Exercises: Implementation techniques for libraries of transactional concurrent data types

Liuba Shrira

June 26, 2017

Question

The lecture discussed two specifications for the UniqueId object. One where unique ids are successive integers, and another where they are uninterpreted unique values. Show two transactions T1 and T2 containing uniqueID object operations, and a single serializable execution where T1 and T2 running concurrently block or abort using one specification but complete without blocking or aborting using the other. Explain.

Question

A Set is a collection of items without duplicates. Consider a linearizable implementation of a Set of integers object with *add(x), remove(x)*, and *contains(x)* operations. A call to add() or remove() returns a Boolean indicating whether the set was modified. A call to contains(x) returns a Boolean indicating whether x is in the Set.

The lecture described the boosting approach for implementing a transactional Set object using a linearizable Set object. Boosting requires every transactional Set operation to have an inverse operation. Give an inverse for each transactional Set operation using the linearizable Set object operations. Keep in mind that the inverse of a call often depends on its result.

Question

Consider the following specification of a PriorityQueue object. The *add (x)* operation inserts an integer element value into the set, allowing duplicates.

The *Integer: removeMin* operation removes and returns the smallest element of the set, or null if empty.

Question A: explain when *add* and *removeMin* operations commute?

Question B: describe changes needed to PriorityQueue operations to apply Boosting to create a transactional PriorityQueue object.

Question

The lecture described STO, an alternative approach to Boosting, that allows to improve the performance of transactional objects. Explain the repeated lookup problem that arises in optimistic approaches and describe STO solution.