Static Analysis Using Facebook Infer to Find Atomicity Violations

Dominik Harmim

Supervisor: prof. Ing. Tomáš Vojnar, Ph.D.

xharmi00@stud.fit.vutbr.cz

Brno University of Technology, Faculty of Information Technology

3rd June 2019

Motivation

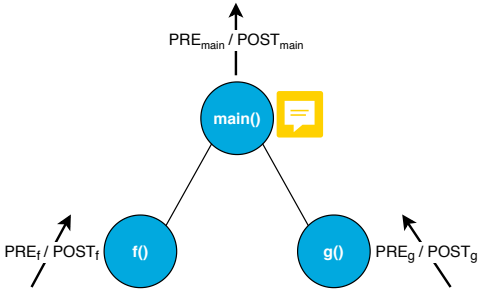
- Atomicity – indivisible series of operations.
  - Often required in concurrent programs.
  - Violation may cause significant damage.

```
void replace(int *array, int a, int b) {
    int i = index_of(array, a);
    if (i >= 0) set(array, i, b);
}
```

index_of and **set** should be ~~invoked~~ atomically

- Shortcomings of current analysers for finding atomicity violations:
  - a high rate of false positives,
  - scalability,
  - ...

- An open-source static analysis framework for interprocedural analyses.
  - Based on abstract interpretation.
  - Checks, e.g., for buffer overflows, data races, null-dereferencing, or memory leaks.
- Follows principles of compositionality.
  - Computes function summaries.
  - Incremental $\Longrightarrow$ highly scalable.
- Supports Java, C, C++ and Objective-C.



- Atomicity violations for sequences of functions.
- Sequences executed atomically once should be executed always atomically.
- Targets C/C++ programs that use PThread locks.

1. Detection of atomic sequences.

```
void f(int *array) {
    pthread_mutex_lock(&lock);
    int i = index_of(array, 42);
    if (i >= 0) set(array, i, 3);
    pthread_mutex_unlock(&lock);
}
```

summary$_f$: {(index_of set)}

2. Detection of violations of the atomic sequences.

```
void g(int *array) {
    int i = index_of(array, 66);
    if (i >= 0) set(array, i, 5);
}
```

ATOMICITY VIOLATION!

Experimental Evaluation

- The correctness was verified on hand-crafted programs.
- Evaluated on real-life low-level concurrent programs from Debian GNU Linux.
  - Analysed thousands of lines of code.
  - High false alarms ratio.

Program	Lines of Code	Atomicity Violations
glfw 2.7.9	10,230	13
alsa-utils 1.1.0	7,735	1
c-icap 0.4.2	24,923	174
npth 1.2	1,593	26
rt-tests 0.96	1,795	0
sslsplit 0.4.11	22,457	344

Summary

The static analyser for finding atomicity violations:

- Proposed and implemented as a module of Facebook Infer.
- Successfully tested and experimentally evaluated.

Future goals:

- Increase the accuracy (by, e.g., considering function parameters).
- Enhance the scalability (to be able to analyse more extensive programs).
- Extend the analysis for other types of locks.
- Create ~~the~~ Pull Request to the `master` branch of Facebook Infer.

Otázky oponenta

Diskutujte obtížnost rozšíření analyzátoru Atomer o podporu formálních parametrů funkcí a návratových hodnot.

- Využití: rozlišení kontextu volaných funkcí.
- Vhodná charakterizace hodnot staticky a abstraktně.
  - Možnost použít syntaktické „access paths“ (lokace na haldě).
  - K bližšímu rozlišení nutno použít ukazatelové analýzy.