소프트웨어재난연구센터 2023 겨울정기워크숍 주요연구발표

Concrat

An Automatic C-to-Rust Lock API Translator for Concurrent Programs

홍재민, 류석영(KAIST) · ICSE 2023

Concrat

An Automatic C-to-Rust Lock API Translator for Concurrent Programs

Concrat

An **Automatic C-to-Rust** Lock API **Translator** for Concurrent Programs

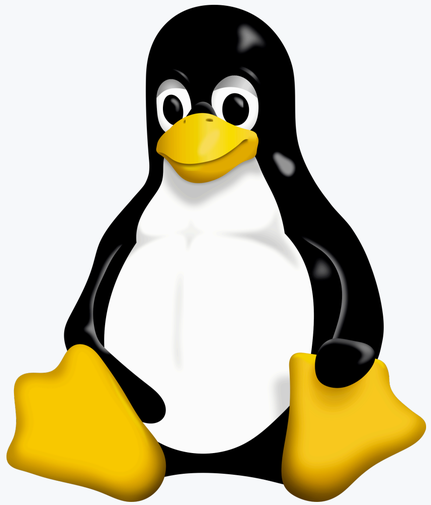
Concrat

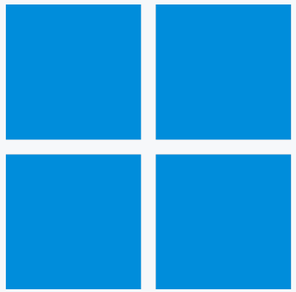
An Automatic C-to-Rust Lock API Translator for Concurrent Programs

Concrat

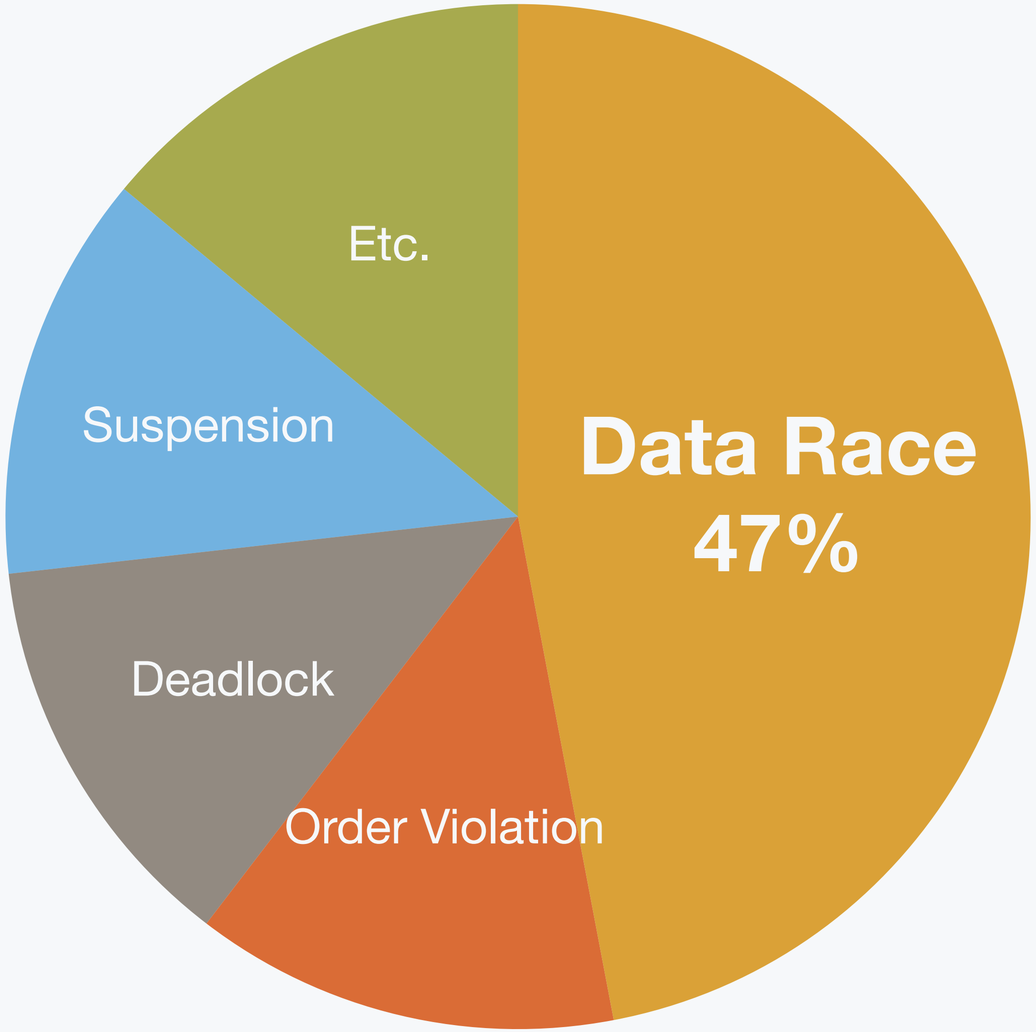
An **Automatic C-to-Rust Lock API Translator** for **Concurrent Programs**

