C언어의 재탄생: 믿고 쓰는 C언어 환경 만들기

윤용호 강지훈
서울대학교

더러운 C 표준 문서

- 700페이지 가량의 줄글
- 수학적 실행 의미 정의가 아닌, case-by-case 스펙 제시
- 철저히 컴파일러 개발사의 편이를 위한 표준
- 당신이 짠 C 프로그램도 다 틀린 프로그램!

  SPEC 2000 벤치마크에서도 틀린 프로그램이 쏟아짐

"그럼 뭐가 문제인가요? 어차피 다들 틀리는데?"

컴파일러가 제멋대로 최적화를 돌립니다.
당신 생각하고 다르게 컴파일 됩니다.
그런 버그는 잡는데 한 세월이죠.

아무도 지키지 못하는 표준이 무슨 의미가 있는가?

컴파일러 버그 존재

GCC도 LLVM도 버그 투성이 (PLDI'11, Yang)
컴파일러도 사람이 만드는 프로그램이니까.

심 지 어,

다 증명 됐다고 해서 믿고 써도 될줄 알았던
CompCert에서도 버그 발견 (by 강지훈, 허충길)
PLDI'11에선 발견하지 못한 버그들

https://github.com/AbsInt/CompCert/commit/9f2ca10
https://github.com/AbsInt/CompCert/commit/5aecefe

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

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

검증하지 않았던 부분에서의 버그

```
long long __int64_dtos(float a)
{ return 2; }
int main() {
  float a = 0.0;
  printf("%lld", (long long)a);
}
// expected: 0, actual: 2
```

잘못된 Axiom으로 인한 버그

교훈: 검증이 이렇게 중요함,
Axiom 하나도 신경써서 가정해야함

그럼, 뭘 믿고 C를 써야하나?

C 정의 다시하기

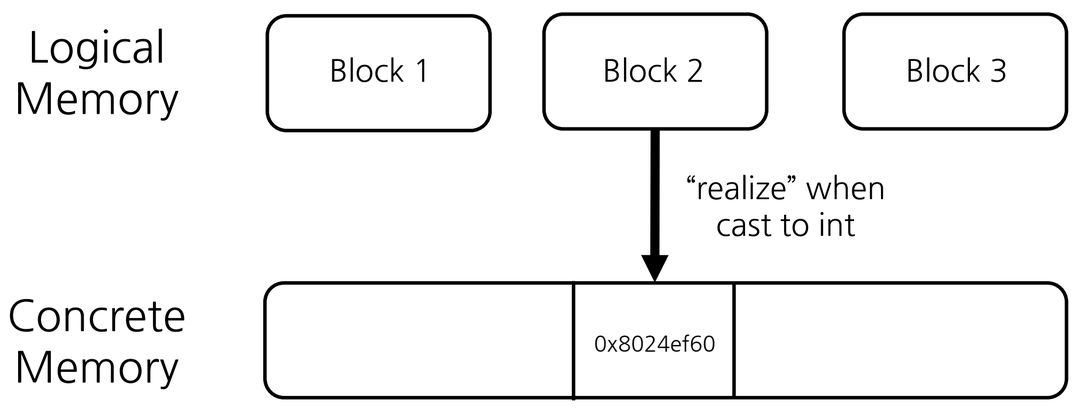
C언어에 대한 성질이 아닌 수학적 정의를!
기존 C 컴파일러의 중요한 최적화들 대부분을 설명 가능하게.

몇 가지 정리해야 할 문제들

- Integer-Pointer Cast ⟵ 우리가 해결
- Undef Value, Dangling Pointer, Non-termination, Evaluation Order, Strict Aliasing, Etc.

정수와 포인터간 변환을 지원하는 메모리 모델

- 기존의 연구들은 변환을 제대로 지원하지 못함
- 변환을 지원하면서도 기존의 최적화 룰들을 깨지 않는 메모리 모델 정의



- 이 모델로 기존의 많은 최적화들이 정당화 가능함을 Coq으로 보임
- PLDI 2015에 제출

컴파일러에 검산기 달기

Coq으로 올바름이 증명된 검산기를 장착하자
LLVM 컴파일러의 모든 과정을 검산할 것

SimpleBerry

- Control-Flow Graph를 변경하지 않는 대부분의 최적화 패스를 검산할 수 있는 틀
- InstCombine 패스의 꼬마 최적화들에 대해 검산 가능
- 더 많은 패스를 커버하도록 확대 예정

Alias Analysis 검산하기

- 최적화 과정에서 참조하는 AlaisAnalysis 결과를 검산
- 메모리 참조가 얽힌 최적화를 검산하는 데에 필수
- 초기단계 진행중.

CFG를 바꾸는 패스 검산하기

Assembly 단계에서의 최적화 검산하기

기타 등등

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