The University of New South Wales

COMP3151/9154
Foundations of Concurrency

Final Exam

Term 2, 2022

Time Allowed: 24 Hours. Submit by 8AM Sydney time on August 23.

Total Marks Available: 100

Brief answers are **strongly** preferred. Extreme verbosity may cost marks.

Produce a typeset PDF file, via LaTeX or otherwise, with your answers.

Submit with `give cs3151 exam exam.pdf` or with the `give` web interface.

The exam is open-book. You may read anything you like, and in general use any passive resource.

You **may not** use active resources—don't solicit, offer, or accept help of any kind, with one exception: you may ask private questions on Ed. Johannes will monitor Ed regularly, except when he sleeps (around 10PM–6AM).

Include the following statement in your PDF file:
I declare that all of the work submitted for this exam is my own work, completed without assistance from anyone else.

You must adhere to the UNSW student conduct requirements listed at `https://student.unsw.edu.au/conduct`.

bool $x, y :=$ false	
forever do	**forever do**
p_1 *non-critical section*	q_1 *non-critical section*
p_2 $x :=$ true	q_2 $y :=$ true
p_3 **await** $\neg y$	q_3 **if** x **then**
p_4 ***critical section***	q_4 $y :=$ false
p_5 $x :=$ false	q_5 **await** $\neg x$
	q_6 **goto** q_2
	q_7 ***critical section***
	q_8 $y :=$ false

Table 1: A critical section algorithm.

Part I

Question 1 (10 marks)

Answer the following, briefly and in your own words.

(a) (2 marks) What is a linear-time property?

(b) (2 marks) Why does Owicki-Gries require interference freedom checks?

(c) (2 marks) How do distributed consensus algorithms get around the FLP theorem?

(d) (2 marks) Why does the King algorithm require one more round than the number of traitors?

(e) (2 marks) What is the difference between a permission-based and a token-based distributed mutual exclusion algorithm?

Question 2 (15 marks)

These questions are about the critical section algorithm in Table 1. The purpose of this algorithm is to get away with the least amount of shared state possible, which turns out to be one bit per process.

(a) (5 marks) Of the four main critical section desiderata (mutual exclusion, deadlock freedom, eventual entry, and absence of unnecessary delay), which ones are satisfied by this algorithm? For any properties that are not satisfied, describe a behaviour that is in violation.

(b) (5 marks) Suppose we rewrite process p to be exactly like process q, but with the roles of x and y swapped. Would this change your answer to the previous question?

(c) (5 marks) Why are there no algorithms with only a single bit of shared state? You don't have to produce a formal proof, but try to make a convincing informal argument.

For the purposes of this question, we consider an algorithm correct if it satisfies at least mutual exclusion and deadlock freedom (in the sense "if a process is trying to enter the critical section, then assuming weak fairness, some process will eventually enter the critical section").

Question 3 (15 marks)

Assume you have an underlying monitor implementation I with priority ordering $E < S < W$. Suppose we would like a monitor M to behave as if it had priority ordering $E < W < S$. Show how to implement our desired monitor M using our underlying implementation I.

Present your solution in the form of pseudocode snippets to be executed upon monitor entry, monitor exit, signalling and waiting. That is, assume that whenever monitor M would like to execute, e.g., **waitC**(*cond*), it runs your code snippet instead. You may introduce auxiliary variables and condition variables as needed. Some of the snippets might be empty, in which case they can be left out.

$$(a.P)\backslash x = a.(P\backslash x) \quad \text{if } x \notin \{a, \overline{a}\} \qquad P\backslash x\backslash x = P\backslash x \qquad P\backslash x\backslash y = P\backslash y\backslash x$$

Table 2: Some algebraic laws of CCS.

Question 4 (10 marks)

(a) (5 marks) Consider the LTL formula $\Diamond\Box\Diamond\varphi$. Give a simpler but logically equivalent formula. Explain why it's equivalent.

(b) (5 marks) Let the CCS process P be defined as follows:

$$P = (x.P)\backslash x$$

Simplify the following CCS expression step by step:

$$(x.(P\backslash x))\backslash x$$

Hint: it simplifies a lot! A selection of algebraic laws about restriction that you may use can be found in Table 2.

Part II

These questions are about the paper *Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms* by Maged M. Michael and Michael L. Scott (PODC 1996: 267-275).

Question 5 (15 marks)

(a) (5 marks) Section 1 of the paper includes an extensive summary of the state of the art in 1996. How did Michael and Scott's paper improve the state of the art?

(b) (3 marks) What is the difference between a block-free algorithm and a wait-free algorithm? Of the two, which kind of algorithm did the specification of Assignment 1 ask for?

(c) (2 marks) The performance analysis in Section 4 found no situation where Michael and Scott's two-lock algorithm outperforms lock-free algorithms. What's the benefit of the two-lock algorithm?

(d) (5 marks) The compare-and-swap (CAS) on line E17 of the non-blocking algorithm has the accompanying comment "Try to swing Tail to the inserted node". This suggests that, if the CAS fails, an enqueue operation can terminate even if its work is unfinished. Is this a problem? Explain why or why not.

Question 6 (25 marks)

The ABA problem is discussed a lot in the paper.

(a) (4 marks) What is the ABA problem? Explain informally and in your own words.

(b) (3 marks) How does the authors' use of modification counters help mitigate the ABA problem?

(c) (5 marks) Ominously, the safety analysis in Section 3.1 is predicated upon the assumption that the ABA problem will never occur. Describe a scenario where, in the lock-free algorithm, the queue can end up corrupted if the ABA problem occurs.

Hint: make the modification counters wrap around.

int $n := 0$	
int $i := 10$	**int** $j := 10$
while $i \neq 0$	**while** $j \neq 0$
p_1 **repeat**	q_1 **repeat**
p_2 $x := n$	q_2 $y := n$
p_3 **until** $\mathsf{CAS}(n, x, (x+1) \bmod 3)$	q_3 **until** $\mathsf{CAS}(n, x, (x+1) \bmod 3)$
p_4 $i := i - 1$	q_4 $j := j - 1$

Table 3: A toy counter program.

(d) (3 marks) Sometimes, ABA-type situations are innocuous. Consider the program in Table 3. Describe how the CAS at p_3 may succeed, despite n having changed since the read on line p_2.

(e) (10 marks) The end result of running the program in Table 3 is unaffected by whether ABA situations happen or not. Or to be precise, the program will satisfy the following Hoare triple:

$$\{n = 0\}\ P||Q\ \{n = 2\}$$

Where P and Q are the left and right processes, respectively. Formulate a single global invariant that suffices to establish this Hoare triple.

Question 7 (10 marks)

The last item in Section 3.1 states "*Tail* always points to a node in the linked list, because it never lags behind *Head*, so it can never point to a deleted node".

(a) (5 marks) This statement is false for the two-lock concurrent queue. Describe how a state can be reached where *Tail* points to a deleted node.

Hint: look for a state where one of the locks is held.

(b) (5 marks) Is it possible to reach a state where *Head* points to a deleted node? Explain why or why not.

Dont forget to submit your work and include the statement given on the front page.

— END OF EXAM —