동시성 프로그램





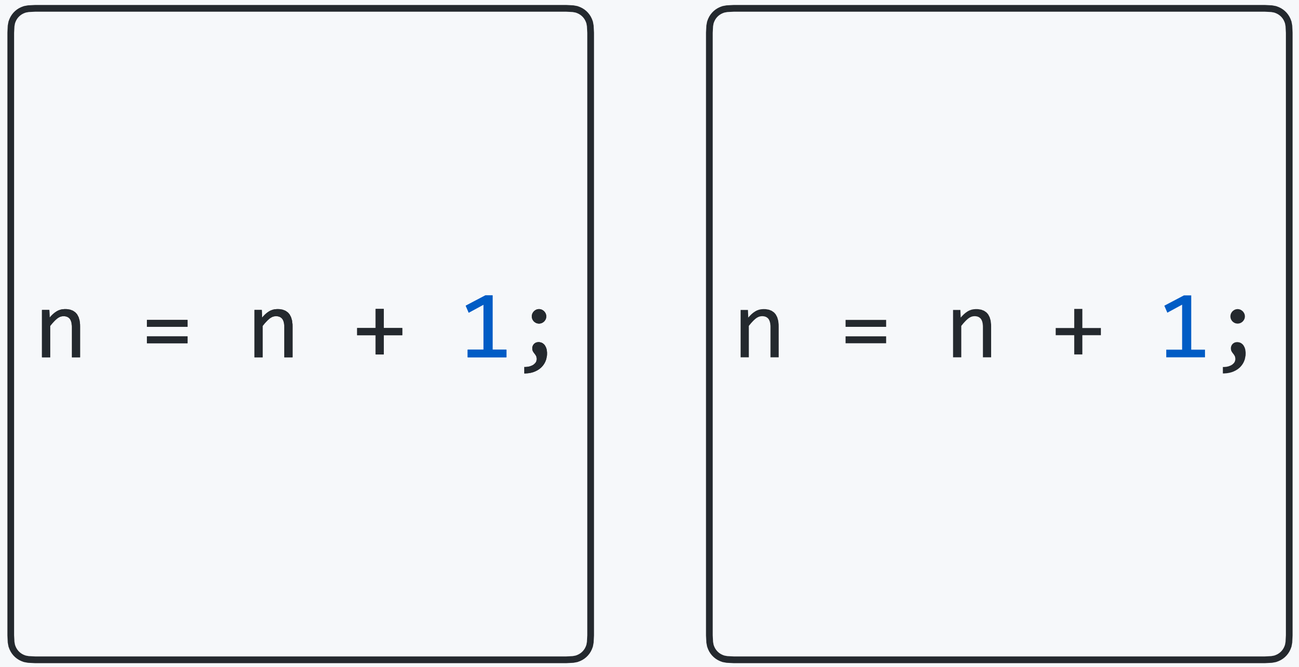
동시성 프로그램 버그의 유형별 빈도



Sara Abbaspour Asadollah et al., Concurrency bugs in open source software: a case study, Journal of Internet Services and Applications

데이터 경쟁

```
int n = 0;
```



C의 락 API

```
int n = 0;
pthread_mutex_t m = …;

pthread_mutex_lock(&m);
n = n + 1;
pthread_mutex_unlock(&m);
```

데이터와 락의 불일치로 인한 데이터 경쟁

```
int n1 = 0, n2 = 0;
pthread_mutex_t m1 = …, m2 = …;

?? pthread_mutex_lock(&m2);
   n1 = n1 + 1;
?? pthread_mutex_unlock(&m2);
```

프로그램 흐름과 락의 불일치로 인한 데이터 경쟁

```
int n = 0;
pthread_mutex_t m = …;

n = n + 1;
?? pthread_mutex_lock(&m);
…
pthread_mutex_unlock(&m);
```

프로그램 흐름과 락의 불일치로 인한 데이터 경쟁

```
   int n = 0;
   pthread_mutex_t m = …;

   pthread_mutex_lock(&m);
   …
?? pthread_mutex_unlock(&m);
   n = n + 1;
```

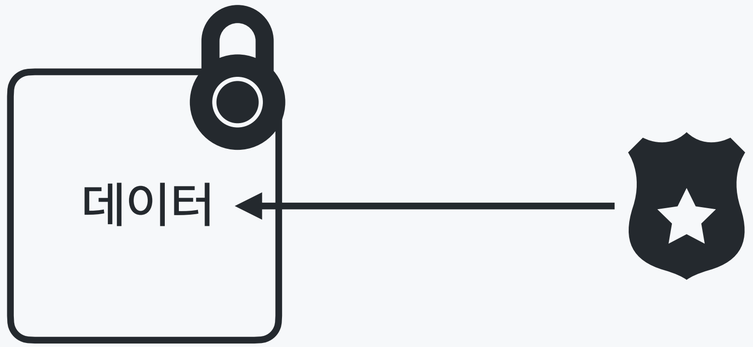
러스트의 락 API

```
static m = Mutex::new(0);
```

데이터와 락의 관계가 명시적

러스트의 락 API

```
static m = Mutex::new(0);

let guard = m.lock();
*guard = *guard + 1;
drop(guard);
```



프로그램 흐름과 락의 관계가 명시적

프로그램 흐름과 락의 불일치

```
static m = Mutex::new(0);

let guard;
*guard = *guard + 1;
?? guard = m.lock();
…
drop(guard);
```

컴파일러가 프로그램 흐름과 락의 불일치를 방지

```
static m = Mutex::new(0);

let guard;
*guard = *guard + 1;
?? guard = m.lock();
…
drop(guard);
```

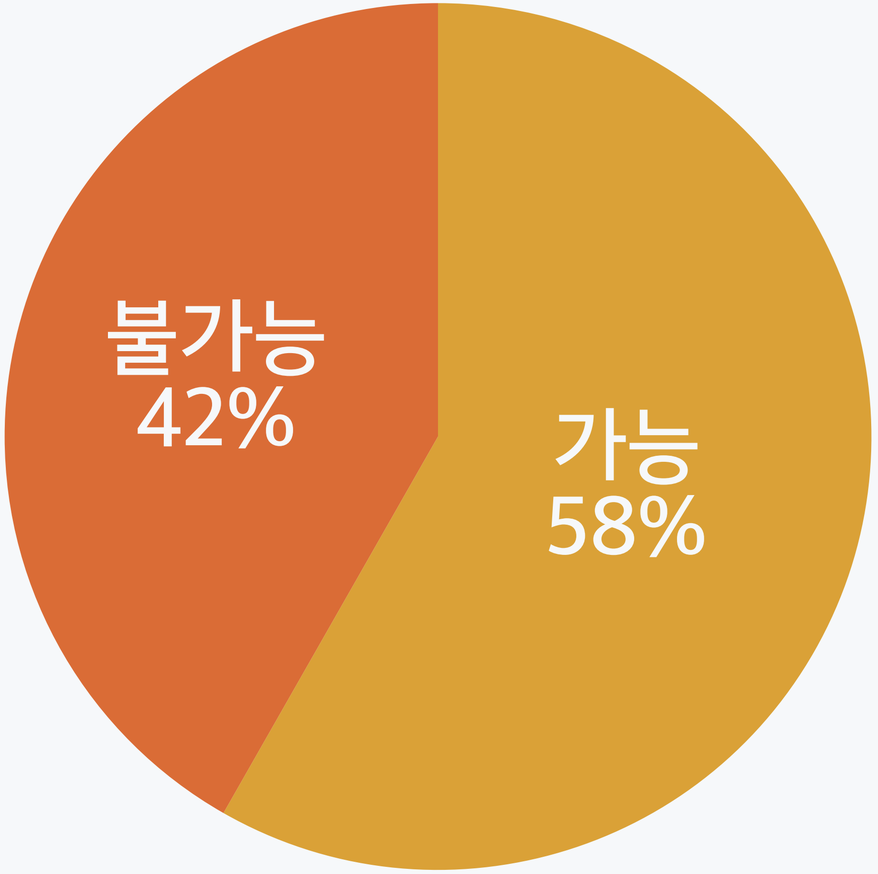
프로그램 흐름과 락의 불일치

```
static m = Mutex::new(0);

let guard = m.lock();
…
?? drop(guard);
*guard = *guard + 1;
```

컴파일러가 프로그램 흐름과 락의 불일치를 방지

```
static m = Mutex::new(0);

let guard = m.lock();
…
?? drop(guard);
*guard = *guard + 1;
```

러스트가 cURL의 버그를 얼마나 많이 막을 수 있었을까?



