CompCert로 따로따로 컴파일한 뒤 링킹하기

강지훈 허충길
서울대학교 소프트웨어 원리 연구실

ROSAEC 웍샵
201501

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

CompCert

- 엄밀히 검증된 C 컴파일러
  - Xavier Leroy (INRIA) et al.
    - 10년동안 Coq으로 구현 및 검증
- 버그가 없다…고들 한다.
  - Finding and understanding bugs in C compilers. PLDI 2011

근데 우리가 버그 찾음

ccomp a.c b.c && ./a.out && echo $?

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

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

// expected: 0, actual: 2

링킹 검증하다 버그 찾음

- 링킹: 프로그램들을 합치기 (보통 Assembly 프로그램)
- CompCert는 링킹한 프로그램에 대한 검증 X
- 우리가 분연히 일어서서 검증
- 증명이 잘 안돼서 살펴보니 버그 발견!
  - https://github.com/AbsInt/CompCert/commit/5aecefe
  - https://github.com/AbsInt/CompCert/commit/9f2ca10

AbsInt / CompCert

Unwatch 12 Unstar 24 F

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 13 days ago 1 parent 06841a5 commit 5aecefe808aaaad6ab05037e7d5e7f53b53e0b94

AbsInt / CompCert

Star 24 F

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

메시지

- CompCert는 쓸모없는 프로젝트이다 (X)
- CompCert의 가치가 더욱 빛난다 (O)
  - 엄밀한 검증 혹은 면밀한 검토 없이는,
    소프트웨어에 버그가 있을 확률이 엄청 크다.

결론

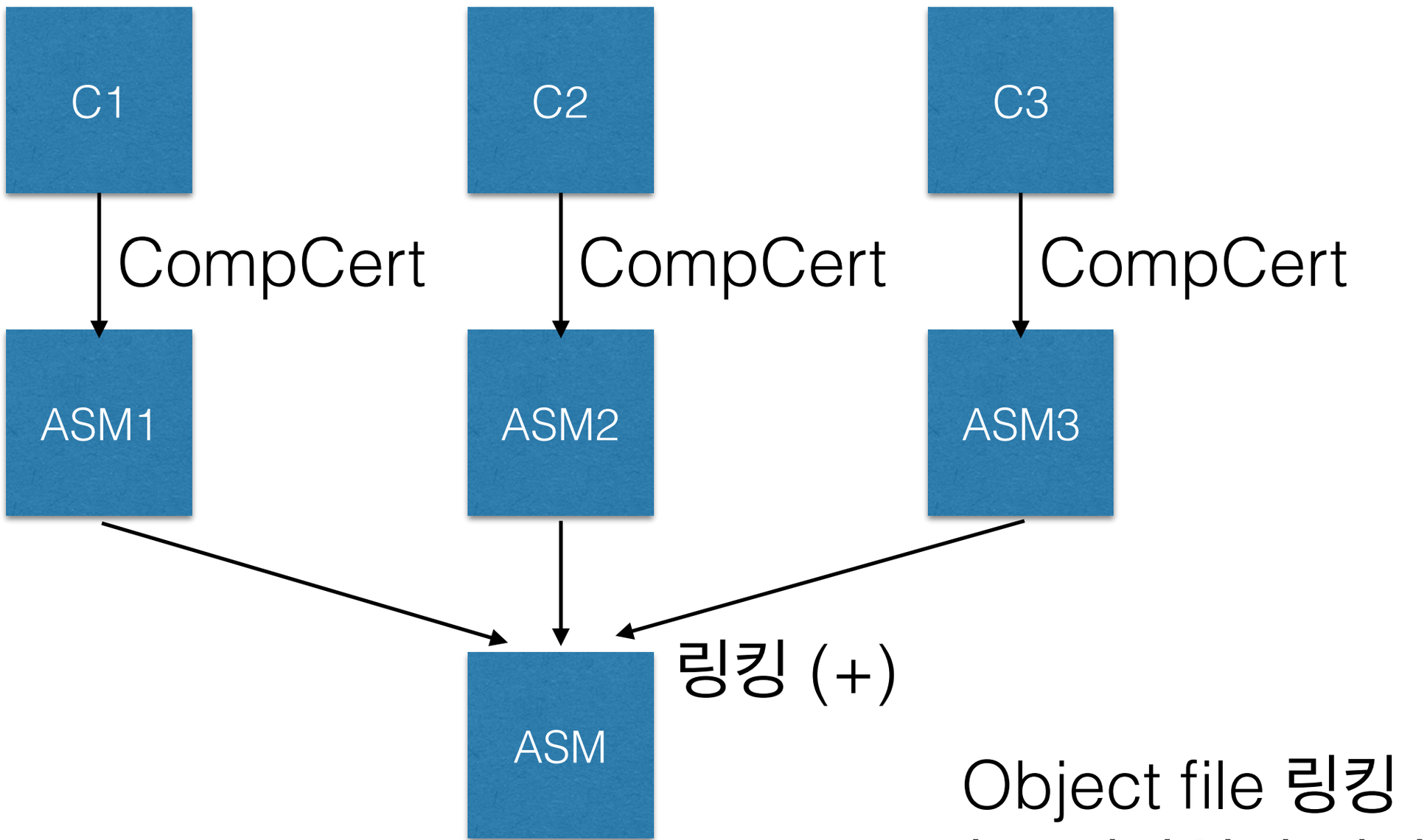
- 이제는 CompCert로 따로따로 컴파일한 뒤 링킹해도 옳다.

