TraceInspector - 추적 가능한 시각화 기반 코드 정적분석기

배경

정적 분석(Static Analysis)은 프로그램을 실행하지 않고 코드의 실행 특성을 추론하여 잠재적인 오류를 탐지하는 기술이다. IDE, 컴파일러, CI 등 다양한 개발 과정에서 코드 품질과 안정성을 높이기 위해 활용되고 있음.
기존 분석기들은 분석 결과만 제시할 뿐 어떻게 오류가 탐지되었는지 파악하기 어렵고, 분석 과정을 일종의 “블랙박스”처럼 받아들이고 결과를 맹신하게 됨.

TraceInspector 소개

TraceInspector는 요약해석(Abstract Interpretation) 이론을 바탕으로 한 정적 분석기와 더불어 분석 과정을 시각적으로 보여주는 인터페이스로 구성되어 있음.
분석 중 생성되는 요약 상태(Abstract State)의 변화 과정을 사용자에게 실시간으로 보여주어 각 프로그램 지점에서 정보가 어떻게 전파 및 정제되는지를 직관적으로 관찰할 수 있도록 구현함.

분석 과정

1: Go 코드 → 중간 표현 언어(Imp)

Go AST를 간단한 형태의 Imp 언어로 번역. 분석기는 Imp 코드에 대해 동작하며, 분석기 구현 및 타 언어 이식 시 용이하도록 설계.

```
func main() {
    a := 5
    arr := make_array(a, 1337)
    for i := 0; i < 10; i++ {
        arr[i] = i
    }
    Print(arr)
}
```

→

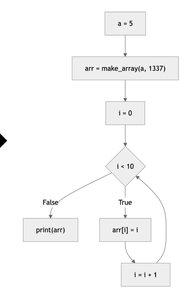
```
fun main() none {
        a = 5
        arr = make_array(a, 1337)
        i = 0
        while i < 10 {
                arr[i] = i
                i = i + 1
        }
        print(arr)
}
```

2: Imp → 제어 흐름 그래프**(CFG)**

프로그램 내 분기, 반복 등의 제어 흐름 단위로 나타내는 제어 흐름 그래프 형태로 표현. 각각의 명령문들은 단일 노드로 표현됨.

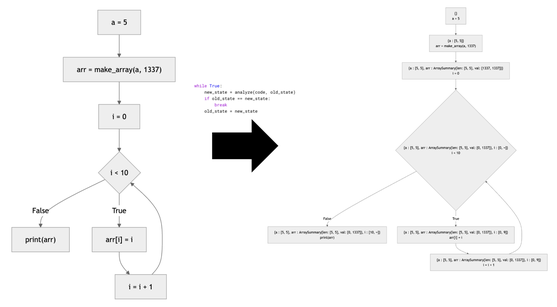
```
fun main() none {
        a = 5
        arr = make_array(a, 1337)
        i = 0
        while i < 10 {
                arr[i] = i
                i = i + 1
        }
        print(arr)
}
```

→



3: CFG 위에서 고정점 근사

CFG 노드별 진입 시점에서의 프로그램 상태를 근사. 고정점에 도달할 때까지 반복적으로 요약실행 및 상태 join/widening 수행.



분석 예시**:** 무한 루프 및 **Dead Code Detection**

```
1 func main() {
2     a := 1          a: [1, 1]
3     for a > 0 {     a: [1, 5]
4         a++         a: [1, ∞]
5     }               a: ⊥
6     Print(a)        a: ⊥
7 }
```

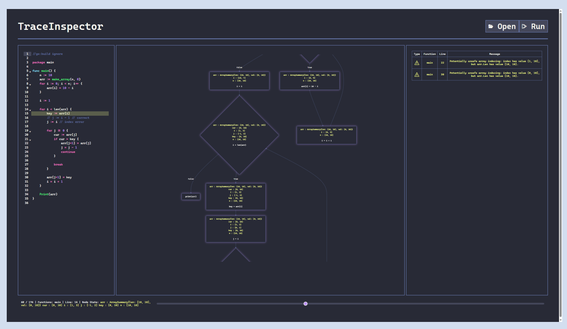
무한히 반복되는 코드의 경우 계산된 상태에 대해 widening 기법을 적용하여 분석이 종료할 수 있도록 유도. 이후 filtering을 통해 결과를 정제하여 정확도 손실을 방지.
추가로 항진/모순명제를 탐지하여 도달할 수 없는 코드에 대해 경보.

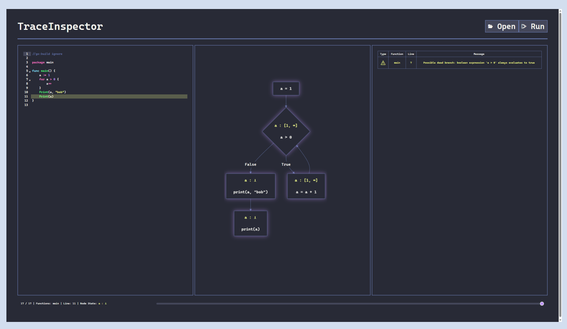
분석 예시**:** 배열 인덱싱 범위

```
1 i := 1                        i: [1, 1]
2 for i < len(arr) {            i: [1, 10], len(arr): [10, 10]
3     key := arr[i]             i: [1, 9]
4     arr[i+1] = arr[i]         i: [1, 9], key: [0, 10]
5 Print(i)                      i: [10, 10], key: [0, 10]
```

배열의 길이와 인덱스 변수에 대한 요약상태를 계산 후 안전한 인덱싱 범위인지 확인.

시각화 및 인터페이스





- 분석 과정에서 생성되는 상태 변화를 단계별로 재생 가능하도록 구현.
- 함수별 및 단계별 상태를 자유롭게 탐색 가능.

한계 및 향후 계획

- 관계형 요약도메인 : `i < len(arr)` 등과 같이 구체적인 값이 아닌 변수간의 논리적 관계를 나타낼 수 있는 대수적 도메인.
- 분석 정확도 개선 : reduced product domain, narrowing 구현 및 SimpleProp 체계 확장.
- Pointer inference : 포인터 및 메모리 동작 추상화를 통한 포인터 연산 검증.
- C → Imp, Rust → Imp 등 Imp를 활용한 다양한 언어 지원.
- 보다 다양한 언어 기능 지원 : 확장된 타입 체계 등.

2조 - 20212978 김신영, 20212952 강민성
국민대학교 소프트웨어학부