Tim Hutt, Would Rust secure cURL?, https://blog.timhutt.co.uk/curl-vulnerabilities-rust/

DEVELOPMENT

Linux 6.1 Officially Adds Support for Rust in the Kernel

LIKE DISCUSS

DEC 20, 2022 • 1 MIN READ

by

Sergio De Simone

FOLLOW

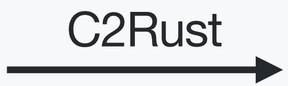
After over two years in development, support for using Rust for kernel development has entered a stable Linux release, Linux 6.1, which became available a couple of weeks ago.

Previous to its official release, Rust support has been available in linux-next, the git tree resulting from merging all of the developers and maintainers trees, for over a year. With the stable release, Rust has become the second language officially accepted for Linux kernel development, along with C.

Initial Rust support is just the absolute minimum to get Rust code building in the kernel, say Rust for Linux maintainers. This possibly means that Rust support is not ready yet for prime-time development and that a number of changes at the infrastructure level are to be expected in coming releases. Still, there has been quite some work work going on on a few actual drivers that should become available in the next future. These include a Rust nvme driver, a 9p server, and Apple Silicon GPU drivers.

기존의 자동 번역

```c
int n = 0;
pthread_mutex_t m = …;

pthread_mutex_lock(&m);
n = n + 1;
pthread_mutex_unlock(&m);
```



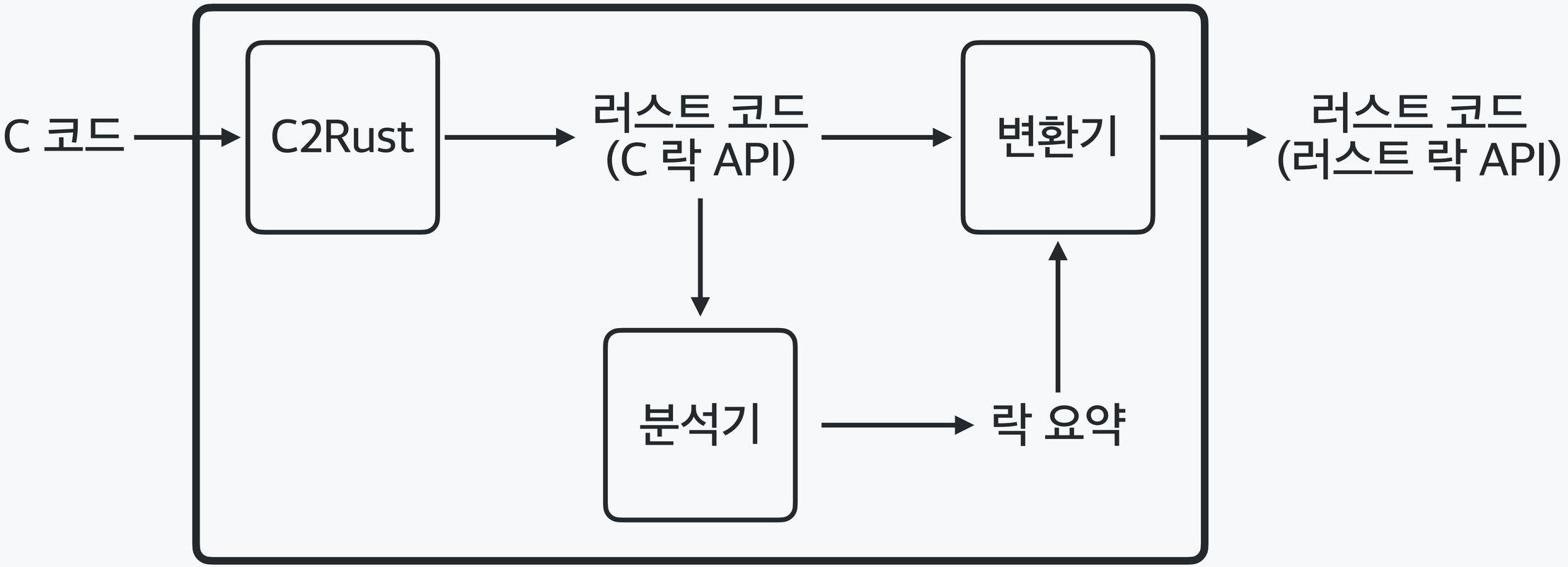

```c
static n = 0;
static m = …;

pthread_mutex_lock(&m);
n = n + 1;
pthread_mutex_unlock(&m);
```

이상적인 자동 번역

```c
int n = 0;
pthread_mutex_t m = …;

pthread_mutex_lock(&m);
n = n + 1;
pthread_mutex_unlock(&m);
```

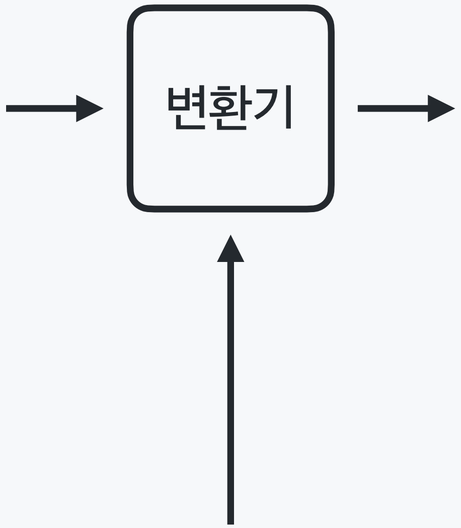
```rust
static m = Mutex::new(0);

let guard = m.lock();
*guard = *guard + 1;
drop(guard);
```

