 AttributeError 허위 경보 축소

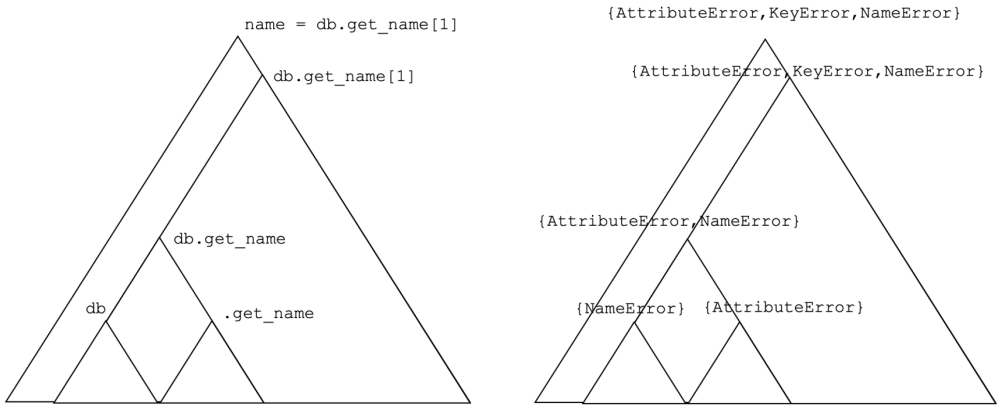
집합 제약식 기반 분석만을 이용해서는 어떤 클래스에 어떤 속성이 정의되어 있는지 알기 어렵다. 따라서 집합 제약식 생성 규칙에서는 속성값에 접근하는 경우 AttributeError가 항상 발생한다고 분석한다. 허위 경보인 AttributeError를 줄이기 위해서 NameError 허위 경보를 줄인 방법과 유사하게 허위 경보를 줄일 수 있다. AttributeError 허위 경보를 줄이기 위해 파이썬 정적 타입 분석기로 객체의 속성 값에 접근하는 전체 코드의 타입을 분석하고 타입 분석 결과를 결합하여 AttributeError 허위 경보를 줄인다.

4.2 반복된 불필요한 경보 압축 방안

프로그램 지점들은 포함 관계에 따라 나타내면 프로그램은 그림 4.1(a)과 같이 나무 모양의 구조를 갖는다. 그림 4.1(a)은 코드 `name = db.get_name[1]`을 이차원의 나무구조로 나타낸 그림이고 그림 4.1(b)는 코드 `name = db.get_name[1]`를 분석했을 때 예외가 발생하는 지점들을 나타낸 그림이다. `db.get_name`의 타입이 딕셔너리라고 가정한다. 맨 위 꼭짓점(root)인 코드 `name = db.get_name[1]`에서 발생 가능하다고 분석되는 예외는 {AttributeError, KeyError, NameError}이고 그 바로 아래의 꼭짓점인 `db.get_name[1]`에서 발생한다고 분석되는 예외들

은 {AttributeError, KeyError, NameError}이다. 각각의 꼭짓점 db.get_name, db, .get_name에서 발생 가능하다고 분석되는 예외들은 각각 {AttributeError, NameError}, {NameError}, {AttributeError}이다.

반복적으로 발생하는 예외들의 집합을 제거하기 위해서 프로그램의 나무 구조에서 전위 순회하면서 중복되는 예외를 지워 불필요한 경보를 줄인다. 중복되는 예외를 제거하는 방법은 다음과 같다. 프로그램의 나무 구조를 순회하면서 말단 꼭짓점(leaf node)이거나 발생 가능하다고 분석되는 예외가 달라지는 경우에 예외들을 분석 결과에 포함한다. 그림. 4.1(b)에서는 프로그램의 가장 위의 꼭짓점과 그 바로 아래의 꼭짓점이 모두 {AttributeError, KeyError, NameError}로 중복된다. 따라서 중복을 제거하여 둘 중 아래의 {AttributeError, KeyError, NameError}만 결과에 포함되도록 하였다. 그 아래에 있는 db, .get_name에서 발생 가능하다고 분석되는 예외들은 말단 꼭짓점이기 때문에 결과에 포함한다. 따라서, 그림 4.1(a)의 예시에서 발생 가능 한 예외는 db.get_name[1], db.get_name, db, .get_name으로 4군데에서 발견된다.



(a) 프로그램의 구조 (b) 프로그램의 구조별 발생 가능한 예외

그림 4.1: 프로그램의 구조 및 발생 가능한 예외

4.3 외부 라이브러리 함수 호출 시 놓치는 예외 축소

3.2.3에서는 라이브러리 함수에서는 예외가 발생하지 않는다고 가정했었지만 파이썬 기본 라이브러리(built-in library)가 아닐 경우에는 라이브러리 함수 내에서 예외가 발생할 수 있고 이에 따라 해당 라이브러리 함수 호출 시 예외가 발생할 수 있다. 따라서 라이브러리 함수 호출 시 예외를 놓치는 문제가 있었다. 이를 해결하기 위해 파이썬 정적 타입 분석기가 타입 분석할 때 외부 라이브러리를 참고하여 타입을 분석할 수 있도록 파일을 자동으로 가져오는 코드를 추가했으며 대검찰청에서 사용한 외부 라이브러리 일부를 분석용 단위 모듈에 추가하여 동시에 분석하였다.

제 5 장 실험 결과

그림 5.1은 그림 3.3의 예시 코드를 허위 경보를 줄인 분석기로 분석했을 때의 결과이다. 그림 5.1에서 `[41:53] can raise {NameError, IndexError}`와 `[41:46] can raise {IndexError}` , `[35:53] can raise {NameError, IndexError}`을 보면 KeyError 허위 경보가 제거된 것을 확인할 수 있다.

```
===============unhandled error report start ===============

[104:119] can raise {NameError, }
[49:53] can raise {NameError, }
[41:46] can raise {IndexError, }
[41:53] can raise {NameError, IndexError, }

unhandeled error: 4
```

그림 5.1: 허위 경보 축소 결과

표 5.1은 대검찰청의 디지털 포렌식 소프트웨어의 분석용 단위 모듈 9 개와 분석용 단위 모듈에 라이브러리 코드를 추가한 벤치마크 1 개, 테스팅용 일반 파이썬 코드 2 개를 허위 경보를 축소하기 전과 후의 분석기로 분석한 결과이다. 9 개의 분석용 단위 모듈 벤치마크(bench1 ∼ bench9.py) 및 라이브러리를 추가한 벤치마크(add_lib.py)는 대검찰청에서 작성한 코드로 외부 노출이 불가하여 예외가 실행 도중 발생하는 테스팅용 코드 벤치마크 2 개를 추가로 작성하였다. testing1.py는 그림 5.2에서 확인할 수 있으며, testing2.py는 그림 3.3에서 확인할 수 있다. 실험 결과에서 허위 경보의 개수는 분석 결과에 실제로 예외가 발생하지 않는 프로그램 지점인데 예외가 발생한다고 분석되어진 지점들의 개수이다.

