Testing of software with concurrent behavior

PhD Student: Cordemans, Piet$^{1,2}$  
Promoter(s): Steegmans, Eric$^2$; Boydens, Jeroen$^{1,2}$  
Assessors: Berbers, Yolande$^2$; Jacobs, Bart$^2$  
Institution(s): $^1$"KHBO" Dept of Industrial Science & Technology  
$^2$"KU Leuven" Dept of Computer Science  
Contact: piet.cordemans@khbo.be

The lock-based shared memory model has proven to be error prone and difficult to test. With the advent of multi-core programming, the exploration of alternative concurrency models has begun. However, testing concurrent software remains problematic because of two properties inherent to the concurrent execution of software. First and foremost, concurrent software exhibits non-deterministic behavior. Namely, the test result is not guaranteed to be identical for each test run. Moreover, replaying a particular test scenario requires a tool recording the specific set of thread interleavings. On the other hand, the number of possible states exponentially increases with the program size. This makes it practically impossible to cover the entire state space in a test run.

Principally, non-deterministic behavior should be isolated, which allows testing the deterministic part. More specifically, it is possible to deterministically run through different paths in the state space, a technique called state space exploration. However, due to computational restrictions, this exploration is limited to a fixed number of thread interleavings. Nevertheless, as a testing strategy it is valid under the assumption that a significant number of bugs will manifest in a limited number of thread interleavings.

The goal of this PhD research is developing test concepts for concurrent software. For this purpose a tool will be developed, which enhances conventional white box test scenarios to explore a well-defined set of paths in the state space. The primary goal is to maximize test coverage of the state space, while minimizing the overhead of exploring redundant states. Secondary metrics are the exclusion of false positive test results, while limiting the number of false negatives. Novel to this approach is the classification of patterns in specific alternative concurrent programming models. This permits partial order reduction in the state space.

Currently, research is focused on an asynchronous concurrent model, in which patterns for race conditions between an asynchronous task and its continuation are examined.

Keywords: testing, concurrent software, asynchronous, state space exploration.