Concrat의 구조



변환기 작동 예시

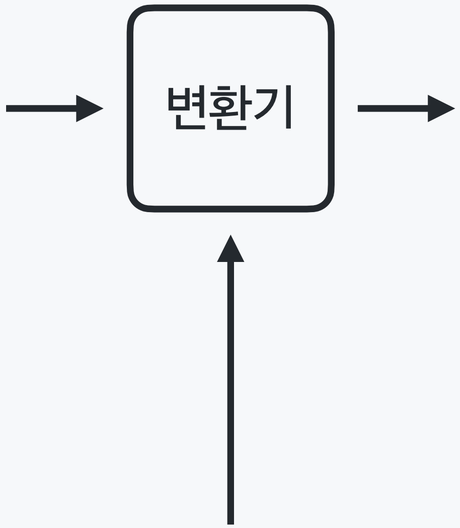
```
static n = 0;
static m = …;



pthread_mutex_lock(&m);
n = n + 1;
pthread_mutex_unlock(&m);
```




```
static m = Mutex::new(0);


let m_guard;
m_guard = m.lock();
*m_guard = *m_guard + 1;
drop(m_guard);
```

변환기 작동 예시?

```
fn inc() {
  n = n + 1;
}



pthread_mutex_lock(&m);
inc();
pthread_mutex_unlock(&m);
```




```
fn inc() {
  *guard = *guard + 1;
}


let guard;
guard = m.lock();
inc();
drop(guard);
```

??

변환기 작동 예시

```
fn inc() {
  n = n + 1;
}



pthread_mutex_lock(&m);
inc();
pthread_mutex_unlock(&m);
```

m이 n을 보호
inc를 호출할 때, m이 잡혀 있음
inc가 반환할 때, m이 잡혀 있음

변환기 작동 예시

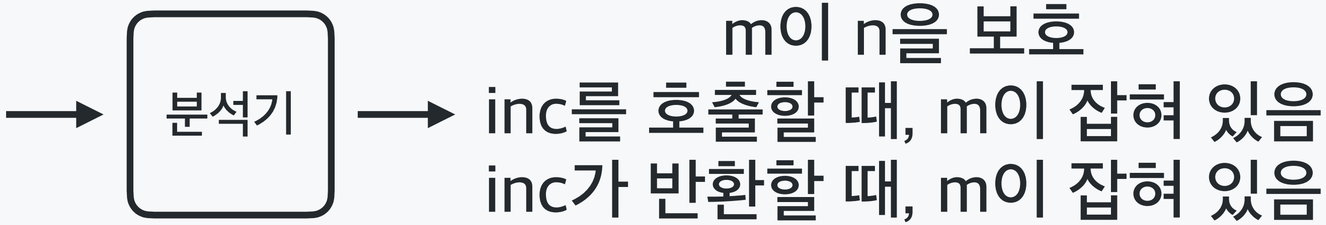
```
fn inc() {
  n = n + 1;
}



pthread_mutex_lock(&m);
inc();
pthread_mutex_unlock(&m);
```


```
fn inc(guard: Guard<i32>) {
  *guard = *guard + 1;
  guard
}
let guard;
guard = m.lock();
guard = inc(guard);
drop(guard);
```

m이 n을 보호
inc를 호출할 때, m이 잡혀 있음
inc가 반환할 때, m이 잡혀 있음

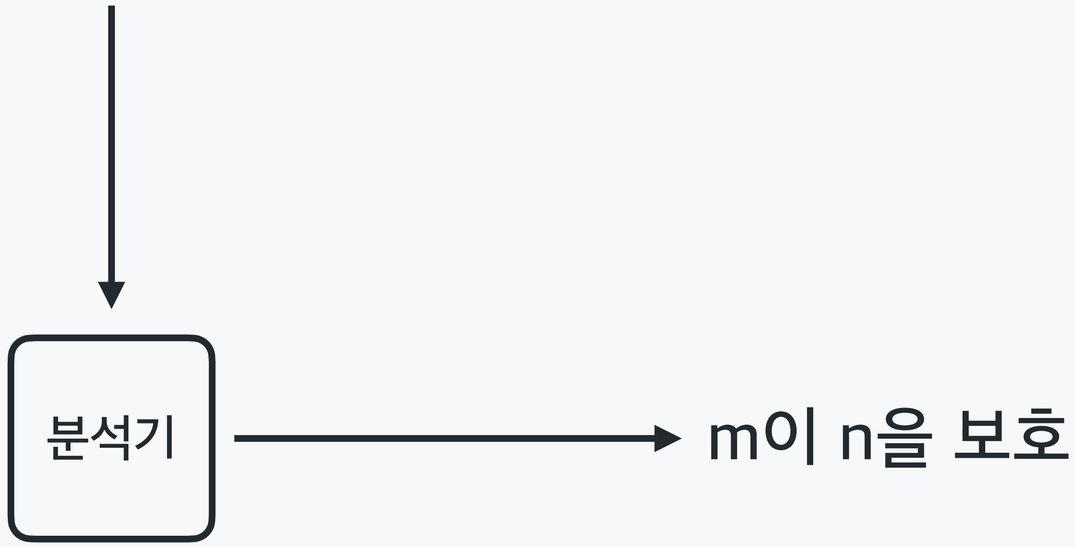
분석기의 역할

```
fn inc() {
  n = n + 1;
}



pthread_mutex_lock(&m);
inc();
pthread_mutex_unlock(&m);
```