대검찰청에서 제공한 bench1.py부터 bench9.py까지의 벤치마크를 분석한 결

과 대검찰청에서 찾지 못한 3 가지 패턴의 발생 가능한 예외를 10여 개를 발견하고 허위 경보를 73% ~ 89% 제거하였다. 3 개의 패턴들은 각각 KeyError, IndexError, ZeroDivisionError를 발생시킬 수 있다. 또한, 표를 보면 분석용 단위 모듈(bench1 ~ bench9.py)에 대하여 허위 경보를 78% ~ 89% 줄인 것을 확인할 수 있다.

또한, 두 테스팅용 코드에 대하여도 각각 62%, 87%의 허위 경보를 제거하고 있다. 그림 5.3의 예시 코드에 대하여 처리되지 않은 예외를 분석한 결과 허위 경보를 줄이기 전에 총 14군데에서 예외가 발생한다고 분석됐으며 허위 경보 처리 후에는 6군데에서 예외가 처리되지 않았다고 분석되었다. 두 분석 결과에서 결과에서 `li[5]`에서 예외가 발생한다고 분석한 결과인 `[24:26] can raise {NameError}`는 허위 경보가 아니기 때문에 허위 경보는 각각 13 개, 5 개이다. testing2.py도 마찬가지로 허위 경보를 줄이기 전에는 16군데에서 예외가 발생한다고 분석되었으며 허위 경보 제거 후에는 4군데에서 예외가 발생한다고 분석되었다. 이 분석 결과들에서 허위 경보 줄이기 전에는 `li[7]`에서 1 개의 경보가, 줄인 후에는 2 개의 경보가 허위 경보가 아니었기 때문에 각각 15 개, 2 개의 허위 경보가 발생하였다.

```
zero = 0
try:
    tmp = li[5] / zero
except ZeroDivisionError as e:
    n = e.args
    raise NameError
```

그림 5.2: 처리되지 않은 예외 프로그램 예시

	bench1.py	bench2.py	bench3.py	bench4.py	bench5.py	bench6.py
축소 전	256	579	537	839	316	640
축소 후	30	83	80	191	47	139
	bench7.py	bench8.py	bench9.py	add_lib.py	testing1.py	testing2.py
축소 전	400	316	382	586	13	15
축소 후	62	47	69	161	5	2

표 5.1: 불필요한 경보 축소 전후 비교

4.3에서와 같이 외부 라이브러리의 코드를 대상 코드에 추가하여 함께 분석한 결과, 분석 정확도가 향상하였다. 그러나 다중 상속이나 함수 오버로딩, 함수 오버라이딩, 변수 포함 문자열, 재귀 함수가 있는 파이썬 코드에 대해서는 허위 경보가 다수 발생하였다. 다중 상속과 함수 오버로딩은 호출되는 함수를 정적으로 알기 어렵게하고 이로 인해 분석기의 정확도를 저하시킬 수 있다. 다중 상속 받은 클래스의 내의 함수가 어떤 부모에게서 상속 받은 것인지 정적으로 알기 어렵고 안전한 분석을 위해서는 함수의 MRO를 살펴보면서 호출되는 경우의 수를 모두 고려해야하기 때문에 분석 정확도가 떨어진다. 함수 오버로딩이나 함수 오버라이딩의 경우에도 동일하게, 부모 클래스 내의 함수가 호출되는지 자식 클래스의 함수가 호출되는 지나 동일한 이름의 함수들 중 어느 함수가 호출되는지 정적으로 알기 어렵고 안전한 분석을 위해 두 경우에 발생할 수 있는 예외들이 모두 분석 결과에 포함되도록 하면서 허위 경보가 다수 발생해 분석 정확도가 하락한다. 따라서, 처리되지 않은 예외 분석기가 분석 정확도가 높도록 하기 위해서는 다중 상속과 함수 오버로딩, 함수 오버라이딩을 지향해야한다. 또한, 외부 라이브러리에서와 같이 변수의 값을 문자열에 나타나도록 하기 위해서 format 함수와 %를 사용하면 분석기의 정확도가 저하될 수 있어 두 방법 대신, f-string을 사용하면 정확도에 영향을 안 끼칠 수 있다. 이 두 방법은 실제 문자열과 코드의 순서가 직관적으로 대응되지 않아 코드의 이해가 어렵다는 문제뿐만 아니라 변수의 타입의 정보가 불충분하면 %는 나누기 연산으로 인식되어 ZeroDivisionError가 허위 경보로 발

생하거나 .format가 객체의 속성에 접근하는 것으로 인식되어 AttributeError가 허위 경보로 발생할 수 있다. 그러나, f-string을 사용하면 코드의 이해가 용이하고 중의적으로 인식되지 않아 분석기의 정확도가 저하되지 않는다. 추가적으로, 사용자가 변수의 타입을 알려주는 타입 명시(type annotation)를 분석 대상 코드에 추가하면 분석기의 정확도를 향상할 수 있다. 실험 결과, 충분히 제거되지 않은 허위 경보의 대부분이 타입 분석이 제대로 되지 않아 제거되지 못한 허위 경보이다. 따라서, 사용자가 타입을 부분적으로 주면 더 많은 허위 경보들을 제거하여 분석기의 정확도를 향상시킬 수 있다. 또한, 코드에서 재귀함수 사용 또한 되도록이면 피하고 반복문을 사용해야 한다. 재귀함수와 반복문은 프로그램 상에서 동일하다. 그러나 재귀함수의 경우, 코드를 이해하기 어렵게 한다는 문제와 프로세스 스택에 지역 변수들 등의 변수들과 반환값 등이 저장되면서 스택 오버플로우(overflow)가 발생할 수 있다는 문제가 있다. 따라서, 로컬 변수 사용이 많은 DB와의 연결 작업이나 DB에서 데이터를 읽고 쓰는 작업이 빈번한 디지털 포렌식 소프트웨어에서는 재귀함수 사용은 바람직하지 않다. 따라서 다중 상속 및 함수 오버로딩, 오버라이딩이 없고 재귀 함수 대신 반복문으로 작성되었으며 f-string으로 문자열을 표현한 단순한 파이썬 코드일수록 분석 정확도가 우수하였다. 또한, 외부 라이브러리가 같이 제공되면 놓치는 예외를 줄일 수 있어 분석 정확도를 향상시킬 수 있다.

