Resit Concurrency & Multithreading

VU University Amsterdam, 14 January 2015, 18:30-21:15

(At this exam, you may use the textbook of Herlihy and Shavit. Answers can be given in English or Dutch. Use of the slides or a laptop is not *allowed.)*

(The exercises in this exam sum up to 90 points; each student gets 10 points bonus.)

1. Suppose the FIFO queue class is augmented with a `peek()` method that returns but does not remove the first element in the queue. Show that this augmented queue has infinite as consensus number. (10 pts)

2. Consider the construction of an atomic MRSW from atomic SRSW registers. Suppose that the atomic SRSW registers are replaced with regular SRSW registers. Give an example to show that then the construction no longer yields an atomic MRSW register. (12 pts)

3. In the MCS queue lock, a thread that wants the lock, first sets `locked` of its own node to *true*, and then redirects `next` of its predecessor to its own node. Suppose this order would be reversed. Give a scenario to show that then the MCS queue lock could deadlock. (10 pnts)

4. Explain for the following two statements what could go wrong, and how it can be improved.

   (a) `if (x > 0) condition.await();`

   (b) `if (x == 0) condition.signal();` (9 pts)

5. Give a scenario for the optimistic synchronization implementation of list-based sets in which a thread is forever attempting to delete a node. (10 pts)

6. Consider the unbounded lock-free stack in Section 11.2 of the textbook. Give a scenario to show that in the absence of proper garbage collection, the ABA problem could arise in the `pop()` method. Also explain how (the implementation of) the unbounded lock-free stack could be adapted to avoid this ABA problem. (12 pts)

7. Give a highly parallel multithreaded algorithm for multiplying an $n \times n$ matrix by a length-n vector, which achieves work $\Theta(n^2)$ and critical path $\Theta(\log^2 n)$. Analyze the work and critical-path length of your algorithm. (15 pts)

8. Give a (Java pseudocode) implementation of `pop()` and `push()` methods for a bounded transactional stack. (12 pts)