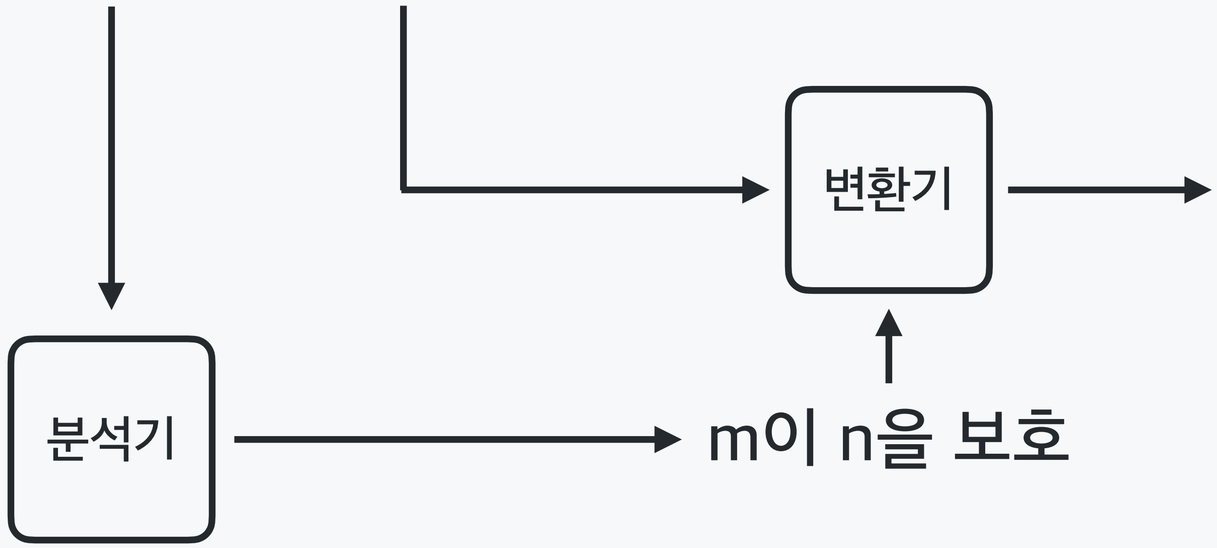
분석기의 정확도

```
if b { pthread_mutex_lock(&m); }
…
if b { n = n + 1; }
…
if b { pthread_mutex_unlock(&m); }
```



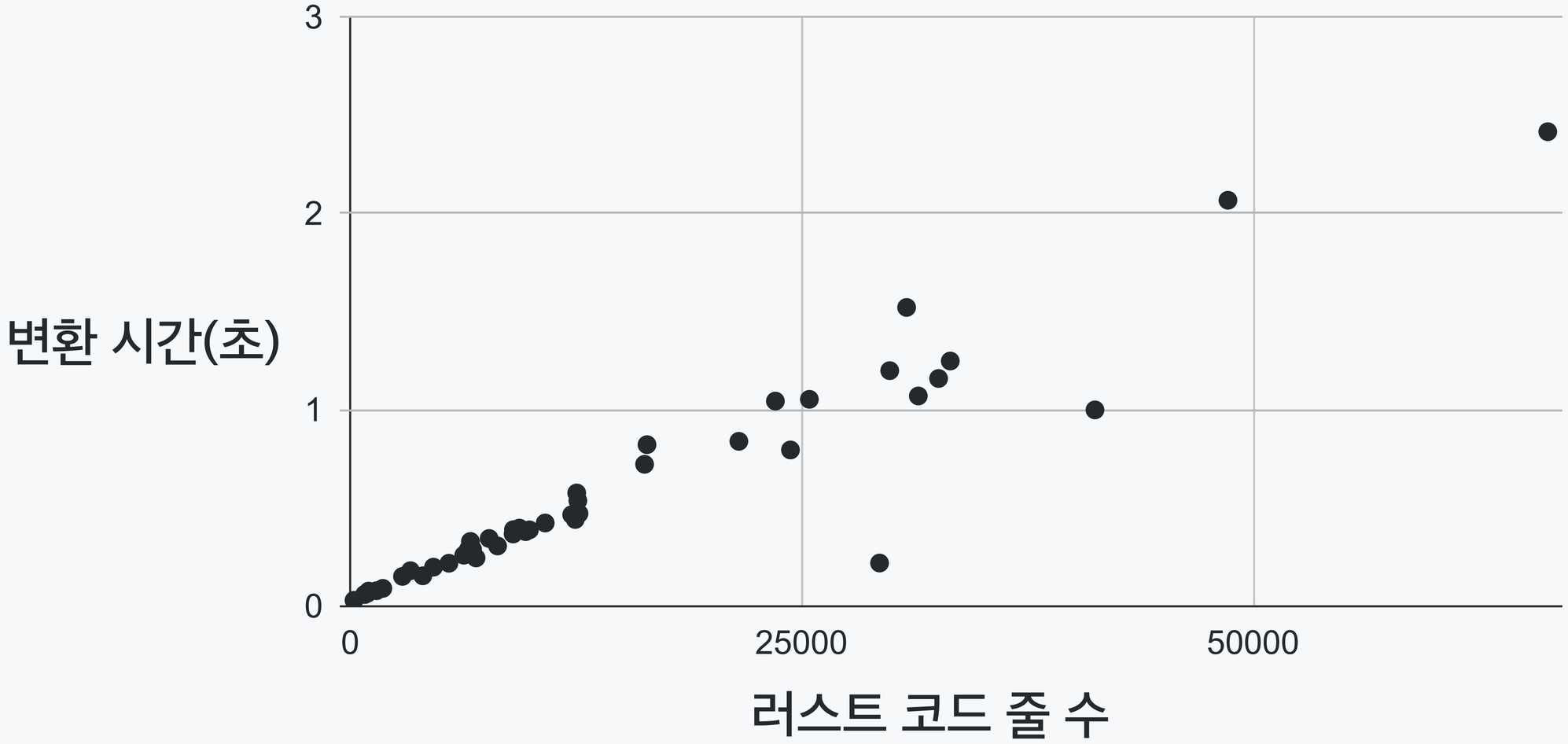
분석기의 정확도

```
if b { pthread_mutex_lock(&m); }
…
if b { n = n + 1; }
…
if b { pthread_mutex_unlock(&m); }
```