제 6 장 논의 및 결론

6.1 처리되지 않은 예외 분석기의 타입 분석을 통한 분석 정확도 향상의 한계

5장의 실험 결과에서 허위 경보를 73 ~ 89% 줄였음에도 불구하고 여전히 허위 경보가 발생하는데 그 원인으로 파이썬 타입 분석 정확성의 한계로 인해 허위 경보가 줄어들지 않는 경우를 확인하였다. 파이썬의 경우 변수의 타입이 동적으로 결정되고 (dynamic typing) 실제 클래스와는 무관하게 멤버 함수로만 타입을 판단하는 덕 타이핑(duck typing)을 지원하기 때문에 파이썬에서 타입을 분석하는 것이 어렵다. 특히, 파이썬에서는 정적 타이핑(static typing)을 하는 C나 Java에서처럼 클래스 선언 시 멤버 변수들을 명시적 선언 없이 사용하기 때문에 멤버 변수의 값이 할당되지 않는 멤버 함수에서 멤버 변수의 타입을 분석하기 힘들다. 또한, 덕 타이핑으로 인해 함수 인자의 타입을 실행하기 전까지는 알기 어렵다. 따라서 분석용 단위 모듈에서 클래스의 멤버 변수의 타입 및 함수의 인자들의 타입에 대한 분석을 충분히 정확하게 수행할 수 없기 때문에 이로 인해 허위 경보를 제대로 줄이지 못한다. 5장에서 기술된 bench1.py의 허위 경보를 줄인 이후의 분석 결과는 예외 발생 지점 30군데 중에서 16군데에서는 변수의 타입 분석의 부정확성으로 인해 허위 경보를 줄이지 못하고 있다.

6.2 추가 연구

본 연구를 통해 충분히 제거되지 못한 허위 경보를 줄일 수 있는 추가 연구 주제들을 살펴본다. 6.1에서 제시한 타입 분석의 실패로 인해 줄어들지 않는 허위 경보들을 타입을 이용한 방법이 아닌 다른 방법으로 줄일 수 있는 방안을 고려할 수 있다. 또한, 중복된 허위 경보를 추가로 줄이기 위해 먼저, 클래스에 정의되어 있는 멤버 변수들을 사전 분석을 통해 미리 파악하고 이 정보를 활용하여 허위

경보를 제거할 수 있다. 또한, 처리되지 않는 예외들을 분석하는 과정에 함수의 인자들의 정보를 같이 제공할 수 있다. 마지막으로, 허위 경보의 부모 자식 관계를 활용하여 중복되는 허위 경보를 줄일 수 있다.

6.3 결론

본 연구에서는 대검찰청의 디지털 포렌식 소프트웨어의 분석 부분의 단위 모듈에서 처리되지 않은 예외를 분석하였다. 약한 분석 규칙으로 분석한 결과에서 출발하여 보다 분석 규칙을 정교화하는 과정을 통해 분석의 정확도를 향상시켰다. 그 결과 안전한(sound)한 분석 결과를 유지하면서 허위 경보를 줄이는 성능의 향상을 가져왔다. 특히, 대검찰청의 소프트웨어에서 기존에 찾을 수 없었던 예외 발생 지점 10여 개를 찾을 수 있었으며 평균 84%, 최대 89%의 허위 경보를 제거하였다. 규모가 크고 복잡한 디지털 포렌식 소프트웨어에 대해서는 기존의 연구에서처럼 [14] 실행 결과를 분석하면서 오류를 발견하기 어렵지만 실행 이전에 정적으로 처리되지 않은 예외가 있는지의 분석을 통해 실제 디지털 포렌식 소프트웨어의 신뢰성을 확인하고 버그의 위치를 발견할 수 있다.

또한, 최근 널리 쓰이는 언어인 파이썬에서의 처리되지 않은 예외를 분석한다는 점에도 의의가 있다. 기존에 처리되지 않은 예외에 대한 연구[6, 8, 9]와 타입 분석 및 타입과 관련된 예외에 대한 연구[7]가 진행된 바 있다. 그러나 파이썬에서 타입과 관련된 예외가 아닌 다른 처리되지 않은 예외에 대한 연구는 활발히 연구가 되지 않았다. 또한, 본 연구는 현재 대검찰청의 디지털 포렌식 소프트웨어의 분석용 단위 모듈을 분석 대상으로 하지만 일부 다른 파이썬 프로그램으로 확장하여 처리되지 않은 예외를 분석할 수 있다.

참고문헌

[1] Pyright. `https://github.com/microsoft/pyright`.

[2] Alexandar Aiken. Introduction to set constraint-based program analysis. In *Science of Computer Programming*, volume 35, pages 79–111, Berlin, Heidelberg, 1999. Elsevier.

[3] ArshadHumaira, JantanAman Bin, and AbiodunOludare Isaac. Digital forensics: Review of issues in scientific validation of digital evidence. *Journal of Information Processing Systems*, 14(2):346–376, 4 2018.

[4] Nevin Heintze and Joxan Jaffar. Set constraints and set-based analysis. In Alan Boring, editor, *Principles and Practice of Constraint Programming(PPCP)*, volume 874, pages 281–298, Berlin, Heidelberg, 1994. Springer, Berlin, Heidelberg.

[5] H. Jhoo, S. Kim, W. Song, K. Park, D. Lee, and K. Yi. A static analyzer for detecting tensor shape errors in deep neural network training code. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 337–338, Los Alamitos, CA, USA, may 2022. IEEE Computer Society.

[6] Jangwoo Jo, Byeong-Mo Chang, Kwangkeun Yi, and Kwang-Moo Choe. An uncaught exception analysis for java. *Journal of Systems and Software*, 72(1):59–69, 2004.

[7] Raphaël Monat, Abdelraouf Ouadjaout, and Antoine Miné. Static Type Analysis by Abstract Interpretation of Python Programs. In Robert

Hirschfeld and Tobias Pape, editors, *34th European Conference on Object-Oriented Programming (ECOOP 2020)*, volume 166 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:29, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[8] Kwangkeun Yi. An abstract interpretation for estimating uncaught exception in Standard ML programs. *Science of Computer Programming*, 31(1):147–173, 1998.

[9] Kwangkeun Yi and Sukyoung Ryu. A cost-effective estimation of uncaught exceptions in sml programs. *Theoretical Computer Science*, 277(1–2):185–217, 2002.

[10] 권양섭. 디지털 증거의 증거 능력에 관한 고찰. *사단법인 한국법이론실무학회*, 4(1):149–168, 2016.

[11] 박광하. 경찰 디지털포렌식분석건수 3년새 2배 증가. *정보통신신문*.

[12] 김주석 ; 손지영. *디지털 증거의 증거능력 판단에 관한 연구*, 2015.