- 엄밀한 검증 혹은 면밀한 검토 없이는,
  소프트웨어에 버그가 있을 확률이 엄청 크다.

좀 더 구체적으로

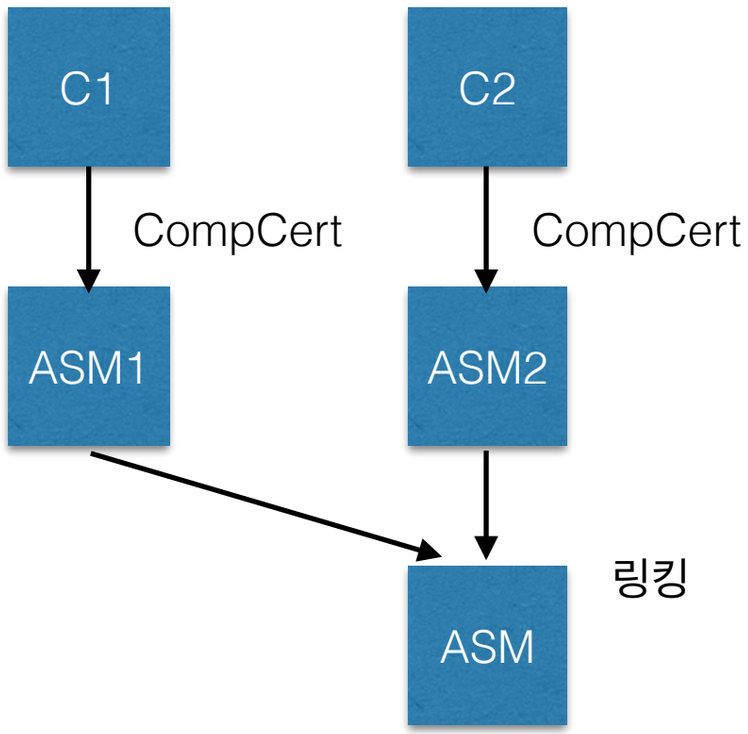
슬라이드는 총 22장

따로따로 컴파일한 뒤 링킹하기



Object file 링킹
다른 컴파일러 링킹

올바른 컴파일



- C1, C2가
  ASM1, ASM2로 컴파일되면

- ASM=ASM1+ASM2의 행동이
  C1+C2의 행동과 같다.

