The University of New South Wales

Final Exam

2009/11/03

COMP3151/COMP9151

Foundations of Concurrency

Time allowed: **2 hours (9:45–12:00)**
Total number of questions: **5**
Total number of marks: **45**

Textbooks, lecture notes, etc. are not permitted, except for 2 double-sided A4 sheets of hand-written notes.

Calculators may not be used. (Not that they would be of any help.)

Not all questions are worth equal marks.

Answer all questions.

Answers must be written in ink.

You can answer the questions in any order.

You may take this question paper out of the exam.

Write your answers into the answer booklet provided. Use a pencil or the back of the booklet for rough work. Your rough work will not be marked.

Shared-Variable Concurrency (15 Marks)

Question 1 (8 marks)

Let $k > 1$. Let A be an algorithm which was designed to solve the mutual exclusion problem for 2 processes. The algorithm B for $n = 2^k$ processes is built up inductively by splitting the n processes into two groups of $\frac{n}{2}$ processes each. In each group the processes compete to enter the critical section using recursively the solution for $\frac{n}{2}$ processes. The winners of each of the two groups use A to determine which one is allowed to enter the top-level critical section.

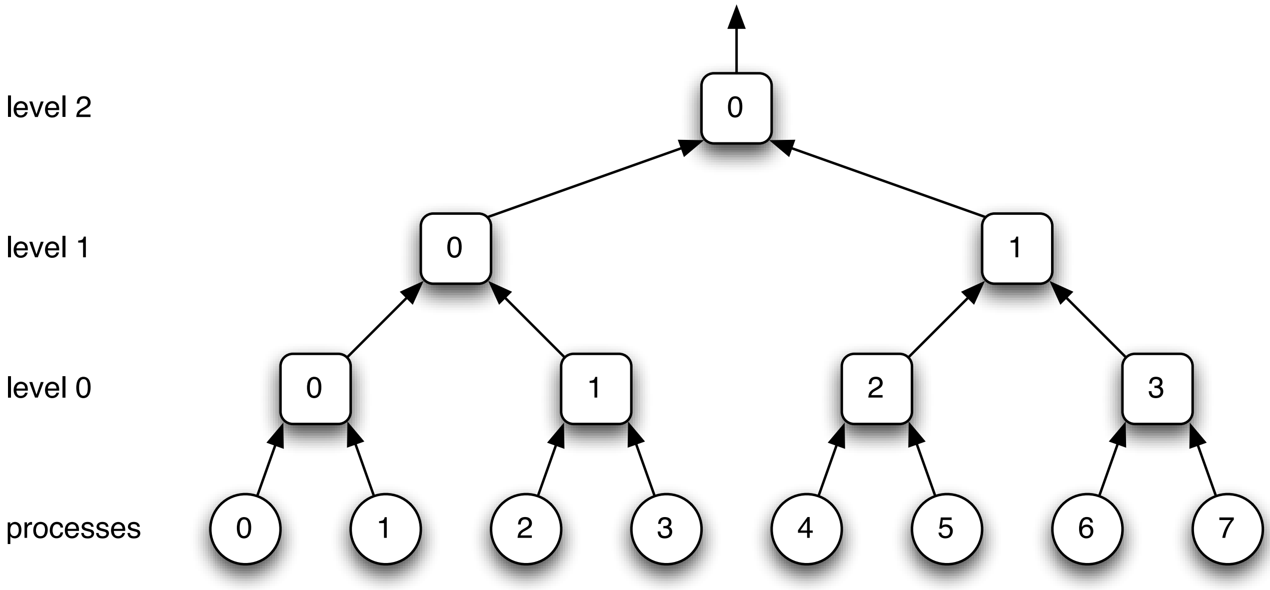

Figure 1: The tournament tree for 8 processes. At each level the nodes are numbered from left to right starting from 0. Thus, each node in the tree is uniquely determined by its level and node number.

Another way to view this idea is to consider the competition between the processes as a knock-out tournament, as illustrated in Figure 1. Processes start as leaves in a balanced binary tree. To enter its critical section, each process attempts to progress to the root of the tree, where at each level of the tree it participates in an instance of A with at most one process in its neighbour's sub-tree. The winner at the top level is allowed to enter its critical section. Upon exiting its critical section, the winner traverses the reverse path (from the root to the leaf) executing its post protocol of A at each level. Each instance of A will use a separate set of shared and local variables to avoid interference.

Prove or disprove:

(a) If A satisfies mutual exclusion then so does B.

(b) If A satisfies eventual entry then so does B.

Question 2 (7 marks)

Recall the fast mutual exclusion algorithm:

Algorithm: Fast algorithm for two processes	
integer gate1 $\leftarrow$ 0, gate2 $\leftarrow$ 0 boolean wantp $\leftarrow$ false, wantq $\leftarrow$ false	
p	**q**
p1: gate1 $\leftarrow$ p wantp $\leftarrow$ true	q1: gate1 $\leftarrow$ q wantq $\leftarrow$ true
p2: if gate2 $\neq$ 0 wantp $\leftarrow$ false goto p1	q2: if gate2 $\neq$ 0 wantq $\leftarrow$ false goto q1
p3: gate2 $\leftarrow$ p	q3: gate2 $\leftarrow$ q
p4: if gate1 $\neq$ p wantp $\leftarrow$ false await wantq = false	q4: if gate1 $\neq$ q wantq $\leftarrow$ false await wantp = false
p5: if gate2 $\neq$ p goto p1 else wantp $\leftarrow$ true critical section	q5: if gate2 $\neq$ q goto q1 else wantq $\leftarrow$ true critical section
p6: gate2 $\leftarrow$ 0	q6: gate2 $\leftarrow$ 0
p7: wantp $\leftarrow$ false	q7: wantq $\leftarrow$ false

Does it matter if we change the order of the last two statements in p (lines p6 and p7)?

Message-Passing Concurrency (30 Marks)

Answers to questions that require programming can be formulated using Ben-Ari's pseudo-code notation, Promela, or (if you must) C with MPI.

Question 3 (8 marks)

Develop an implementation of a time-server process. The server provides two operations that can be called by client processes: one to get the time of the day and one to delay for a specified interval. In addition, the time server receives periodic "tick" messages from a clock interrupt handler. Also show the client interface to the time server for the time of day and delay operations.

Question 4 (7 marks)

Develop a solution for the dining philosophers problem under the restriction that a channel must be connected to exactly one sender and one receiver.

Question 5 (15 marks)

Recall the dining cryptographers' problem. As before we assume that each cryptographer C_i has a secret bit $p_i \in \{0, 1\}$ that is 1 iff C_i paid. At most one of the cryptographers paid, but maybe none of them paid but the NSA did.

Consider the following algorithm for a party of n of them. Cryptographer C_1 secretly flips a coin to generate a result $c \in \{0, 1\}$. She then XORs c with p_1 and passes the result $c \oplus p_1$ on to her neighbour C_2. Every cryptographer C_i where $i > 1$ waits for input from C_{i-1}, XORs that input with p_i, and passes it on to his neighbour $C_{(i \bmod n)+1}$. When C_1 receives the bit b from C_n, she announces "the NSA paid" if $b = c$ and "one of us paid" otherwise.

3 marks Model the algorithm with transition diagrams.

2 marks Formulate a pre- and a postcondition to capture the crucial aspect of the algorithm, namely that the final announcement by C_1 is truthful.

5 marks Prove validity of the resulting Hoare-triple.

5 marks Consider the case $n = 3$. Prove or disprove that C_1 does not learn which one of C_2 and C_3 paid if one of them did.