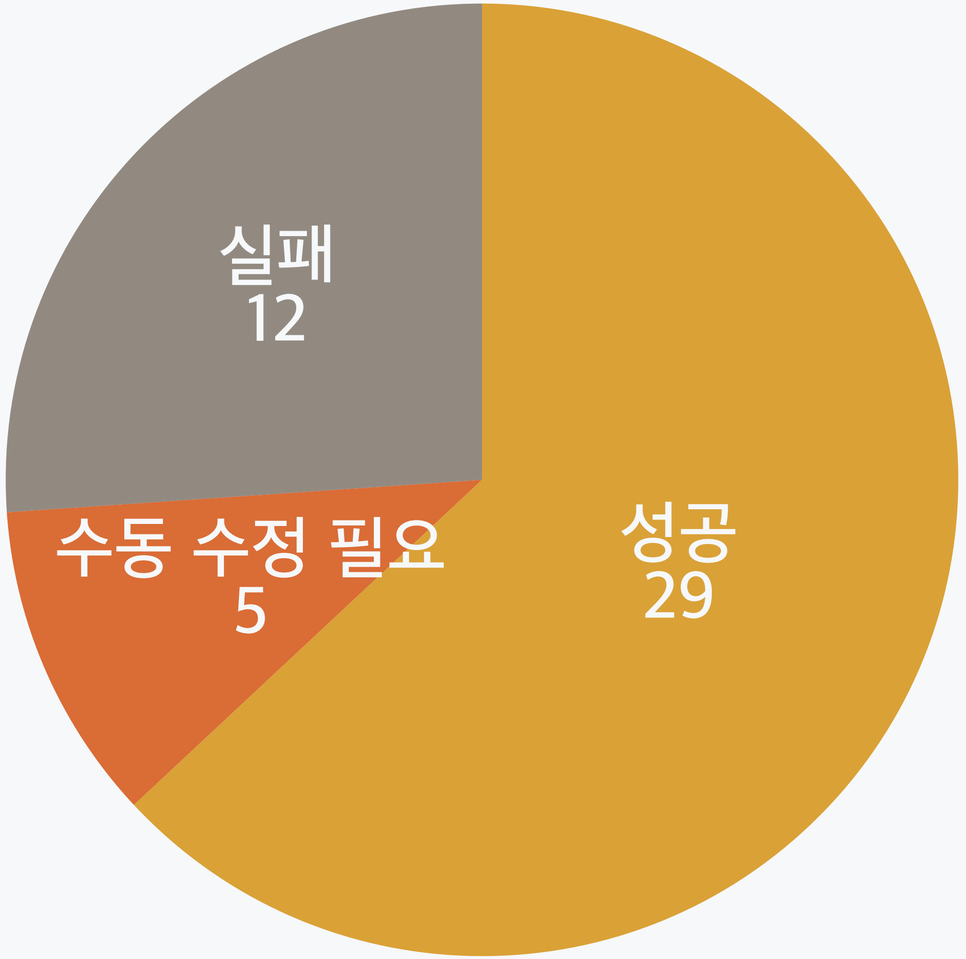



```
let guard;
if b { guard = m.lock(); }
…
if b { *guard = *guard + 1; }
…
if b { drop(guard); }
```

얼마나 효율적으로 프로그램을 변환하는가?



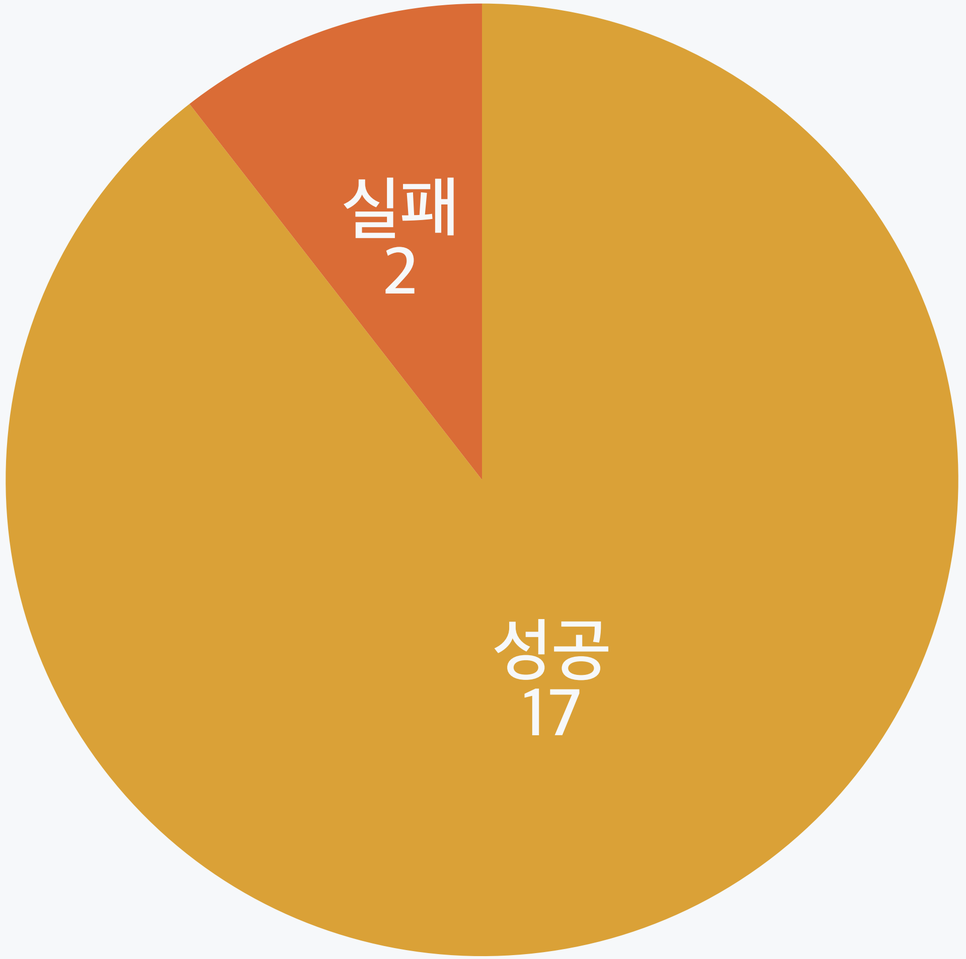
얼마나 많은 프로그램을 컴파일 가능한 코드로 변환하는가?



