Lightweight Specifications for Parallel Correctness

by

Jacob Samuels Burnim

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Koushik Sen, Chair
Professor George Necula
Professor David Wessel

Fall 2012

Lightweight Specifications for Parallel Correctness

Abstract

Lightweight Specifications for Parallel Correctness

by

Jacob Samuels Burnim

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Koushik Sen, Chair

With the spread of multicore processors, it is increasingly necessary for programmers to write parallel software. Yet writing correct parallel software with explicit multithreading remains a difficult undertaking. Though many tools exist to help test, debug, and verify parallel programs, such tools are often hindered by a lack of any specification from the programmer of the intended, correct parallel behavior of his or her software.

In this dissertation, we propose three novel lightweight specifications for the parallelism correctness of multithreaded software: semantic determinism, semantic atomicity, and nondeterministic sequential specifications for parallelism correctness. Our determinism specifications enable a programmer to specify that any run of a parallel program on the same input should deterministically produce the same output, despite the nondeterministic interleaving of the program's parallel threads of execution. Key to our determinism specifications are our proposed bridge predicates — predicates that compare pairs of program states from different executions for semantic equivalence. Our atomicity specifications use bridge predicates to generalize traditional atomicity, enabling a programmer to specify that regions of a parallel or concurrent program are, at a high-level, free from harmful interference by other threads. Finally, our nondeterministic sequential (NDSeq) specifications enable a programmer to completely specify the parallelism correctness of a multithreaded program with a sequential but nondeterministic version of the program and, further, enable a programmer to test, debug, and verify functional correctness sequentially, on the nondeterministic sequential program.

We show that our lightweight specifications for parallelism correctness enable us to much more effectively specify, test, debug, and verify the use of parallelism in multithreaded software, independent of complex and fundamentally-sequential functional correctness. We show that we can easily write determinism, atomicity, and nondeterministic sequential (NDSeq) specifications for a number of parallel Java benchmarks. We propose novel testing techniques for checking that a program conforms to its determinism, atomicity, or nondeterministic sequential specification, and we apply these techniques to find a number of parallelism errors in our benchmarks. Further, we propose techniques for automatically inferring a likely determinism or NDSeq specification for a parallel program, given a handful of representative executions.