Verified Instrumentation for Race Detection (Draft)

Abstract

Writing race-free concurrent code is notoriously difficult, and races can result in bugs that are difficult to isolate and reproduce. Dynamic race detection is often used to catch races that cannot (easily) be detected statically. One approach to dynamic race detection is to instrument the potentially racy code with operations that store and compare metadata, where the metadata implements some known race detection algorithm (e.g. vector clock race detection). In this paper, we lay out an instrumentation pass for race detection in a simple language, and present a mechanized formal proof of its correctness: all races in a program will be caught by the instrumentation, and all races detected by the instrumentation are possible in the original program.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

Keywords dynamic race detection, interactive theorem proving

1. Introduction

Our contributions are:

- Mechanized proofs of correctness of vector clock race detection and FastTrack
- An instrumentation pass that has been proved to implement vector clock race detection for a simple language

2. Race Detection Algorithms

2.1 Vector Clock Race Detection

2.2 FastTrack

3. Instrumenting a Simple Language

3.1 The Language

We define a simple multithreaded language that is just complicated enough to have races and implement race detection instrumentation.

3.2 Instrumentation

3.3 Necessary Synchronization

Completely unsynchronized instrumentation cannot be sound and complete. (example) At the same time, adding too much synchronization could significantly hurt performance.

4. Verification

We verify the correctness of the instrumentation in two steps. First, we show that a simple abstract algorithm for vector clock race detection is sound and complete. Second, we show that instrumentation records the same information and performs the same checks as the algorithm would perform on any program.

4.1 Verifying VC Race Detection

Maybe we should keep this short, since it's pretty simple.

4.2 Verifying the Instrumentation

5. Related Work

FastTrack, etc.; verified instrumentation like SoftBound

6. Conclusions and Future Work

verified FastTrack instrumentation
more faithful/detailed implementation
relaxed memory, if it matters

References