[13] 이창환. 지니(gini) 내비게이션 디지털포렌식 분석 사례 및 그 한계. *법과학의 신동향*, pages 140–193, 5 2021.

[14] 이태림. Reliability verification of evidence analysis tools for digital forensics, 2010.

Abstract

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

Seowoo Lee
School of Computer Science Engineering
Collage of Engineering
The Graduate School
Seoul National University

This thesis designs a novel scheme for detecting uncaught exceptions from the digital forensic software is under development in the supreme prosecutors' office and measures its performance, aiming at enhancing the forensic software's reliability. The target forensic software is being coded in Python, while exceptions can be raised from the built-in python library as well as from the faults embedded in the forensic program itself.

Our analyzer design is built on the set constraint-based analysis scheme to detect the uncaught exception statically. This static analysis method begins with the identification of the sets of exceptions that can be raised at each observation point of the program, defines the set constraints between those sets, which finally solves the equations derived from the set constraint. To lower the false alarm rate, we also integrate the Pyright type checker, eliminating the false alarms such as KeyError and IndexError, which cannot be raised in the

list and dictionary type simultaneously. Additionally, our analyzer reorganizes the given code based on the inclusion relationships between each program point into a tree form and traverses it in preorder to remove repeated false alarms.

Our static analyzer has found more than ten spots in the nine benchmarks, while each case belongs to one of three patterns of KeyError, IndexError, and ZeroDivisionError. In addition, our scheme cuts down false alarms by up to 84% on average and 89% at maximum in the nine benchmarks. However, there is still room for improvement by removing the repeated exceptions in the program.

Keywords: uncaught exception, static analysis, Digital Forensic software, Python program

Student Number: 2021-26902

A 부록: 파이썬 IR의 요약된 문법 및 의미구조

A.1 요약된 문법

$$
\begin{array}{rcl}
\textit{Program} & ::= & \textit{stmt} \\
\textit{stmt} & ::= & \texttt{Pass} \\
& | & \texttt{Expr(}\textit{expr}\texttt{)} \\
& | & \texttt{Seq(}\textit{stmt}\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{Assign(}\textit{left-val}\ \texttt{,}\ \textit{expr}\texttt{)} \\
& | & \texttt{If(}\textit{expr}\ \texttt{,}\ \textit{stmt}\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{ForIn(}\textit{left-val}\ \texttt{,}\ \textit{expr}\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{Return(}\textit{expr}\texttt{)} \mid \texttt{Continue} \mid \texttt{Break} \\
& | & \texttt{Let(}\textit{left-val}\texttt{,}\ (\textit{expr} \mid \texttt{Undef})\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{FuncDef(}\textit{id}\ \texttt{,}\ \textit{expr}^*\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{Try(}\textit{stmt}\texttt{,}\ (\textit{exceptionclass} \mid \texttt{Undef}\ \texttt{,}\ \textit{stmt})^*\ \texttt{,}\ \textit{stmt}\ \texttt{,}\ \textit{stmt}\texttt{)} \\
& | & \texttt{Raise(}\textit{exceptionclass}\ \texttt{,}\ (\textit{expr} \mid \texttt{Undef})\texttt{)} \\
\textit{expr} & ::= & n \in \mathbb{Z} \mid r \in \mathit{Float} \mid s \in \mathit{String} \mid \texttt{True} \mid \texttt{False} \mid \texttt{None} \\
& | & \textit{object} \\
& | & \texttt{BinOp(}\textit{Binary-op}\ \texttt{,}\ \textit{expr}\ \texttt{,}\ \textit{expr}\texttt{)} \\
& | & \texttt{UnaryOp(}\textit{Unary-op}\ \texttt{,}\ \textit{expr}\texttt{)} \\
& | & \textit{left-val} \\
& | & \texttt{Call(}\textit{expr.id}\ \texttt{,}\ \textit{expr}^*\texttt{)} \\
& | & \texttt{LibCall(}\textit{id}\ \texttt{,}\ \textit{expr}^*\texttt{)} \\
& | & \textit{exceptionclass} \\
\textit{left-val} & ::= & \texttt{Name(}\textit{id}\texttt{)} \\
& | & \texttt{Attribute(}\textit{expr}\ \texttt{,}\ \textit{id}\texttt{)} \\
& | & \texttt{Subscript(}\textit{expr}\ \texttt{,}\ \textit{expr}\texttt{)} \\
\textit{exceptionclass} & ::= & \texttt{Except(}\textit{Exception-id}\texttt{)} \\
\textit{id} & ::= & \textit{string} \\
\textit{object} & ::= & \textit{id} \mid \texttt{List(}\textit{expr}^*\texttt{)} \mid \texttt{Dictionary(}(\textit{expr}\ \texttt{,}\ \textit{expr})^*\texttt{)} \mid \texttt{Tuple(}\textit{expr}^*\texttt{)} \\
\textit{Numeric-op} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{/} \mid \texttt{//} \mid \texttt{\%} \mid \texttt{**} \\
\textit{Bool-op} & ::= & \texttt{and} \mid \texttt{or} \mid \texttt{is} \mid \texttt{is not} \mid \texttt{in} \mid \texttt{not in}
\end{array}
$$

$$
\begin{array}{rcl}
\textit{Compare-op} & ::= & \texttt{==} \mid \texttt{!=} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \\
\textit{Binary-op} & ::= & \textit{Numeric-op} \mid \textit{Bool-op} \mid \textit{Compare-op} \mid \textit{Bit-op} \\
\textit{Bit-op} & ::= & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \\
\textit{Unary-op} & ::= & \texttt{not} \mid \texttt{-} \\
\textit{Exception-id} & ::= & \textit{id} \mid \texttt{Exception} \mid \texttt{IndexError} \mid \texttt{ZeroDivisionError} \\
 & & \texttt{KeyError} \mid \texttt{AttributeError} \mid \texttt{NameError}
\end{array}
$$

A.2 도메인

$$
\begin{array}{lcrcl}
\sigma & \in & Env & = & Id \overset{fin}{\to} Addr \\
\mathcal{H} & \in & Heap & = & Addr \overset{fin}{\to} Value \\
v & \in & Value & = & ValueExpr + Addr + Object + Func + Exception \\
 & & & & + Cont + \{\ \texttt{None, NotImpl, Undef}\ \} \\
s & \in & ValueExpr & = & String + Boolean + Float + Int \\
e & \in & Exception & = & Id + \{\texttt{Exception, NameError, AttributeError,} \\
 & & & & \texttt{ZeroDivisionError, KeyError, IndexError}\} \\
o & \in & Object & = & (Id + Int + String) \overset{fin}{\to} Value \\
f & \in & Func & = & Id \times Id^* \times Stmt \times Env \\
\kappa & \in & Cont & = & \{\,\texttt{run}, \texttt{cnt, brk}\,\} \\
l & \in & Addr & = & Int(\text{address space}) \\
i, x & \in & Id & = & String(\text{identifier}) \\
stmt & \in & Stmt & = & \texttt{Pass} + ...
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{"length"}, \texttt{""}(\text{empty string}), \ldots & \in & String \\
\_\_len\_\_, \_\_add\_\_, \_\_getitem\_\_, \ldots & \in & Id
\end{array}
$$

