Testing Concurrent Software

Shmuel Ur

Shmuel Ur Innovation, 20164, Shorashim, Israel
shmuel.ur@gmail.com

1 Introduction

This short two hour tutorial tries to teach practical techniques for testing, review and debugging of concurrent software, based on the Dr. Ur's research and practical industrial experience in the last ten years.

The tutorial starts by explaining, in depth, why testing and debugging concurrent software is really hard, and why it is essential to do it well. A number of typical concurrent bugs are presented and explanation is given as to why finding and debugging them is going to be hard. Then classification of bugs into bug patterns is shown and every bug pattern is explained. It is demonstrated to the audience that programming concurrent code correctly requires detailed understanding of topics such as atomicity, synchronization, lock discipline, and memory layers which unfortunately is not common to people who work in the field. Some advice is given on how to program concurrent programs so as to make them less bug prone.

We briefly discussed a number of technologies for finding concurrent bugs. Those include model checking, static analysis, race detection, cloning, trace analysis, state space exploration and replay.

The tutorial's main practical testing focus is on how to use noise making techniques for revealing concurrent bugs. We show how to effectively do both unit testing and system test for concurrent programs. We explain how noise making can be used to find bugs without any change to the testing process. We then show both how to modify the testing process to take advantage of noise making capabilities as well as specific topics such as synchronization coverage, uncovering of deadlocks and concurrent debugging.

A brief introduction to the interleaving review technique, a review technique specializing in finding concurrent bugs is given. This technique has had very good success in the field and is very practical. Tool support is a secondary issue for review and while it exists, it was not covered in the tutorial.

A few advanced topics are presented. Those include how to move from testing to healing concurrent bugs, automatic pinpointing the location of bugs, and advanced deadlock detection techniques.