CompCert로 따로따로 컴파일한 뒤 링킹하기

서울대학교 컴퓨터공학부 소프트웨어 원리 연구실 강지훈 허충길

CompCert: 엄밀히 검증한 C 컴파일러

C → CompCert → Asm

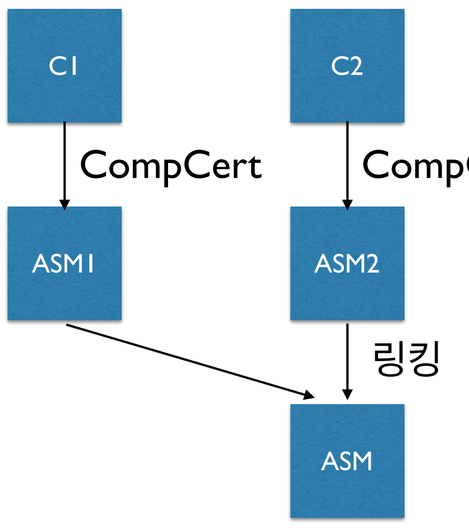
$$C(c) = \text{OK } a \rightarrow \text{Behav}(a) \subseteq \text{Behav}(c)$$

한꺼번에 컴파일한 경우만 엄밀히 검증

existing libraries compiled with `gcc`. This is convenient for testing, although the formal guarantees of semantic preservation apply only to whole programs that have been compiled as a whole by CompCert C.

AirBus: "Whole program compilation이 뭐에요?"

따로따로 컴파일한 뒤 링킹 검증

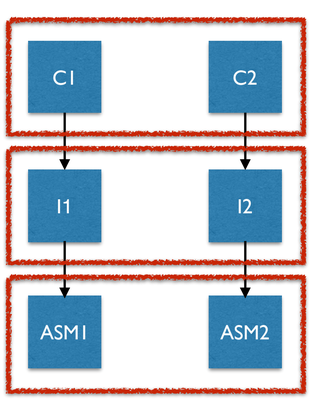


- 현대 소프트웨어 개발에 중요한 툴
- CompCert의 가장 큰 trusted computing base
- 기존 시도: Princeton, Yale

$$C(c_1) = \text{OK } a_1, C(c_2) = \text{OK } a_2 \rightarrow \text{Behav}(a_1 + a_2) \subseteq \text{Behav}(c_1 + c_2)$$

검증 방법

컴파일 단계별로 쪼개 증명 ($C = C_b \circ C_f$)



$$C_f(c_1) = \text{OK } i_1, C_f(c_2) = \text{OK } i_2 \rightarrow \text{Behav}(i_1 + i_2) \subseteq \text{Behav}(c_1 + c_2)$$

$$C_b(i_1) = \text{OK } a_1, C_b(i_2) = \text{OK } a_2 \rightarrow \text{Behav}(a_1 + a_2) \subseteq \text{Behav}(i_1 + i_2)$$

쉬운 증명의 핵심 아이디어

Monotonicity

- Compiler가 입력 프로그램을 분석해서 분석한 성질을 컴파일에 이용하는 경우,
- 그 성질은 "더 큰" 프로그램에 대해서도 성립해야 한다.
- $A(c_1)$ should be sound w.r.t. $c_1 + c_2$

Monotonicity 증명하다 CompCert 뒷단 버그 2개 찾음

기존 증명 재사용

- CompCert 코드의 2%만 변경
  - Monotonicity를 제외하면 기계적인 변경
- CompCert를 모르고 시작해서 한사람이 한달만에 끝냄

CompCert 2.4 (최신버전) 버그

Monotonicity 위반: Value Analysis

- CompCert 2.2에 추가됨 (다른 최적화 단계를 돕기 위해)
- extern int* const p;
  가 pointer를 가지고 있지 않다고 잘못 분석 (링킹시 틀림)

```
// a.c
int b;
extern int* const a;
int main() {
  b = 1;
  *a = 0;
  return b + b; }  // expected: 0, actual: 2
```

```
// b.c
extern int b;
int* const a = &b;
```

AbsInt / CompCert ★ Star 24 Fo

More prudent analysis of uninitialized const global variables. Browse code

In the presence of separate compilation and linking, an uninitialized const global variable may be initialized elsewhere with a pointer value, falsifying the points-to analysis. Report and fix by Chung-Kil Hur and Jeehoon Kang.

master

xavierleroy authored 14 days ago 1 parent 06841a5 commit 5aecefe808aaaad6ab05037e7d5e7f53b53e0b94

Monotonicity 증명 불가: 알고보니 잘못된 Axiom

Axiom: "__i64_dtos"라는 함수가 extern으로 선언되어 있고, 그 실행 의미는 float-to-long long 캐스팅이다.

```
long long __i64_dtos(float t) { return 3; }
int main() { return (long long) 5.0f; }
  // expected: 5, actual: 3
```

AbsInt / CompCert ★ Star 24 Fo

Protect against redefinition of the __i64_xxx helper library functions. Browse code

This is achieved by declaring these functions in C2C.ml, then re-checking their declarations in the global environment during the Selection pass.

In passing, the "hf" parameter for SelectLong functions was removed and replaced by fixed identifiers corresponding to the actual names of the helper functions.

master

xavierleroy authored 3 days ago 1 parent 28d7ff1 commit 9f2ca1049957004161834090a697cd4118e2fb08

비교

- Princeton (POPL 2015), Yale (CPP 2015)
- (-) CompCert 많이 변경, 검증하는데 오래 걸림
- (-) CompCert 전체 검증 못함, 최신 버전 검증 못함
- (+) CompCert 말고 다른 컴파일러와도 링킹 가능 (우리 추후 연구)

CompCert의 가치

- CompCert는 버그가 있어서 쓸모없는 프로젝트이다 (X)
- 엄밀히 검증한 CompCert의 가치가 더욱 빛난다 (O)

엄밀한 검증 혹은 면밀한 검토 없이는 소프트웨어에 버그가 있을 확률이 높습니다.

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 NRF ERC
SEOUL NATIONAL UNIVERSITY