요약 환경 (*Env*) 는 식별자(Id)와 요약 메모리 주소 (*Addr*)의 매핑(mapping)이다. 또한, 요약 메모리 (*Heap*) 는 요약 메모리 주소 (*Addr*)와 요약 값 (*Value*)의 매핑

(mapping)이다. 요약 값은 String, Float, Int, Boolean 등의 값 (*ValueExpr*) 또는 요약 메모리 주소, 객체 (*Object*), 함수 (*Func*), 예외 클래스 (*Exception*), 프로그램 진행 값 (*Cont*), NULL 값 `None`, 구현 안된 값 `NotImpl`, 정의 안된 값 `Undef` 이다. 객체는 Id 또는 Int, String으로 부터 요약 값으로의 매핑이다. 함수는 함수 이름을 나타내는 식별자, 함수 인자를 나타내는 식별자, 함수 몸체 ((*Cont*)), 요약 환경으로 구성된다. 프로그램 진행 값은 프로그램 진행(pass, `run`), continue(`cnt`), 프로그램 진행 중단(break, `brk`) 중 하나이다.

A.3의 의미구조에서 도메인 *Object*는 글씨체의 차이에 의해 구분된다. 도메인 *String* 는 딕셔너리 또는 `__getitem__` 함수의 도메인이다 (e.g., `obj["key"]`). *Id* 는 객체의 속성 및 `__getattr__` 함수의 도메인이다 (e.g., `obj.key`). *Int* 는 리스트나 `__getitem__` 함수의 도메인이다 (e.g., `obj[0]`).

클래스, 리스트, 딕셔너리 등 모든 파이썬의 객체들은 *Object*으로 나타낼 수 있다. 최적화를 위해서, *Object* 와 *Func*의 실제 구현에서는 함수의 디폴트 인자등의 추가적인 정보를 가질 수 있다.

```
class A:      # A.__mro__ = (A, object)
  pass
class B(A):   # B.__mro__ = (B, A)
  pass
b = B()       # b.__mro__ = (B, A)
```

그림 1: 클래스와 객체에서의 $\_\_mro\_\_$ 호출 결과의 예

각 *Object*의 값마다 두 개의 빌트인 속성이 있다. *length*속성은 리스트와 같은 객체들에서 원소의 수와 같은 객체의 길이를 나타낸다. $__mro__$는 파이썬에서 Method Resolution Order(MRO)를 의미한다. 본 연구에서는 각 객체의 $\_\_mro\_\_$ 속성마다 길이가 2인 튜플을 할당한다. 튜플의 첫번째 원소는 객체의 클래스이다. 객체가 클래스 일 때는 객체 자신이 첫 번째 원소가 된다. 두번째 원소는 첫번째 원소의 부모 클래스 이다. 실제 파이썬의 문법과는 다르게 요약된 문법에서는 다중 상속을 지원하지 않는다. 객체의 $\_\_mro\_\_$ 속성들이 클래스

상속 트리에서 간선이 된다는 의미이다.

A.3 의미구조

의미구조를 간략하게 나타내기 위해서 도우미 함수 $call$, $attr$, $item$를 아래와 같이 정의한다.

- $call(\mathcal{H}, f, v)$: 힙 공간 $\mathcal{H}$아래에서 인자 v 로 함수 f를 계산한다 .

$$\frac{f = (x_f, x_1, S, \sigma_f) \qquad l_f, l_1 \notin Dom(\mathcal{H}) \qquad \sigma_f[x_f \mapsto l_f, x_1 \mapsto l_1], \mathcal{H}[l_f \mapsto f, l_1 \mapsto v_1] \vdash S \Rightarrow v', \mathcal{H}'}{call(\mathcal{H}, f, v) = v', \mathcal{H}'}$$

- $attr(\mathcal{H}, o, i)$: 힙 공간 $\mathcal{H}$아래에서 객체 o의 속성 i 을 검색한다 . 속성 i가 객체o의 속성이 아니면, 함수 $__getattr__$에 인자 i를 주어 호출한다. 만약 함수 $__getattr__$가 i에 없으면, 함수$__mro__$를 활용해 기본 클래스(base class)를 찾는다.

- $item(\mathcal{H}, o, i)$: 위와 같으나 i 와 함수 $__getitem__$를 사용한다.

$$\frac{o(i) = v}{attr(\mathcal{H}, o, i) = v, \mathcal{H}} \text{ (Attr-Direct)}$$

$$\frac{i \notin Dom(o) \qquad o(\_\_getattr\_\_) = f \qquad call(\mathcal{H}, f, i) = v, \mathcal{H}'}{attr(\mathcal{H}, o, i) = v, \mathcal{H}'} \text{ (Attr-Method)}$$

$$\frac{i, \_\_getattr\_\_ \notin Dom(o) \qquad o(\_\_mro\_\_)(0) \neq o \qquad attr(\mathcal{H}, o(\_\_mro\_\_)(0), i) = v, \mathcal{H}'}{attr(\mathcal{H}, o, i) = v, \mathcal{H}'} \text{ (Attr-Instance)}$$

$$\frac{i, \_\_getattr\_\_ \notin Dom(o) \qquad o(\_\_mro\_\_)(0) = o \qquad attr(\mathcal{H}, o(\_\_mro\_\_)(1), i) = v, \mathcal{H}'}{attr(\mathcal{H}, o, i) = v, \mathcal{H}'} \text{ (Attr-Class)}$$