링킹이 실패할 수도

따로따로 컴파일은 중요한 문제

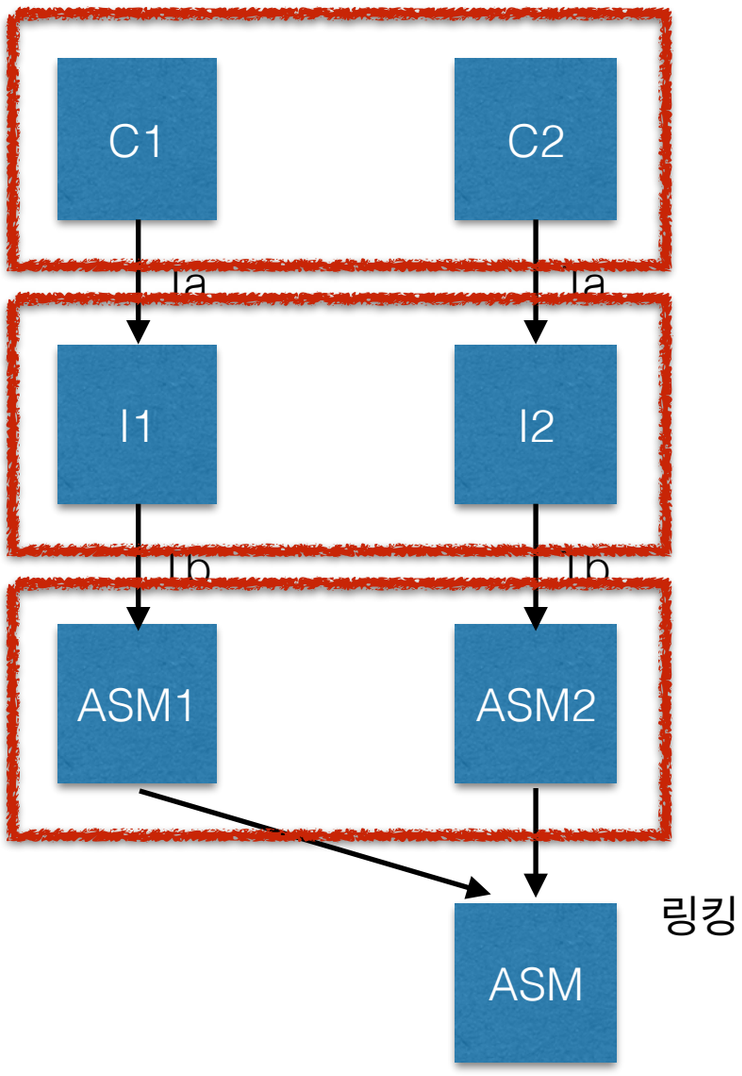
Part 3: Assembling and linking. The abstract syntax tree for PowerPC or ARM or x86 assembly language produced by part 2 is printed in concrete assembly syntax. The system's assembler and linker are then called to produce object files and executable files, respectively. This part is not yet formally verified. A benefit of using the standard assembler and linker is that object files produced by CompCert can be linked with 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.

compcert.inria.fr

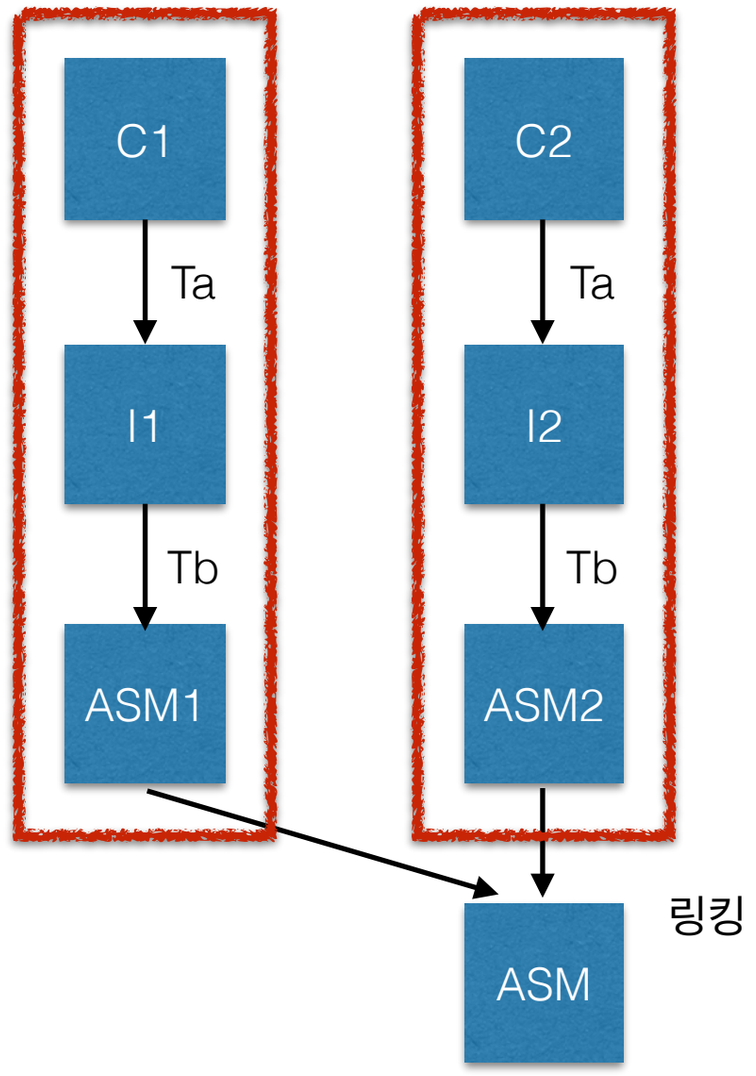
- AirBus: “whole-program compilation이 뭐에요?”

