Title

Using Formal Methods to Verify Transactional Abstract Concurrency Control

Author

Trek S. Palmer, *University of Massachusetts - Amherst* Follow

Document Type

Open Access Dissertation

Degree Name

Doctor of Philosophy (PhD)

Degree Program

Computer Science

Year Degree Awarded

Summer 2014

First Advisor

J. Eliot B. Moss

Second Advisor

Jack Wileden

Third Advisor

Neil Immerman

Fourth Advisor

George Avrunin

Subject Categories

Programming Languages and Compilers

Abstract

Concurrent application design and implementation is more important than ever in today's multi-core processor world. Transactional Memory (TM) Concurrent application design and implementation is more important than ever in today's multi-core processor world. Transactional Memory (TM). Each has its own particular advantages and disadvantages. However, these techniques each need some extra information to `glue' the non-transactional operation into a transactional context. At the most general level, non-transactional code must be decorated in such a way that the TM run-time can determine how those non-transactional operations commute with one another, and how to `undo' the non-transactional operations in case the run-time needs to abort a software transaction. The TM run-time trusts that these programmer-provided annotations are correct. Therefore, if an implementor needs to employ one of these transactional `escape hatches', it is crucially important that their concurrency control annotations be correct. However, reasoning about the commutativity of data structure operations is often challenging, and increasing the burden on the programmer with a proof requirement does not simplify the task of concurrent programming. There is a way to leverage the structure that these TM extensions require to reduce greatly the burden on the programmer. If the programmer could describe the abstract state of the data structure and then reason about it with as much machine assistance as possible, then there would be much less opportunity for error. Abstract state is preferable to a more concrete state, because it permits the programmer to use different concrete implementations of the same abstract data type. Also, some TM extensions such as open nesting can handle concrete state conflicts without programmer intervention (making the abstract state the appropriate state for reasoning about commutativity). A solution to the problem of specifying and verifying the concurrency properties of abstract data structures is the subject of this thesis. We will describe a new language, ACCLAM, for describing the abstract state of a data structure and reasoning about its concurrency control properties. This thesis also describes a tool that can process ACCLAM descriptions into a machine verifiable form (they are converted to a SAT problem). We will also provides a more detailed overview of transactional memory and the more popular extensions, a detailed semantic description of ACCLAM and a set of example data structure models and the results of processing those examples with the language processing tool.

Recommended Citation

Palmer, Trek S., "Using Formal Methods to Verify Transactional Abstract Concurrency Control" (2014). *Doctoral Dissertations*. 237.
https://scholarworks.umass.edu/dissertations_2/237