Expression의 의미구조

$$\boxed{\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}'}$$

$$\frac{E \in \mathbb{Z} + \mathit{Float} + \mathit{String} + \mathit{Bool} + \texttt{\{ None \}}}{\sigma, \mathcal{H} \vdash E \Rightarrow c, \mathcal{H}} \quad \text{(Constraints)}$$

$$\frac{\sigma, \mathcal{H} \vdash E \Rightarrow l, \mathcal{H}' \qquad \mathcal{H}'(l) = v, v \notin \{\mathit{Object}, \texttt{Undef}\}}{\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}'} \quad \text{(Address)}$$

$$\frac{i \in \mathit{Dom}(\sigma)}{\sigma, \mathcal{H} \vdash \texttt{Name(}i\texttt{)} \Rightarrow \sigma(i), \mathcal{H}} \quad \text{(Id)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow v_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v_2, \mathcal{H}'' \\ o = \{0 \mapsto v_1, 1 \mapsto v_2, \texttt{"\$length"} \mapsto 2\} \\ l \notin \mathit{Dom}(\mathcal{H}'')\end{array}}{\sigma, \mathcal{H} \vdash \texttt{List(}E_1, E_2\texttt{)} \Rightarrow l, \mathcal{H}''[l \mapsto o]} \quad \text{(List)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow v_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v_2, \mathcal{H}'' \\ o = \{0 \mapsto v_1, 1 \mapsto v_2, \texttt{"\$length"} \mapsto 2\} \\ l \notin \mathit{Dom}(\mathcal{H}'')\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Tuple(}E_1, E_2\texttt{)} \Rightarrow l, \mathcal{H}''[l \mapsto o]} \quad \text{(Tuple)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow v_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v_2, \mathcal{H}'' \\ \sigma, \mathcal{H} \vdash E_3 \Rightarrow v_3, \mathcal{H}''' \\ \sigma, \mathcal{H}' \vdash E_4 \Rightarrow v_4, \mathcal{H}'''' \\ o = \{0 \mapsto v_1, 1 \mapsto v_2, 2 \mapsto v_3, 3 \mapsto v_4, \texttt{"\$length"} \mapsto 2\} \\ l \notin \mathit{Dom}(\mathcal{H}'''')\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Dictionary((}E_1, E_2\texttt{),(}E_3, E_4\texttt{))} \Rightarrow l, \mathcal{H}''''[l \mapsto o]} \quad \text{(Dictionary)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow v_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v_2, \mathcal{H}'' \\ v_1, v_2 \in \mathit{Int} + \mathit{Float} + \mathit{Bool}\end{array}}{\sigma, \mathcal{H} \vdash \texttt{BinOp(+,}\ E_1,\ E_2\texttt{)} \Rightarrow v_1 + v_2, \mathcal{H}''} \text{(Binary Operation)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v_1, \mathcal{H}'' \\ \mathcal{H}''(l) = o, \mathit{attr}(\mathcal{H}'', o, \_\_\mathit{add}\_\_) = f, \mathcal{H}''' \\ \mathit{call}(\mathcal{H}''', f, v_1) = v, \mathcal{H}'''' \qquad v \neq \texttt{NotImpl}\end{array}}{\sigma, \mathcal{H} \vdash \texttt{BinOp(+,}\ E_1,\ E_2\texttt{)} \Rightarrow v, \mathcal{H}''''} \text{(Binary Operation-LM)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow l_2, \mathcal{H}'' \\ \mathcal{H}''(l_1) = o_1, \mathcal{H}''(l_2) = o_2, \_\_\mathit{add}\_\_ \notin \mathit{Dom}(o_1) \\ \mathit{attr}(\mathcal{H}'', o_2, \_\_\mathit{radd}\_\_) = f, \mathcal{H}''' \\ \mathit{call}(\mathcal{H}''', f, l_1) = v, \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{BinOp(+,}\ E_1,\ E_2\texttt{)} \Rightarrow v, \mathcal{H}''''} \text{(Binary Operation-RM)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l_1, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow l_2, \mathcal{H}'' \\ \mathcal{H}''(l_1) = o_1, \mathcal{H}''(l_2) = o_2, \mathit{attr}(\mathcal{H}'', o_1, \_\_\mathit{add}\_\_) = f_1, \mathcal{H}''' \\ \mathit{call}(\mathcal{H}''', f_1, l_2) = \texttt{NotImpl}, \mathcal{H}'''' \\ \mathcal{H}''''(l_2) = o_2', \mathit{attr}(\mathcal{H}'''', o_2', \_\_\mathit{radd}\_\_) = f_2, \mathcal{H}''''' \\ \mathit{call}(\mathcal{H}''''', f_2, l_1) = v, \mathcal{H}''''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{BinOp(+,}\ E_1,\ E_2\texttt{)} \Rightarrow v, \mathcal{H}''''''} \text{(Binary Operation-RMN)}$$

$$\frac{\sigma, \mathcal{H} \vdash E_1 \Rightarrow v_1, \mathcal{H}' \qquad v_1 \in \mathit{Int} + \mathit{Float} + \mathit{Bool}}{\sigma, \mathcal{H} \vdash \texttt{UnryOp(-,}E_1\texttt{)} \Rightarrow -v_1, \mathcal{H}'} \quad \text{(Unary Operation)}$$

$$\frac{\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}' \qquad \mathcal{H}'(l) = o, \mathit{attr}(\mathcal{H}', o, \_\_\mathit{neg}\_\_) = f, \mathcal{H}'' \qquad \mathit{call}(\mathcal{H}'', f, l) = v, \mathcal{H}''' \qquad v \neq \texttt{NotImpl}}{\sigma, \mathcal{H} \vdash \texttt{UnryOp(-,}\ E_1\texttt{)} \Rightarrow -v, \mathcal{H}'''} \quad \text{(Unary Operation-O)}$$

$$\frac{\sigma, \mathcal{H} \vdash E_1 \Rightarrow f, \mathcal{H}' \qquad \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}'' \qquad \mathit{call}(\mathcal{H}'', f, v) = v, \mathcal{H}'''}{\sigma, \mathcal{H} \vdash \texttt{Call(}E_1\texttt{,}\ E_2\texttt{)} \Rightarrow v, \mathcal{H}'''} \quad \text{(Call)}$$

$$\frac{\sigma, \mathcal{H} \vdash E_1 \Rightarrow o, \mathcal{H}' \qquad \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}'' \qquad \mathit{attr}(\mathcal{H}'', o, \_\_\mathit{call}\_\_) = f, \mathcal{H}''' \qquad \mathit{call}(\mathcal{H}''', f, v) = v, \mathcal{H}''''}{\sigma, \mathcal{H} \vdash \texttt{Call(}E_1\texttt{,}\ E_2\texttt{)} \Rightarrow v, \mathcal{H}''''} \quad \text{(Call-M)}$$

$$\frac{\sigma, \mathcal{H} \vdash E_1 \Rightarrow o, \mathcal{H}' \qquad \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}'' \qquad \mathit{attr}(\mathcal{H}'', o, \_\_\mathit{call}\_\_) = f, \mathcal{H}''' \qquad \mathit{call}(\mathcal{H}''', f, v) = v, \mathcal{H}''''}{\sigma, \mathcal{H} \vdash \texttt{LibCall(}E_1\texttt{,}\ E_2\texttt{)} \Rightarrow v, \mathcal{H}''''} \quad \text{(LibCall)}$$