검증 전략 1: 단계별로 쪼개기

우리 방법 (쉬움)

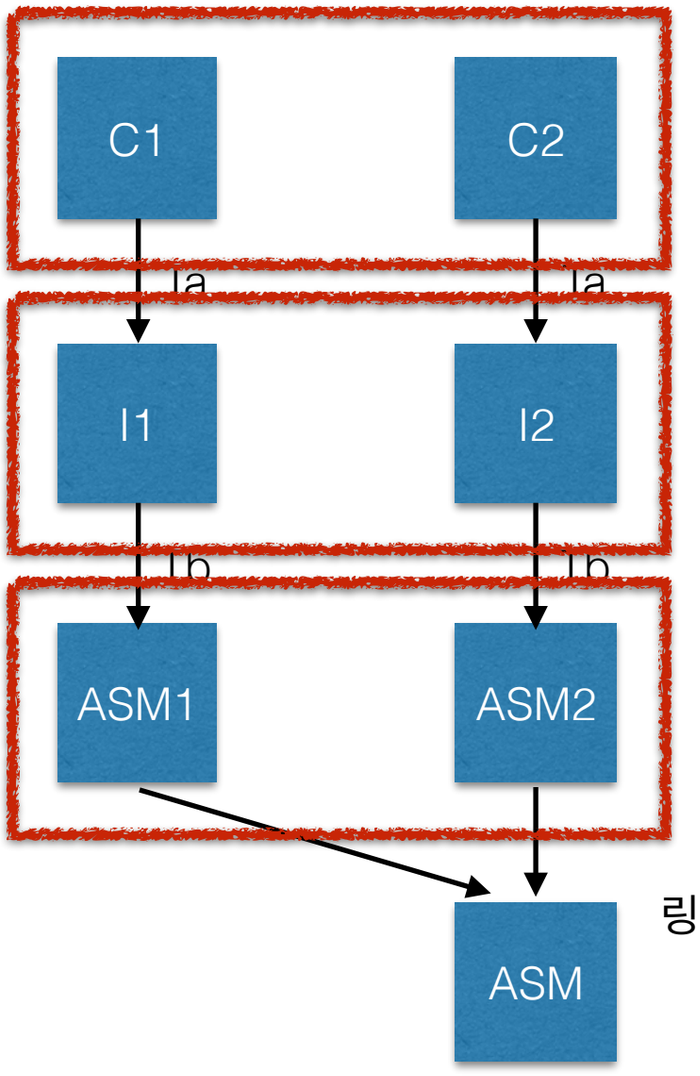


기존 방법 (어려움)



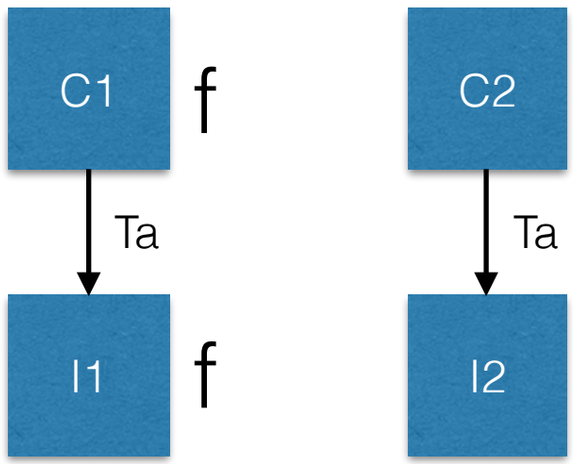
검증 전략 1: 단계별로 쪼개기

우리 방법 (쉬움)



