동시성 메모리 모델

서울대학교 컴퓨터공학부
소프트웨어 원리 연구실
강지훈 (jhkang@ropas.snu.ac.kr)

2014 여름 @ ROSAEC 워크샵

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

서울대학교 컴퓨터공학부
Seoul National University
Dept. of Computer Science and Engineering

이 뭐죠?

메모리 “모델”?

- 이 회로를 곧이곧대로
  다 표현할 수 없음

- 모델링, 즉 추상화가 필요!

우리가 즐겨쓰던 메모리 모델

Collecting Semantics Collecting semantics of program P is an invariant $[\![P]\!] \in \mathbb{C} \to 2^{\mathbb{S}}$ that represents a set of reachable states at each control point, where the concrete domain of states, $\mathbb{S} = \mathbb{L} \to \mathbb{V}$, maps concrete locations ($\mathbb{L}$) to concrete values ($\mathbb{V}$).

Oh et al. Design and Implementation of Sparse Global Analyses for C-like Languages. PLDI 2012

질문: 이 모델이 실제 메모리의 옳은 추상화일까?

답: 네 니오 (쓰레드가 여러개면 틀림)

동시성 메모리 예제

문: 다음 C 코드의 행동으로 올바른 것을 모두 고르시오.

```
int message=0; int flag=0;

// thread 1
message = 100;
flag = 1;

// thread 2
while (!flag) {}
printf(“%d\n”, message);
```

- 100을 출력한다.
- 0을 출력한다.

- (undefined behaviour)

문: 왜 0을 출력할 수 있을까?

성능을 중시한
동시성 메모리 모델

- 위 예제는 버그가 아님
  - 더 나은 CPU 성능을 위해 상식 포기
- 실제 컴퓨터 시스템을 모델링하려면,
  동시성 메모리 모델이 꼭 필요
  - 단, 기존 메모리 모델도
    문제 도메인에 따라 유용할 수도 있음 (뉴턴 역학)

상식 복원하기

```
int message=0; atomic<int> flag;
flag.store(0);

// thread 1
message = 100;
flag.store(1);

// thread 2
while (!flag.load()) {}
printf(“%d\n”, message);
```

- 적절한 동기화가 필수
- 다른 동기화 방법 가능 (mutex, …)

동시성 메모리 모델 디자인 원리

- 성능을 위해 상식 포기
  - 상식이 필요할 때에는 동기화
- 다양한 수준의 동기화
  - 고수: 높은 성능, 약한 동기화
  - 초보: 강한 동기화, 낮은 성능

제가 모델을 공부하는 목적

- **컴파일러 버그박멸단**: 실제 컴퓨터 시스템에서 돌아가는, 널리 쓰이는 컴파일러를 증명해서 버그를 박멸하자!

- C, LLVM, 기계어의 동시성 메모리를 잘 모델링 해야

- 모델링뿐만 아니라 증명에 사용할 논증 모델을 만들어야

튜토리얼로 소개하는 목적

- 재밌어요!
- 유용해요!
- 학교에서 배웠던 상식이 뒤집히는 느낌
  - 예: ARM에서는 미리 실행(speculative execution)의 결과를 프로그래머가 관찰할 수 있음
  - 여러분도 충격 받으세요!

차례

- C11/C++11 동기화 예제
- 연구 동향 소개
- 제가 하고 있는 고민들

C11/C++11 동기화 예제

Batty et al. Mathematizing C++ Concurrency

Load Buffering (LB) 1
undefined behaviour

Semantics는 race 없는 프로그램에 대해서만 정의됨
더 많은 최적화를 위해

x: race, 근데 race는
undefined behaviour

```
int x, y;

//thread 1
assert(y==1);
x=1;

//thread 2
assert(x==1);
y=1;
```

Load Buffering (LB) 2
allowed

relaxed: race는 없음. 근데 아무런 동기화 보장이 X

```
atomic<int> x, y;

//thread 1
assert(y.load(relaxed)==1);
x.store(1, relaxed);

//thread 2
assert(x.load(relaxed)==1);
y.store(1, relaxed);
```

