A

Erratum: Limited-Use Atomic Snapshots with Polylogarithmic Step Complexity

JAMES ASPNES, Yale University
HAGIT ATTIYA, Technion
KEREN CENSOR-HILLEL, Technion
FAITH ELLEN, University of Toronto

This is an erratum for the paper "Limited-Use Atomic Snapshots with Polylogarithmic Step Complexity" published in J. ACM 62(1): 3:1-3:22 (2015). The implementation of a $\texttt{MaxArray}_{k\times h}$ object in Algorithm 2 does not guarantee linearizability. We give here a simple correction to the algorithm and its correctness proof.

1. THE ERROR AND ITS CORRECTION

An example of the error: Suppose that, from the initial configuration, process p_1 invokes a $\texttt{MaxScan}(r)$ operation, op_1, gets 0 when it performs $\texttt{ReadMax}(r.second)$ and assigns 0 to x in Line 11, and reads 0 from $r.switch$ in Line 12. Then, let process p_2 invoke and complete a $\texttt{MaxUpdate1}(r, v_2)$ operation, op_2, with $v_2 > 0$. Afterwards, let process p_3 invoke a $\texttt{MaxUpdate0}(r, v_3)$ operation, op_3, with $0 < v_3 < m$. Since op_2 finishes before op_3 begins, op_2 must be linearized before op_3. Therefore, any $\texttt{MaxScan}(r)$ operation that returns v_3 for component 0 must return at least v_2 for component 1. Finally, let p_1 complete its invocation of op_1. Since p_1 accesses $r.\mathsf{left}$ only after op_3 has been completed, op_1 returns v_3 for component 0. Since p_1 sets x to 0 before op_2 starts, op_1 returns 0 for component 1. This means that the original implementation is not linearizable.

The error in the proof: The error is in the proof of Theorem 3.4. This proof shows that any two $\texttt{MaxScan}(r)$ operations return comparable pairs of values, which implies that any two such operations can be linearized correctly with respect to each other and that any $\texttt{MaxUpdate0}(r, v)$ and $\texttt{MaxUpdate1}(r, v)$ operation can be linearized correctly with respect to any $\texttt{MaxScan}(r)$ operation. However, the proof does not address the linearization of $\texttt{MaxUpdate0}(r, v)$ operations with respect to $\texttt{MaxUpdate1}(r, v)$ operations, which is exactly where the above example fails.

The correction: Forcing each $\texttt{MaxUpdate0}(r, v)$ to perform an embedded $\texttt{MaxScan}(r)$ operation after its $\texttt{WriteMax}(r.\mathsf{second}, v)$ operation overcomes the problem described above. The corrected algorithm has a new line, 9.5, in which $\texttt{MaxScan}(r)$ is invoked.

Linearizability: To show linearizability, first notice that Lemmas 3.1, 3.2 and 3.3 of the original proof remain intact. The first 6 paragraphs of the proof of Theorem 3.4 remain the same, showing that $\texttt{MaxScan}(r)$ operations can be linearized correctly with respect to each other. Then, we modify the linearization of a $\texttt{MaxUpdate1}(r, v)$ operation so that it is linearized after its invocation and immediately before any (perhaps embedded) $\texttt{MaxScan}(r)$ operation returns a value greater than or equal to v for component 1. Finally, $\texttt{MaxUpdate0}(r, v)$ operations are linearized correctly with respect to $\texttt{MaxScan}(r)$ operations, which also implies that they are linearized correctly with respect to $\texttt{MaxUpdate1}(r, v)$ operations.

Step complexity: A $\texttt{MaxUpdate1}(r, v)$ operation now takes $O(\log k \log h)$ steps instead of $O(\log(h))$, due to the embedded $\texttt{MaxScan}$ operation on a $\texttt{MaxArray}_{k\times h}$ object, which takes $O(\log k \log h)$ steps.