- 전체 프로그램에 대해 증명 가능
  - 기존대로 모듈별로 하면 모듈에 대해 증명해야 함
- 기존의 증명을 많이 재사용 가능

검증 전략 2: 기존 증명 재사용



- CompCert에 이미
  C1과 I1의 행동이 같음이
  증명되어 있음

- 우리는 예컨대
  C1의 함수 f와 I1의 함수 f가
  C1+C2, I1+I2에서도 같은 행동을 보임을
  증명하면 됨

- Monotonicity:
  C1에서 성립하는 성질은
  C1+C2에서도 성립한다.

Monotonicity 1
Int64 연산을 라이브러리 함수 호출로

```
float a;                                  float a;
...                                       ...
f((long long) a);                         f(__int64_dtos(a));
```

Monotone해야 하는 성질:
__int64_dtos가 float-to-int64 캐스팅을 구현하는 외부 함수이다.

C1에서 성립하면, C1+C2에서도 성립해야 함
증명하는 중에 **CompCert의 버그 찾음**

Int64 관련 버그

```
#include <stdio.h>

long long __i64_dtos(float t) {
  return 3;
}

int main() {
  printf("%lld\n", (long long) 5.0f);
    // expected: 5, actual: 3
  return 0;
}
```

https://github.com/AbsInt/CompCert/commit/9f2ca10

Monotonicity 2
Value Analysis

- 3개 단계에서 value analysis 결과 이용
- Monotonicity: C1의 value analysis 결과가 C1+C2에서도 옳아야 한다.
- 증명하는 중에 **CompCert의 버그 찾음**
  - 처음에 소개한 버그

정리

- CompCert를 몰랐던 한 사람이 1개월만에
  (교수님 제가 밤새 생각해봤는데 2일은 아니에요…)

- CompCert 2.4 (최신 버전)에 대해

- 링커를 정의하고

- CompCert로 컴파일된 Assembly들을 링킹하면
  원래 C 프로그램과 행동이 같다는 것을 증명

- CompCert의 2%만 수정 (Coq 파일만)

- 그와중에 CompCert 버그 2개 발견 (뒷단에서는 처음)

비교 1: Princeton

- Compositional CompCert. POPL 2015
- (-) 10 사람-달
- (-) CompCert 2.1
- (-) Inlining, common subexpression elim. 단계를 빼먹음
- (-) CompCert에서 정의한 언어의 semantics를 바꿈
- (-) Assembly 링킹이 비현실적
- (+) 다른 컴파일러나 손으로 컴파일한 프로그램끼리도 링킹 가능

비교 2: Yale

- A compositional semantics for verified separate compilation and linking. CPP 2015
- (-) CompCert version?
- (-) 몇개 단계밖에 증명 못함
- (-) CompCert에서 정의한 언어의 semantics를 바꿈
- (+) 다른 컴파일러나 손으로 컴파일한 프로그램끼리도 링킹 가능

결론

- 이제는 CompCert로 따로따로 컴파일한 뒤 링킹해도 옳다.
  - 앞으로 다른 컴파일러나 손으로 컴파일한 프로그램도 링킹해도 옳음을 증명할 예정

- 엄밀한 검증 혹은 면밀한 검토 없이는,
  소프트웨어에 버그가 있을 확률이 엄청 크다.


תודה
Dankie
Gracias
Спасибо
شكراً
Merci
Takk
Köszönjük
Terima kasih
Grazie
Dziękujemy
Dėkojame
Ďakujeme
Vielen Dank
Paldies
Kiitos
Täname teid
谢谢
Thank You
Tak
感謝您
Obrigado
Teşekkür Ederiz
감사합니다
Σας Ευχαριστούμ
ขอบคุณ
Bedankt
Děkujeme vám
ありがとうございます
Tack


?


!