Informatics and Mathematical Modelling
Technical University of Denmark
Building 321
DK–2800 Lyngby
Denmark

HHL 28–11–2008

02152 CONCURRENT SYSTEMS FALL 2008

CP Exercise Class 8

Monday December 1

Exam Problems

1. Do Exam December 2003, Problem 1 (see reverse)

2. Do Exam December 2003, Problem 2 (see other sheet)

3. Do Exam December 2003, Problem 3 (see other sheet)

Notice that all of these problems are from a 2-hours exam so the percentages states should be halved for a 4-hours exam. Also problems for a 4-hours exam may be more complex (some 4-hours exams will appear on the course page soon).

Good luck at the exam!

From Concurrent Systems Exam, December 2003 (2-hours)

PROBLEM 1 (approx. 20 %)

Three processes P_1, P_2, and P_3 execute three operations A, B, and C respectively. The operations are synchronized by means of semaphores:

var SA, SB, SC : *semaphore*;

$SA := 1;\ \ SB := 1;\ \ SC := 0;$

process P_1;	**process** P_2;	**process** P_3;
repeat	**repeat**	**repeat**
`P`(SA);	`P`(SB);	`P`(SC);
A;	B;	C;
`V`(SC)	`V`(SC)	`P`(SC);
forever;	**forever**;	`V`(SA);
		`V`(SB)
		forever;

Question 1.1:

(a) Draw a Petri Net in which the three operations A, B, and C are synchronized in the same way as in the above program. In the net, the operations should be represented by transitions.

(b) Determine which pairs of operations can be executed concurrently.

(c) State with a brief argument whether or not all three operations can be executed concurrently.

From Concurrent Systems Exam, December 2003 (2-hours)

PROBLEM 2 (approx. 30 %)

The questions in this problem can be solved independently of each other.

Question 2.1:

A process P uses three shared integer variables x, y, and z. The variable x is both read and written by other processes, whereas y and z are only read by other processes. Determine which of the following statements in P can be considered to be atomic.

a: $x := x + 1$	d: $y := y + 1$
b: $x := y + 1$	e: $x := y + z$
c: $y := x + 1$	f: $z := y + z$

Question 2.2:

A concurrent program is given by:

$$\textbf{var}\ x, y\ :\ \mathit{integer}\ :=\ 0;$$
$$\textbf{co}\ x\ :=\ y + 1\ \|\ \langle\, y\ :=\ x + 2 \,\rangle;\ x\ :=\ 2\ \textbf{oc}$$

(a) Draw a transition diagram for each process.

(b) Determine all possible final states (x, y) of the program.

Question 2.3:

Let x and y be integer variables. Determine which of the predicates P, Q, and R are preserved by which of the actions a_1, a_2, and a_3, respectively:

$P \triangleq x + y \geq 0$	a_1: $y := 0$
$Q \triangleq 0 \leq y \leq x$	a_2: $\langle\, y < 0 \rightarrow y := x + 1 \,\rangle$
$R \triangleq x \neq y$	a_3: $\langle\, y = 0 \rightarrow x := 0 \,\rangle$

Question 2.4:

Let x and y be integer variables and let the temporal logic formula F be defined by:

$$F \triangleq (\Box\, y > x \geq 0) \wedge (\Box\Diamond\, x = 0) \wedge (x = 0 \leadsto x \neq 0)$$

(a) Let states be given by pairs (x, y). Give an example of an execution for which F holds. The execution should be given as a short sequence of states which is repeated forever.

Now, consider each of the following actions within a program:

a_1: $\langle\, \textbf{await}\ x = 0 \,\rangle$	a_3: $\langle\, \textbf{await}\ x = 0 \vee y > 1 \,\rangle$
a_2: $\langle\, \textbf{await}\ y > 1 \,\rangle$	a_4: $\langle\, \textbf{await}\ x = 0 \wedge y > 1 \,\rangle$

Assume that control has reached the particular action and that F is valid for the program.

(b) Determine which of the actions will be eventually executed assuming weak fairness.

(c) Determine which of the actions will be eventually executed assuming strong fairness.

From Concurrent Systems Exam, December 2003 (2-hours)

PROBLEM 3 (approx. 20 %)

Let N be a positive integer. The server-based module *Batch* given below implements a synchronization mechanism that "collects" a batch of N items provided by calls of *put*() which may then be "removed" by a call of *unload*().

```
module Batch
  op put();
  op unload();
body

  process Control;
    var count : integer := 0;
    repeat
      while count < N do
            in put() → count := count + 1 ni;
      in unload() → count := 0 ni;
    forever;

end Batch;
```

Question 3.1:

Assume $N = 3$. Suppose that, concurrently, *unload*() is called by two processes and *put*() is called by five processes. Assuming no further calls, describe the overall effect of these seven calls.

Question 3.2:

Now, the module *Batch* is to be replaced with a monitor which provides the same operations and behaves in the same way. Write such a monitor.