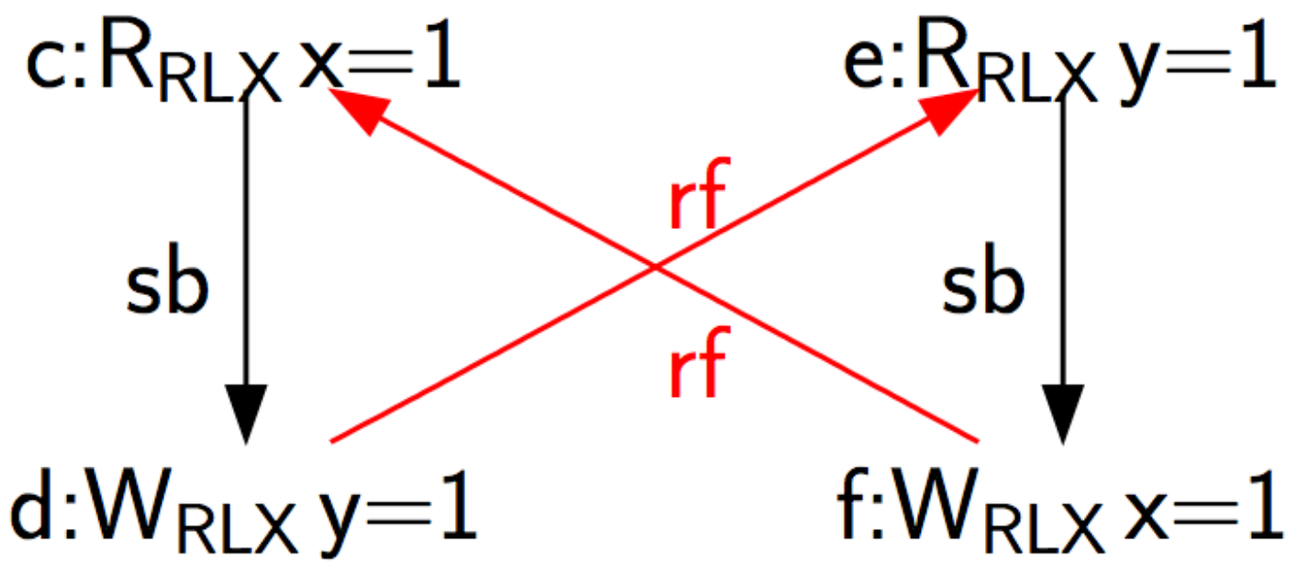
Load Buffering (LB) 2
설명

sb: sequenced-before (스레드 내)
rf: reads-from
relaxed에 대해서는 (거의) 아무런 제약조건이 없음

```
atomic<int> x, y;

//thread 1
assert(y.load(relaxed)==1);
x.store(1, relaxed);

//thread 2
assert(x.load(relaxed)==1);
y.store(1, relaxed);
```



Load Buffering (LB) 3
disallowed

sequentially consistent: 가장 강한 수준의 동기화

```
atomic<int> x, y;

//thread 1
assert(y.load(seq_cst)==1);
x.store(1,seq_cst);

//thread 2
assert(x.load(seq_cst)==1);
y.store(1, seq_cst);
```