$$\frac{\sigma, \mathcal{H} \vdash E \Rightarrow l, \mathcal{H}' \qquad \mathcal{H}'(l) = o, o(i) = v}{\sigma, \mathcal{H} \vdash \texttt{Attribute(}E\texttt{,}\ i\texttt{)} \Rightarrow v, \mathcal{H}'} \quad \text{(Attribute)}$$

$$\frac{\sigma, \mathcal{H} \vdash E \Rightarrow l, \mathcal{H}' \qquad \mathcal{H}'(l) = o, \mathit{attr}(\mathcal{H}', o, i) = v, \mathcal{H}''}{\sigma, \mathcal{H} \vdash \texttt{Attribute(}E\texttt{,}\ i\texttt{)} \Rightarrow v, \mathcal{H}''} \quad \text{(Attribute-M)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow x, \mathcal{H}'' \\ \mathcal{H}''(l) = o,\ \_\_getitem\_\_ \notin Dom(o) \\ x \in Int + String,\ o(x) = v\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Subscript}(E_1,\ E_2) \Rightarrow v, \mathcal{H}''} \quad \text{(SUBSCRIPT)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}'' \\ \mathcal{H}''(l) = o, attr(\mathcal{H}'', o, \_\_getitem\_\_) = f, \mathcal{H}''' \\ call(\mathcal{H}''', f, v) = v', \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Subscript}(E_1,\ E_2) \Rightarrow v', \mathcal{H}''''} \quad \text{(SUBSCRIPT-M)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash e \Rightarrow v, \mathcal{H}' \\ v \in \{\texttt{Exception,NameError,AttributeError,} \\ \texttt{ZeroDivisionError,IndexError,KeyError}\}\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Exception}(e) \Rightarrow v} \quad \text{(EXCEPTIONCLASS)}$$

Statement의 의미구조

$$\boxed{\sigma, \mathcal{H} \vdash S \Rightarrow v, \mathcal{H}'}$$

$$\frac{S \in \{\texttt{Pass,Continue,Break}\}}{\sigma, \mathcal{H} \vdash S \Rightarrow \kappa, \mathcal{H}} \quad \text{(FLOW CONTROL)}$$

$$\frac{\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}'}{\sigma, \mathcal{H} \vdash \texttt{Expr}(E) \Rightarrow \texttt{run}, \mathcal{H}'} \quad \text{(EXCECUTE)}$$

$$\frac{\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}'}{\sigma, \mathcal{H} \vdash \texttt{Return}(E) \Rightarrow v, \mathcal{H}'} \quad \text{(RETURN)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash S_1 \Rightarrow \texttt{run}, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash S_2 \Rightarrow v, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow v, \mathcal{H}''} \quad \text{(SEQUENCE-R)}$$

$$\frac{\sigma, \mathcal{H} \vdash S_1 \Rightarrow \texttt{brk}, \mathcal{H}'}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow \texttt{brk}, \mathcal{H}'} \quad \text{(Sequence-FB)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash S_1 \Rightarrow v, \mathcal{H}' \\ v \neq \texttt{brk} \qquad v \neq \texttt{cnt} \\ \sigma, \mathcal{H}' \vdash S_2 \Rightarrow \texttt{brk}, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow \texttt{brk}, \mathcal{H}''} \quad \text{(Sequence-SB)}$$

$$\frac{\sigma, \mathcal{H} \vdash S_1 \Rightarrow \texttt{cnt}, \mathcal{H}'}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow \texttt{run}, \mathcal{H}'} \quad \text{(Sequence-FC)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash S_1 \Rightarrow v, \mathcal{H}' \\ v \neq \texttt{brk} \qquad v \neq \texttt{cnt} \\ \sigma, \mathcal{H}' \vdash S_2 \Rightarrow \texttt{cnt}, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow \texttt{cnt}, \mathcal{H}''} \quad \text{(Sequence-SC)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash S_1 \Rightarrow v, \mathcal{H}' \\ v \neq \texttt{run} \qquad v \neq \texttt{brk} \qquad v \neq \texttt{cnt} \\ \sigma, \mathcal{H}' \vdash S_2 \Rightarrow v', \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Seq}(S_1, S_2) \Rightarrow v', \mathcal{H}''} \quad \text{(Sequence)}$$

$$\frac{\begin{array}{c}f = (x_f, x_i, S_1, \sigma) \qquad l \notin \mathit{Dom}(\mathcal{H}) \\ \sigma[x_f \mapsto l], \mathcal{H}[l \mapsto f] \vdash S_2 \Rightarrow v, \mathcal{H}'\end{array}}{\sigma, \mathcal{H} \vdash \texttt{FunDef}(x_f, x_i, S_1, S_2) \Rightarrow v, \mathcal{H}'} \quad \text{(Def Function)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}' \qquad l \notin \mathit{Dom}(\mathcal{H}') \\ \sigma[i \mapsto l], \mathcal{H}'[l \mapsto v] \vdash S \Rightarrow v', \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Let}(i, E, S) \Rightarrow v', \mathcal{H}''} \quad \text{(Let)}$$

$$\frac{\begin{array}{c}l \notin \mathit{Dom}(\mathcal{H}) \\ \sigma[i \mapsto l], \mathcal{H}[l \mapsto \texttt{Undef}] \vdash S \Rightarrow v, \mathcal{H}'\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Let}(i, \texttt{Undef}, S) \Rightarrow v, \mathcal{H}'} \quad \text{(Let-U)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}' \\ v \in \mathit{Int} + \mathit{Float} + \mathit{Boolean} + \mathit{String} + \mathit{Object} + \mathit{Func} \\ v \notin \{\ \texttt{False,None,0,0.0,""}\ \} \\ \sigma, \mathcal{H}' \vdash S_t \Rightarrow v_t, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{If}(E, S_t, S_f) \Rightarrow v_t, \mathcal{H}''} \quad \text{(If-TC)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}' \\ v \in \{\ \texttt{False,None,0,0.0,""}\ \} \\ \sigma, \mathcal{H}' \vdash S_f \Rightarrow v_f, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{If}(E, S_t, S_f) \Rightarrow v_f, \mathcal{H}''} \quad \text{(If-FC)}$$

