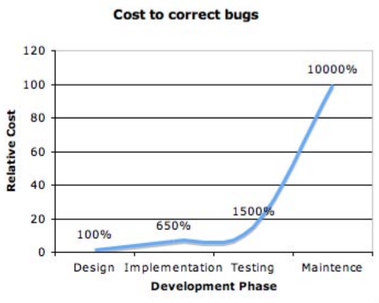
컴파일러 버그박멸단

서울대학교 컴퓨터공학부 프로그래밍 연구실 / 소프트웨어 원리 연구실

시대는 소프트웨어 시스템의 엄밀한 검증을 원한다



버그 비용이 너무 크다
특히 안전이 중요한 시스템인 경우

CPU, OS, 컴파일러는
모든 사람들이 사용
엄밀히 검증할 가치 O

i7은 이미 엄밀한 검증이 테스트 대체
Intel의 경험: 비슷한 가격, 더 높은 안전성
앞으로도 엄밀한 검증을 이어갈 예정

컴파일러는 지금도 버그가 쏟아진다

그래서 LLVM 컴파일러를 검증할거다

기존 컴파일러 그대로 호환성 걱정 없이 쓰도록
많은 사람들이 쓰도록 성능 걱정 없이 쓰도록

재밌고 의미있는 문제가 매우 많다

동시성 메모리, Compositional 논증,
Concurrent Separation Logic,
증명 비용 관리, 기존 증명 재사용, …

당신과 함께하길 원한다

어디로 연락하면 되나?

서울대학교 컴퓨터공학부 허충길 교수님, 혹은 강지훈

ROPAS

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

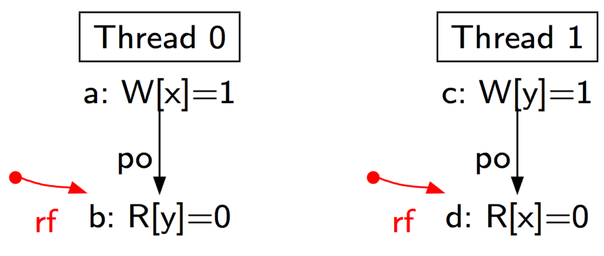
SEOUL NATIONAL UNIVERSITY

무슨 재밌는 문제들이 있나?

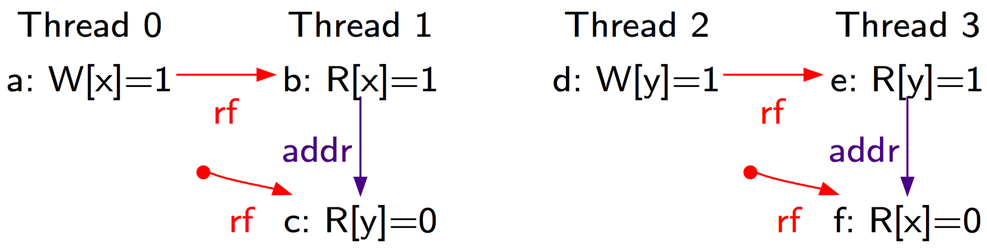
동시성 메모리 모델 다루기

예제: 메모리는 맵이 아니다

* Sarkar et al. Understanding POWER multiprocessors
* Mador-Haim et al. An axiomatic memory model for POWER multiprocessors



Test SB : Allowed
Allowed in: C11/C++11, x86, Sparc, ARM, POWER



Test IRIW+addrs: Allowed
Allowed in: C11/C++11, ARM, POWER

복잡한 동시성 메모리에 대해 어떻게 논증할까?

* Turon et al. GPS: Navigating Weak Memory with Ghosts, Protocols, and Separation

선행 연구들이 있지만, 컴파일러 검증에 쓰기 위해 더 연구해야 함

Compositional 논증

- 컴파일러는 보통 한 번에 프로그램의 일부분만 컴파일
  - Horizontally: 소스 파일 별로 컴파일
  - Vertically: C→IR→기계어
- 컴파일러가 일부분씩 "옳다"는 보장을 해줄 수 있을까? 그리고 나중에 일부분씩 "옳음"을 합쳐서, 전체가 옳음을 증명할 수 있을까?
- "Compositional 논증"이라는 이름의 연구 분야
  - 허충길 교수님이 5년동안 연구
- 할 일: 동시성 메모리 모델에 대한 compositional 논증 방법 고안

그동안 다루지 못했던 최적화 고안하기

- 관찰: GCC, LLVM 컴파일러는 주로 간단한 최적화를 한다.
- 예상 이유: 간단하지 않으면 맞는지 틀리는지 알기 어려움
- 할 일
  - 간단하지 않으면서 성능에 영향을 주는 최적화 고안
    - 예: 자동 parallelization
  - 안전성 증명
- 기대 효과: 간단하지 않은 최적화도 안심하고 적용 가능

글로 된 LLVM 언어를 엄밀히 정의하기

LLVM 문서 예제: 무슨 말인지 아시겠나요?

"Poison values are similar to undef values, however they also represent the fact that an instruction or constant expression which cannot evoke side effects has nevertheless detected a condition which results in undefined behavior."

LLVM는 "정의"되어있다기보다는 "설명"되어 있음

LLVM 문서 엄밀히 정의하기

사람들이 상식적으로 이해하는 LLVM 그대로
버그 없이 명확하고 엄밀하게
엄밀한 정의 위에서 LLVM 최적화가 올바르도록

기존 증명 재사용하기

엄청난 노력이 들어간 컴파일러 검증 작업들

CompCert, CompCertTSO, Vellvm,
SimpleBerry (우리가 이번에 한 일)

기존 컴파일러 검증 작업

CompCert, CompCertTSO, Vellvm,
SimpleBerry (우리가 이번에 한 일)

증명 비용을 줄이기 위해 증명을 재사용하자

난관: 우리 semantics 모델이 기존 작업과 많이 다름
메모리 모델 변경, 광범위한 nondeterminism 도입, …

더 쉬운 증명 도구 개발, 이용하기

Coq: 증명 보조기 계의 기계어

- 자동화 수준이 아쉬움
  - 너무 세세한 detail도 전부 다 써줘야 함
  - SMT solver와의 연동?
- 전문가가 아니면 쉽게 쓰기 힘듬
  - Software Foundations: 1학기 분량의 튜토리얼
- 하지만 Coq 증명이 "옳음"을 보장하는 가장 좋은 방법

증명 비용을 줄이기 위해 다른 도구를 개발하자

아이디어: Coq backend of VeriML / Dafny