Concurrent Algorithms December 12, 2016

Exercise 9

Problem 1.

Each of the following executions represents an interleaving of transactions executed by a Transactional Memory object. For each execution:

- Specify whether it is *opaque* or not.
- If it is not, suggest a modification to make it *opaque*.
- Specify an equivalent sequential execution of transactions.

Reminder: An execution is *opaque* if it is equivalent to some sequential execution in which every transaction, even aborted, observes a consistent state of the memory. A transaction T in a sequential execution observes a consistent state of the memory if for every transactional variable x, every read operation on x within the transaction returns: (I) the value written by the last write operation on x in the transaction T, or (II) the value written by the last write operation on x within a committing transaction (before T), if there are no write operations on x in T, or (III) the initial value of x if there are no write operations on x within the execution (before T).

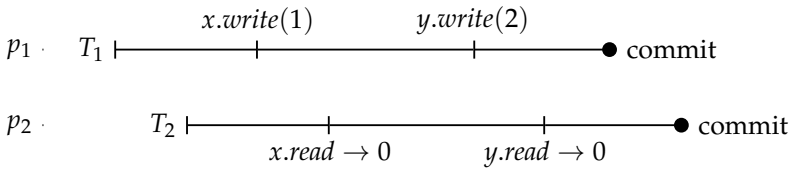

Figure 1: Transactional executions 1.

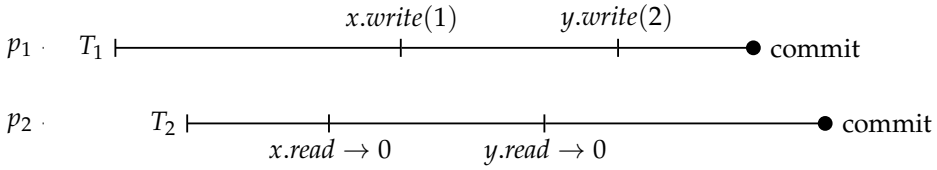

Figure 2: Transactional executions 2.

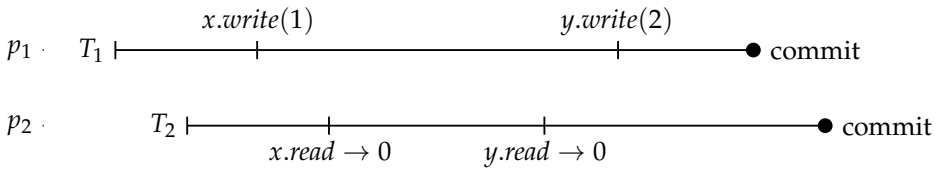

Figure 3: Transactional executions 3.

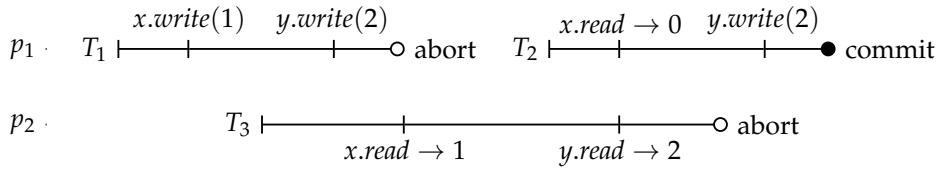

Figure 4: Transactional executions 4.

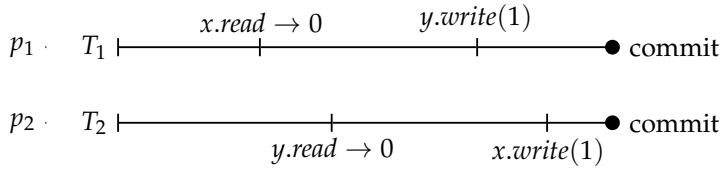

Figure 5: Transactional executions 5.

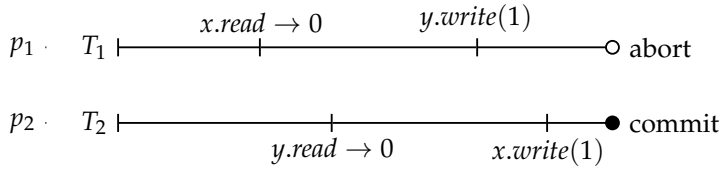

Figure 6: Transactional executions 6.

Problem 2.
Implement following objects using transactional memory:

- A snapshot.
- A strong counter.
- A compare-and-swap that works on several locations in an array. It takes the indices of the locations, expected old values and the new values as parameters. Only if all the locations in the array have the expected values, it swaps them with the new ones.