$$\frac{\begin{array}{c}\sigma(i) = l \\ \sigma, \mathcal{H} \vdash E \Rightarrow v, \mathcal{H}'\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Assign(Name(}i\texttt{)},E\texttt{)} \Rightarrow \texttt{run}, \mathcal{H}'[l \mapsto v]} \quad \text{(Assign-N)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}' \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}'' \\ \mathcal{H}''(l) = o, o' = o[i \mapsto v]\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Assign(Attribute(}E_1, i\texttt{)},}E_2\texttt{)} \Rightarrow \texttt{run}, \mathcal{H}''[l \mapsto o']} \quad \text{(Assign-A)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow l, \mathcal{H}_1 \\ \sigma, \mathcal{H}_1 \vdash E_2 \Rightarrow v_i, \mathcal{H}_2 \\ \sigma, \mathcal{H}_2 \vdash E_3 \Rightarrow v, \mathcal{H}_3 \\ \mathcal{H}_3(l) = o, \mathit{attr}(\mathcal{H}_3, o, \_\_\mathit{setitem}\_\_) = \mathcal{H}_4, f \\ \mathit{call}(\mathcal{H}_4, f, (v_i, v)) = v', \mathcal{H}_5\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Assign(Subscript(}E_1, E_2\texttt{)},}E\texttt{)} \Rightarrow \texttt{run}, \mathcal{H}_5} \quad \text{(Assign-S)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow l, \mathcal{H}' \\ \mathcal{H}'(l) = o \qquad call(\mathcal{H}', \_\_len\_\_, o) = n, \mathcal{H}_0 \\ l' \notin Dom(\mathcal{H}_0) \qquad item(\mathcal{H}_0, o, 0) = v_0, \mathcal{H}'_0 \\ \sigma[i \mapsto l'], \mathcal{H}'_0[l' \mapsto v_0] \vdash S \Rightarrow \kappa_1, \mathcal{H}_1 \\ \vdots \\ item(\mathcal{H}_{n-1}, o, n-1) = v_{n-1}, \mathcal{H}'_{n-1} \\ \sigma[i \mapsto l'], \mathcal{H}'_{n-1}[l' \mapsto v_{n-1}] \vdash S \Rightarrow \kappa_n, \mathcal{H}_n \\ \forall i, \kappa_i \in \{\texttt{run}, \texttt{cnt}\} \\ \text{(o가 S안에서 바뀌지 않는다고 가정)}\end{array}}{\sigma, \mathcal{H} \vdash \texttt{ForIn(}i, E, S\texttt{)} \Rightarrow \texttt{run}, \mathcal{H}_n} \quad \text{(ForIn)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow l, \mathcal{H}' \\ \mathcal{H}'(l) = o \qquad call(\mathcal{H}', \_\_len\_\_, o) = n, \mathcal{H}_0 \\ l' \notin Dom(\mathcal{H}_0) \qquad item(\mathcal{H}_0, o, 0) = v_0, \mathcal{H}'_0 \\ \sigma[i \mapsto l'], \mathcal{H}'_0[l' \mapsto v_0] \vdash S \Rightarrow \kappa_1, \mathcal{H}_1 \\ \vdots \\ item(\mathcal{H}_{k-1}, o, k-1) = v_{k-1}, \mathcal{H}'_{k-1} \\ \sigma[i \mapsto l'], \mathcal{H}'_{k-1}[l' \mapsto v_{k-1}] \vdash S \Rightarrow \texttt{brk}, \mathcal{H}_k \\ \forall i, \kappa_i \in \{\texttt{run}, \texttt{cnt}\} \\ \text{(o가 S안에서 바뀌지 않는다고 가정}, k \leq n)\end{array}}{\sigma, \mathcal{H} \vdash \texttt{ForIn(}i, E, S\texttt{)} \Rightarrow \texttt{run}, \mathcal{H}_k} \quad \text{(ForIn-B)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow e, \mathcal{H}' \\ e \in \{\texttt{Exception,NameError,AttributeError,} \\ \texttt{ZeroDivisionError,IndexError,KeyError}\}\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Raise(}E, \texttt{Undef)} \Rightarrow \texttt{brk}, \mathcal{H}'} \quad \text{(Raise-U)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E_1 \Rightarrow e, \mathcal{H}' \\ e \in \{\texttt{Exception}, \texttt{NameError}, \texttt{AttributeError}, \\ \texttt{ZeroDivisionError}, \texttt{IndexError}, \texttt{KeyError}\} \\ \sigma, \mathcal{H}' \vdash E_2 \Rightarrow v, \mathcal{H}''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Raise}(E_1, E_2) \Rightarrow \texttt{brk}, \mathcal{H}''} \quad \text{(RAISE)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow e, \mathcal{H}' \qquad e \in \{\texttt{Undef}, \texttt{Exception}\} \\ \sigma, \mathcal{H}' \vdash S_1 \Rightarrow \texttt{brk}, \mathcal{H}'' \\ \sigma, \mathcal{H}'' \vdash S_2 \Rightarrow v', \mathcal{H}''' \\ \sigma, \mathcal{H}''' \vdash S_4 \Rightarrow v'', \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Try}(S_1, E, S_2, S_3, S_4) \Rightarrow v'', \mathcal{H}''''} \quad \text{(TRY-UX)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow e, \mathcal{H}' \qquad e \in \{\texttt{Undef}, \texttt{Exception}\} \\ \sigma, \mathcal{H}' \vdash S_1 \Rightarrow v, \mathcal{H}'' \\ \sigma, \mathcal{H}'' \vdash S_3 \Rightarrow v', \mathcal{H}''' \\ \sigma, \mathcal{H}''' \vdash S_4 \Rightarrow v'', \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Try}(S_1, E, S_2, S_3, S_4) \Rightarrow v'', \mathcal{H}''''} \quad \text{(TRY-UNX)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow e, \mathcal{H}' \qquad e \notin \{\texttt{Undef}, \texttt{Exception}\} \\ \sigma, \mathcal{H}' \vdash S_1 \Rightarrow \texttt{brk}, \mathcal{H}'' \\ \sigma, \mathcal{H}'' \vdash S_2 \Rightarrow v', \mathcal{H}''' \\ \sigma, \mathcal{H}''' \vdash S_4 \Rightarrow v'', \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Try}(S_1, E, S_2, S_3, S_4) \Rightarrow v'', \mathcal{H}''''} \quad \text{(TRY-X)}$$

$$\frac{\begin{array}{c}\sigma, \mathcal{H} \vdash E \Rightarrow e, \mathcal{H}' \qquad e \notin \{\texttt{Undef}, \texttt{Exception}\} \\ \sigma, \mathcal{H}' \vdash S_1 \Rightarrow v, \mathcal{H}'' \\ \sigma, \mathcal{H}'' \vdash S_3 \Rightarrow v', \mathcal{H}''' \\ \sigma, \mathcal{H}''' \vdash S_4 \Rightarrow v'', \mathcal{H}''''\end{array}}{\sigma, \mathcal{H} \vdash \texttt{Try}(S_1, E, S_2, S_3, S_4) \Rightarrow v'', \mathcal{H}''''} \quad \text{(TRY-NX)}$$