

Theorem Problem

SNU 4541.664A Program Analysis

Spring 2006

Note 6

Prof. Kwangkeun Yi

계획

1 프로그램 분석(*static program analysis*) 개관

프로그램 분석(static program analysis)

“소프트웨어 MRI” 기술

- sw가 우리가 바라는 대로 실행될것인가?
- 답을 자동으로 해 주는 것이 가능한가?
- 프로그램 분석 기술이 한 해결책
 - “static analysis”, “abstract interpretation”, “type system”, “model checking”, “theorem proving”, “data-flow analysis” 등
- 응용
 - sw 오류검증, sw 최적화, sw 관리, 등등 ∞

프로그램 분석(*static program analysis*)

프로그램의 실행 성질을
실행전에 자동으로
안전하게 어림잡는
일반적인 방법

“정적 분석” “static analysis”

- “실행전”: 프로그램을 실행시키지 않고
- “자동으로”: 프로그램이 프로그램을 분석
- “안전하게”: 모든 가능성을 포섭
- “어림잡는”: 실제 이외의 것들이 포함됨
 - 어림잡지 않으면 불가능
- “일반적”: 소스언어와 성질에 무제한
 - C, C++, C#, Java, ML, Z, UML, JVM, x86, bits, etc.
 - “buffer overrun?”, “memory leak?”,
“x=y at line 2?”, “memory use \leq 2K?”, etc.

프로그램 분석기 개발 싸이클

- 1 분석할 소스 언어 결정:
 - C, C++, Java, C#, ML, or ?
- 2 분석할 실행 성질 결정:
 - buffer overrun, null dereference, constant, class, deadlock, race, uncaught exception, unhandled case, etc.
- 3 (디자인; 구현; 테스트)⁺

모든 프로그램 분석은 3스텝

- ① “연립방정식”을 세운다
 - 요약된 세계에서 (abstract semantic domains)
 - 프로그램의 실행 다이내믹스에 관한 (abstract execution flows)
- ② 그 방정식을 푼다
- ③ 그 해를 가지고 결론을 내린다
 - 있는가 없는가? 같은가 다른가?

분석방정식 세워보기

```
x = 1;  
y = read;  
if (x+y-1 ≤ 0)  
    then x = x+1;  
    else y = x-1;
```


분석방정식 세워보기

```
x = 1;  
while (x ≤ 99)  
    x++;
```

분석방정식 세워보기

```
int x[10]; int *y;  
for (i = 0; i < 10; i++)  
    x[i] = 1;  
y = x + 2;  
y[3] = 0;
```

프로그램 분석 이론

- 올바른 연립방정식을 어떻게 유도하는가?
 - 유도한 연립방정식의 해는 항상 올바른가?
- 유도한 연립방정식의 해는 항상 있는가?
 - 그 해를 어떻게 계산하는가?
 - 유한시간내에 해를 구할 수 있는가?

프로그램 분석 실제

- 우리는 불가능에 도전한다

Theorem (OTL)

주어진 프로그램 분석기를 바보로 만드는 프로그램이 있다, 즉,
 \forall 프로그램 분석기 \exists *pgm*. 프로그램 분석기(*pgm*) =
 “흐리거나 맑겠슴”.

- 분석기는 대상 소스언어로 짜여진 모든 sw에는 효과적일 수 없다.

- 우리의 희망은:

Theorem (Rainbow)

프로그램 분석기는 대상 프로그래밍 언어로 짜여진 흔한/특정 프로그램들에 효과적일 수 있다

- 게임의 법칙:
 - 올바른 분석기 만들기 (꼭 올바라야 할까?)
 - 그 분석기의 효율(*cost-accuracy balance, cost-effectiveness*) 높이기