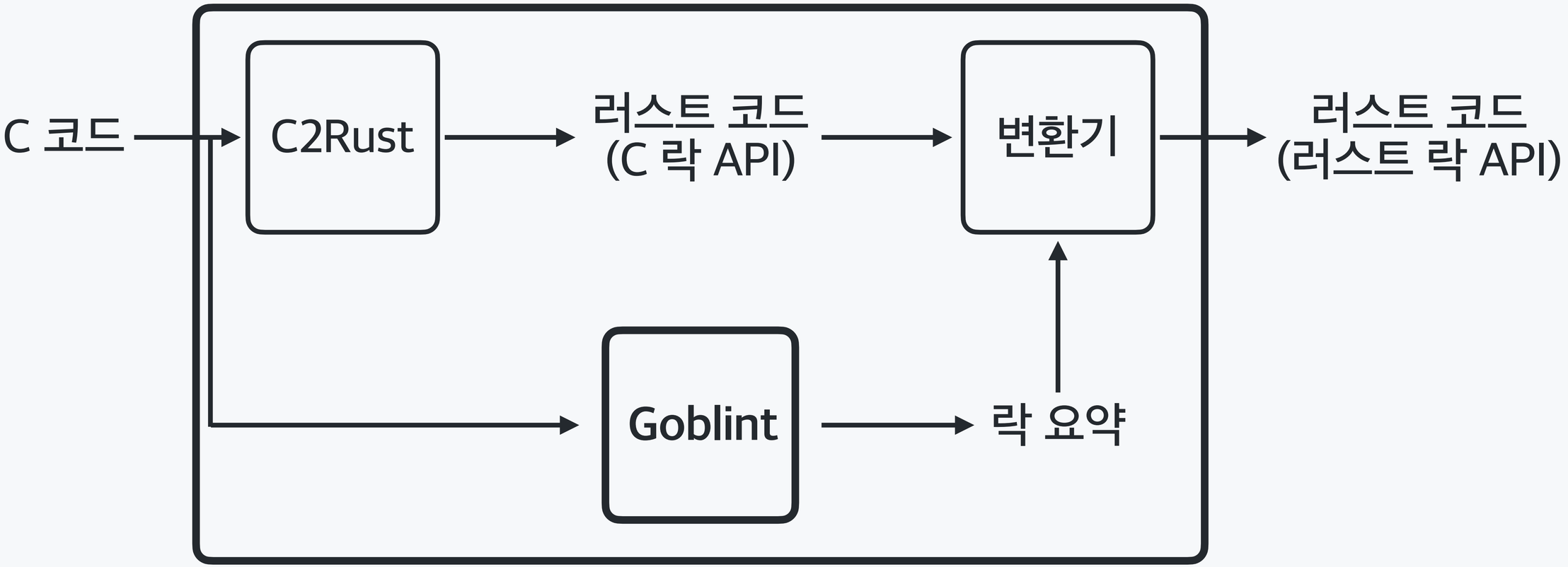
자동 변환을 통해 실제 C 프로그램에서 발견한 버그

```
pthread_mutex_lock(&k->lock);
if (strm_funcall(…) == STRM_NG) {
  pthread_mutex_lock(&k->lock);
  return STRM_NG;
}
```

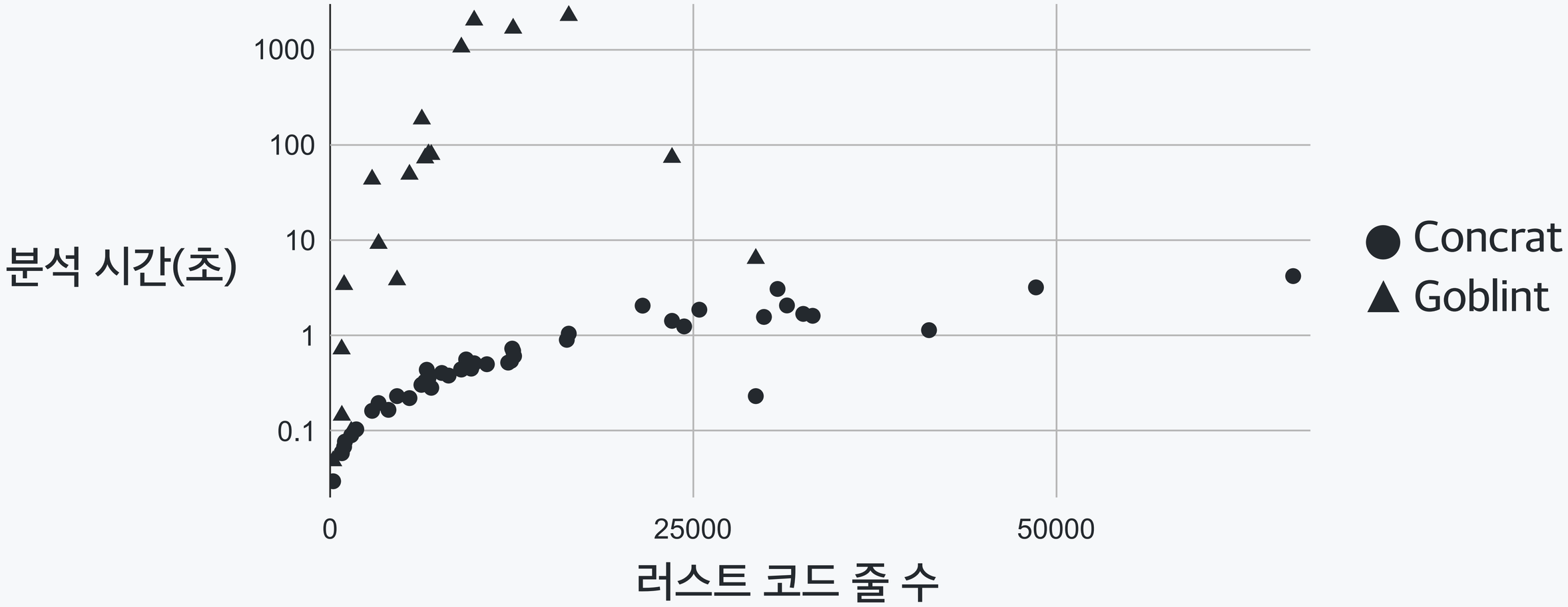
얼마나 많은 프로그램의 의미를 보존하면서 변환하는가?



