Correctness Conditions for Randomized Shared Memory Algorithms

Philipp Woelfel

Department of Computer Science, University of Calgary, Canada

In an asynchronous shared memory system, processes communicate by applying operations on shared *base* objects. From an algorithm designer's perspective it is ideal if the operations on these objects are *atomic*, meaning that each such operation happens instantaneously. However, objects provided by systems are typically not truly atomic, and neither are objects implemented from base objects. As a result, if multiple processes concurrently execute methods on such objects, the set of all possible outcomes is difficult to predict.

For almost two decades, *linearizability*, defined by Herlihy and Wing [4], has been the gold standard among correctness conditions for non-atomic objects. It guarantees that any possible result that can arise from an interleaving of processes using linearizable operations could arise if the operations were atomic. Hence, the worst-case behaviour of algorithms can be analyzed under the assumption that all operations are atomic, even when they are only linearizable. For that reason, the terms linearizability and atomicity have often been used interchangeably (see for example [5]).

Golab, Higham, and Woelfel [2] observed that linearizable implementations do not preserve the probability distribution of the possible results if we replace atomic objects used in a *randomized* algorithm with implemented ones. An *adversary*, which schedules process steps, can "stretch out" a method call that was originally an atomic operation, and inspect the outcome of other processes coin flips before allowing the method call to be completed. As a result, replacing an atomic object with a linearizable one in a randomized algorithm amounts to increasing the power of the adversary. In order to be able to employ the power of randomization in shared memory algorithms, we need to devise new correctness conditions that eliminate the deficiencies of linearizability. In this talk the state of the art [1–3] of finding such correctness conditions will be presented.

References

1. O. Denysyuk and P. Woelfel. Wait-freedom is harder than lock-freedom under strong linearizability, 2015. Manuscript (submitted).
2. W. Golab, L. Higham, and P. Woelfel. Linearizable implementations do not suffice for randomized distributed computation. In *Proc. of 43rd ACM STOC*, pp. 373–382. 2011.
3. M. Helmi, L. Higham, and P. Woelfel. Strongly linearizable implementations: possibilities and impossibilities. In *Proc. of 31st PODC*, pp. 385–394. 2012.
4. M. Herlihy and J. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12:463–492, 1990.
5. N. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., 1996.