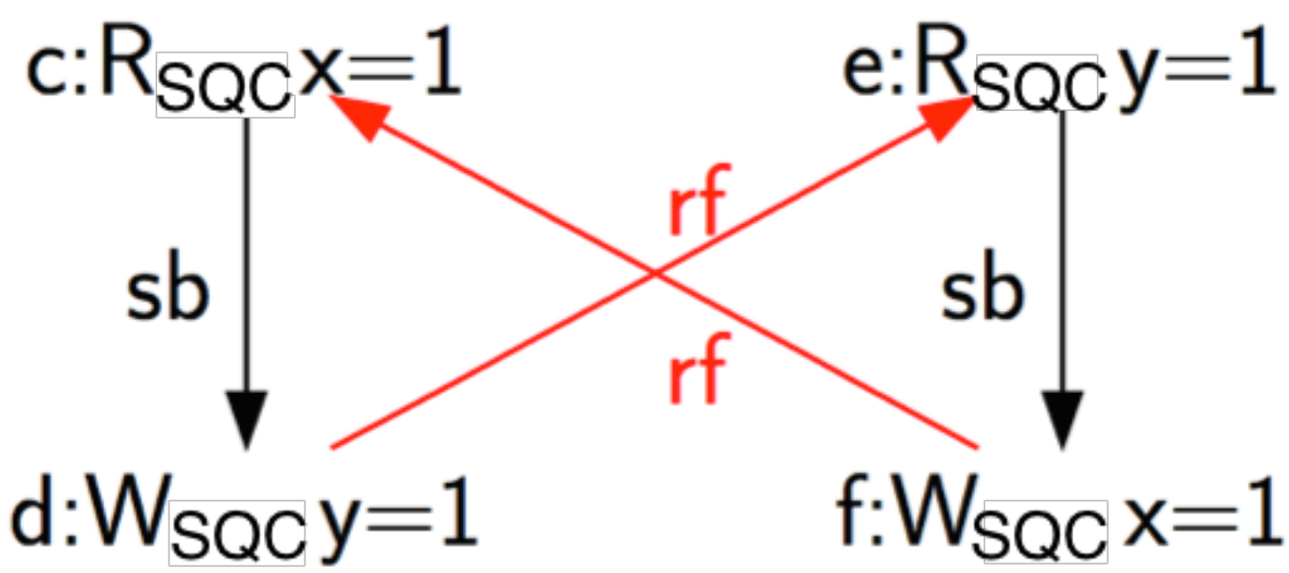
Load Buffering (LB) 3
설명

- Sequentially consistent한 접근 사이에는 sc라는 total order가 있음
- sb는 sc와 어긋나면 안됨
- rf는 sc와 어긋나면 안됨
- 그럼 sc의 cycle을 만들 수 있어서 sc가 total order임이 모순

```
atomic<int> x, y;

//thread 1
assert(y.load(seq_cst)==1);
x.store(1,seq_cst);

//thread 2
assert(x.load(seq_cst)==1);
y.store(1, seq_cst);
```



다양한 동기화

지면이 부족하여 설명은 여기서 줄입니다

높은 성능

- Not Atomic
- Atomic
- Relaxed
- Consume
- Acquire, Release, Acquire/Release
- Sequentially Consistent

LB는 사실 Release, Consume만으로 disallow 가능

더 강한 동기화

연구 동향

여러 동시성 메모리 모델 정의

- 언어별 메모리 모델
  - C11/C++11 메모리 모델 (2011),
    자바 메모리 모델, .NET 메모리 모델
- CPU별 메모리 모델
  - x86 메모리 모델, Sparc 메모리 모델 (2010),
    ARM 메모리 모델, POWER 메모리 모델 (2011)
- 목표: 1) 실제 CPU/컴파일러 행동 포섭, 2) 쉽게 논증

Peter Sewell. http://www.cl.cam.ac.uk/~pes20/weakmemory/index.html
Java Community Process. JSR-000133 JavaTM Memory Model and Thread Specification Revision

동시성 메모리 모델 위에서 논증하기

- 동시성 메모리 모델 위에서 분리 논리 (자원의 소유권)
  - Vafeiadis et al. Relaxed separation logic: a program logic for C11 concurrency
  - Turon et al. GPS: Navigating Weak Memory withGhosts, Protocols, and Separation
- 동시성 메모리 모델 위에서 (간단한) compositional 논증 (작은 증명을 합쳐 큰 증명 만들기)
  - Batty et al. Library Abstraction for C/C++ Concurrency

동시성 메모리 모델 컴파일

이를테면, 컴파일러

- C11/C++11
- ARM

Batty et al. Clarifying and Compiling C/C++ Concurrency: from C++11 to POWER

제가 하고 있는 고민들

LLVM 메모리 모델?

- LLVM: **컴파일러 버그박멸단**의 타깃
- LLVM 메모리 모델은 정의가 없는듯…
  - 몇몇 예제의 설명만 (http://llvm.org/docs/Atomics.html)
  - C11/C++11 메모리 모델과 닮았음
- C11/C++11 메모리 모델 차용? 매우 복잡함…
- LLVM-TSO로 목표를 한정할까? 그럼 ARM은?

연구 방향

- 이론적 목표: 동시성 언어를 다루는 컴파일러에 관한 이론
- 실제적 목표: `clang -O1` 검증

- 얼마나 큰 목표일까? (혹은, 박사과정 내에 끝낼 수 있을까?)
  - 많은 사람들과 같이 일하면 좋겠다!
- 어떤 순서로 일을 할까? (뭐부터 할까? 같이 할까?)
- LLVM, C/C++, 아키텍쳐 전문가가 되야 하나?

팀 구성하기

큰 일이 될겁니다.

허충길 교수님이나
저에게 연락하세요!

감사합니다
질문/답변/코멘트?