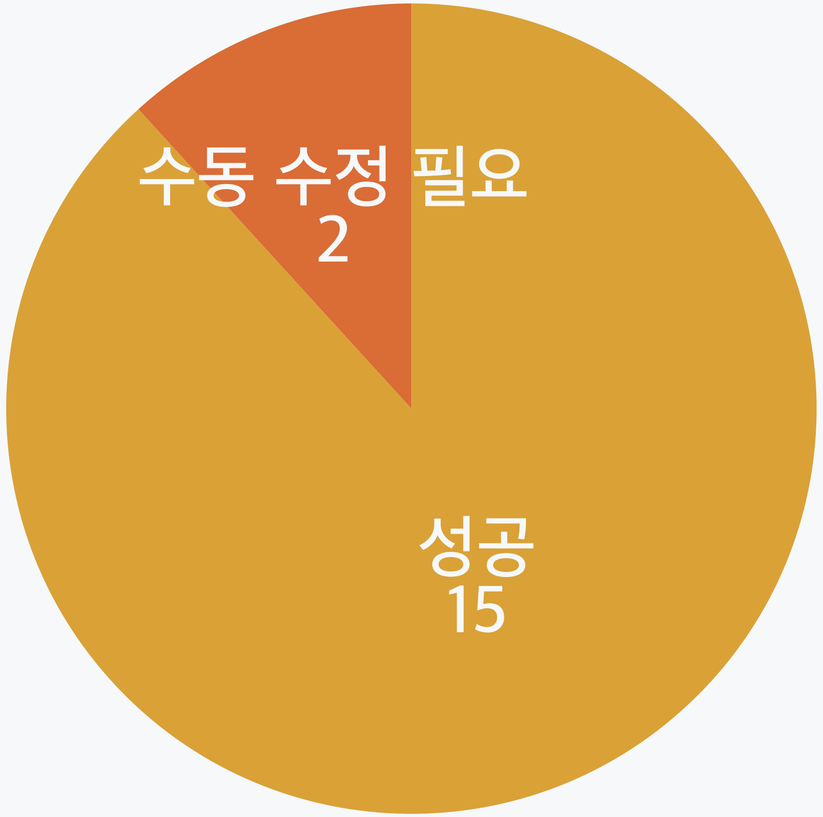
Concrat$_G$의 구조



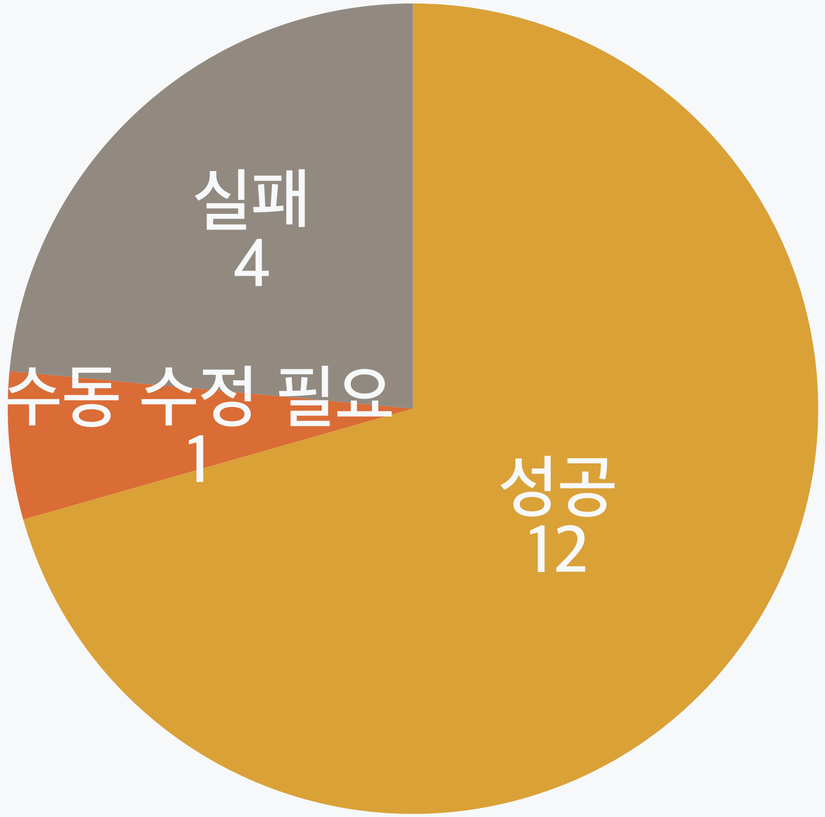
얼마나 효율적으로 프로그램을 분석하는가?



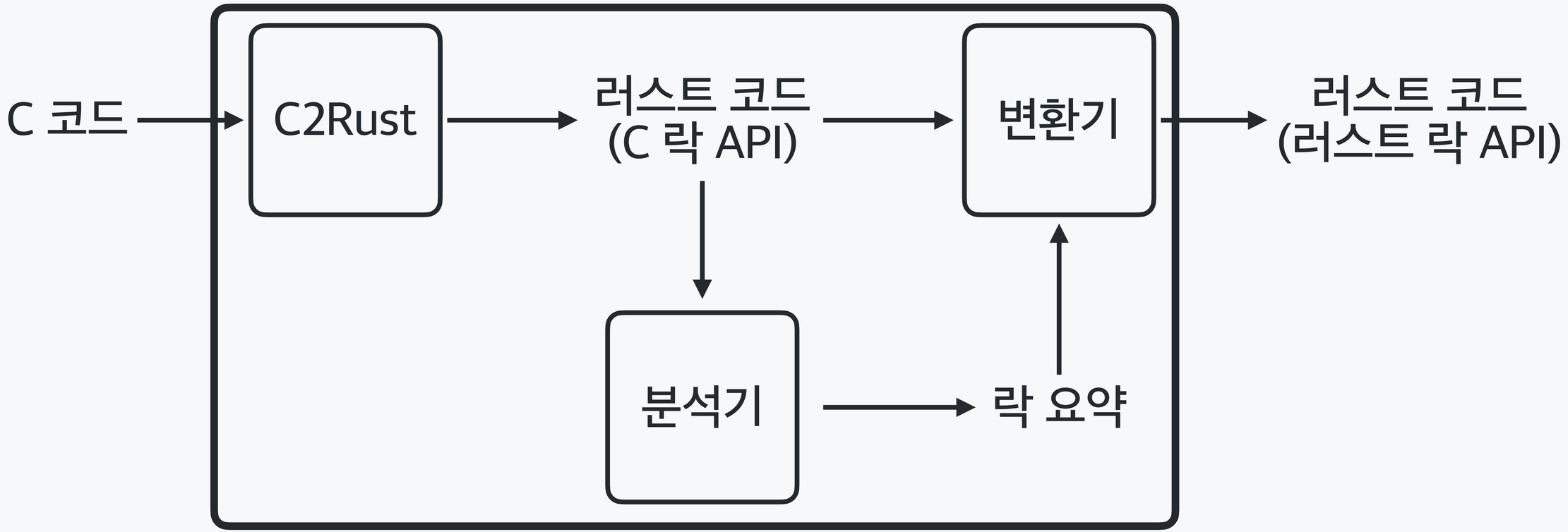
얼마나 정확하게 프로그램을 분석하는가?



Concrat



Goblint



Concrat

An Automatic C-to-Rust Lock API Translator for Concurrent Programs