Peterson, Christina; Cook, Victor; Dechev, Damian
Concurrent correctness in vector space. (English) [Zbl 1472.68092]

Summary: Correctness verification of a concurrent history is challenging and has been proven to be an NP-complete problem. The reason that verifying correctness cannot be solved in polynomial time is a consequence of the way correctness is defined. Traditional correctness conditions require a concurrent history to be equivalent to a legal sequential history. The worst case number of legal sequential histories for a concurrent history is $O(n!)$ with respect to $n$ methods invoked. Existing correctness verification tools improve the time complexity by either reducing the size of the possible legal sequential histories or improving the efficiency of generating the possible legal sequential histories. Further improvements to the time complexity of correctness verification can be achieved by changing the way correctness of concurrent programs is defined. In this paper, we present the first methodology to recast the correctness conditions in literature to be defined in vector space. The concurrent histories are represented as a set of method call vectors, and correctness is defined as properties over the set of vectors. The challenge with defining correctness in vector space is accounting for method call ordering and data structure semantics. We solve this challenge by incorporating a priority assignment scheme to the values of the method call vectors. Using our new definitions of concurrent correctness, we design a dynamic analysis tool that checks the vector space correctness of concurrent data structures in $O(n^2)$ with respect to $n$ method calls, a significant improvement over $O(n!)$ time required to analyze legal sequential histories. We showcase our dynamic analysis tool by using it to check the vector space correctness of a variety of queues, stacks, and hashmaps.

For the entire collection see [Zbl 1471.68017].

MSC:
68Q60 Specification and verification (program logics, model checking, etc.)
68N19 Other programming paradigms (object-oriented, sequential, concurrent, automatic, etc.)
68Q85 Models and methods for concurrent and distributed computing (process algebras, bisimulation, transition nets, etc.)

Keywords:
concurrent programs; correctness condition; dynamic analysis

Full Text: DOI

References:


[40] Vechev, M.; Yahav, E.; Yorsh, G.; Păsăreanu, CS, Experience with model checking linearizability, Model Checking Software, 261-278 (2009), Heidelberg: Springer, Heidelberg. doi:10.1007/978-3-642-02652-2_1


This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.