The University of New South Wales

Final Exam

2007/11/19

COMP3151/COMP9151

Foundations of Concurrency

Time allowed: **3 hours (8:45–12:00)**
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 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.

Be concise — *excessively verbose answers will be penalised.* Use a pencil or the back of the booklet for rough work. Your rough work will not be marked.

Shared-Variable Concurrency (15 Marks)

(8 marks)

Give all possible final values of variable x in the following program. Prove your answer correct in Andrews' *PL*. (See the appendix for the logic.)

```
1    int x = 1, i = 0;
2    co while (i < 3) {
3          ⟨ x = x * 2; ⟩
4          ⟨ i = i + 1; ⟩
5        }
6    // ⟨ x = x − 1; ⟩
7    oc
```

(7 marks)

Is the following barrier algorithm for n processes correct? Justify your answer.

```
1   sem arrival = 1, out = 1, departure = 0;
2   int counter = 0;
3
4   procedure B () {
5     P (arrival);
6     counter++;
7     if (counter == n) {
8       P (out);
9       V (departure);
10    }
11    V (arrival);
12    P (departure);
13    V (departure);
14    P (arrival);
15    counter−−;
16    if (counter == 0) {
17      P (departure);
18      V (out);
19    }
20    V (arrival);
21    P (out);
22    V (out);
23  }
24
25  process S[i = 1 to n] {
26    for[j = 0 to k] { # k is defined elsewhere
27      task(i, j);     # tasks are defined elsewhere
28      B ();           # barrier
29    }
30  }
```

Message-Passing Concurrency (30 Marks)

You are not allowed to use shared mutable state in answers to the programming questions.

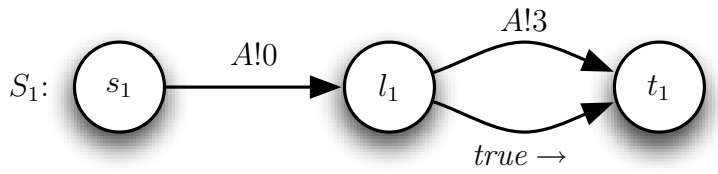
(8 marks)

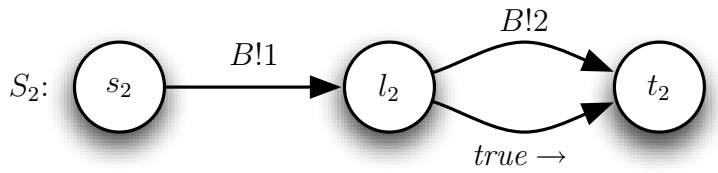
The Bear and the Honeybees. Given are n honeybees and a hungry bear. They share a pot of honey. The pot is initially empty; its capacity is H portions of honey. The bear sleeps until the pot is full, then eats all the honey and goes back to sleep. Each bee repeatedly gathers one portion of honey and puts it in the pot; the bee who fills the pot awakens the bear.

Write a simulation with one process for the bear and n processes for the honeybees.

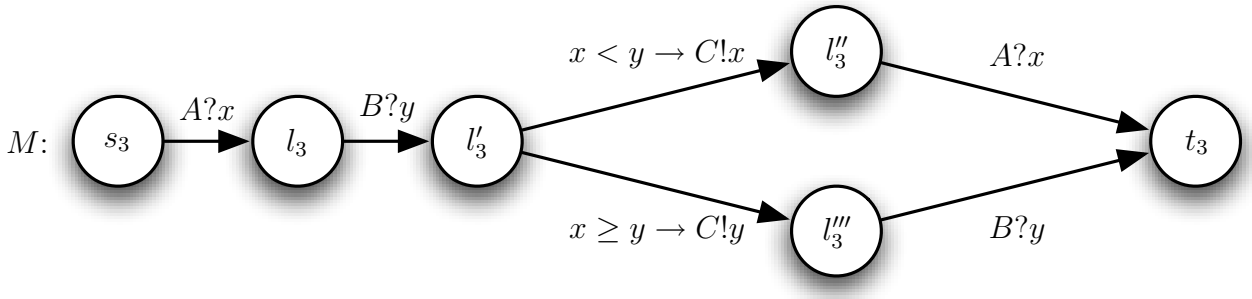
(12 marks)

Consider the following processes. First we have two producers S_1 and S_2.

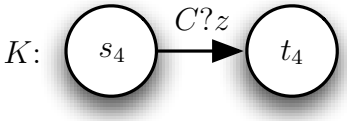




Process M merges the output of S_1 and S_2 and sends it along channel C.



Finally we have a consumer K.



Prove $\{true\}\, S_1 \| S_2 \| M \| K\, \{x = 3 \wedge y = 1 \wedge z = 0\}$ using either the proof method of Levin & Gries or AFR for 10 marks or the closed product for 6 marks. Disprove termination for 2 marks.

(10 marks)

The H_2O *problem.*

Two kinds of atoms, H's and O's, enter a reaction chamber. An O atom cannot leave until it meets two H atoms and an H atom cannot leave until it meets an O atom. Each atom leaves the chamber—without meeting any other processes—once it has met the required number of other processes.

Develop a server process to implement this synchronisation. Show the interface of the processes modelling individual H and O atoms. It is recommended to use the multiple primitives notation (**in** ... **ni**).

For 5 bonus marks generalise your solution to be parametric in the molecule to be built such that it works, e.g., for propanol (C_3H_8O or $CH_3CH_2CH_2OH$).

Andrews' PL (a Proof System for MPD Annotations)

Assignment axiom

$$\{\phi[^{e}/_{x}]\}\, x \texttt{ = } e\, \{\phi\} \qquad \textbf{ass}$$

Composition rule

$$\frac{\{\phi\}\, S_1\, \{\psi\}\,,\ \{\psi\}\, S_2\, \{\psi'\}}{\{\phi\}\, S_1 \texttt{;} S_2\, \{\psi'\}} \qquad \textbf{comp}$$

If-Else statement rule

$$\frac{\{\phi \wedge b\}\, S_1\, \{\psi\}\,,\ \{\phi \wedge \neg b\}\, S_2\, \{\psi\}}{\{\phi\}\, \texttt{if (}b\texttt{) } S_1 \texttt{ else } S_2\, \{\psi\}} \qquad \textbf{if}$$

While statement rule

$$\frac{\{\phi \wedge b\}\, S\, \{\phi\}}{\{\phi\}\, \texttt{while (}b\texttt{) } S\, \{\phi \wedge \neg b\}} \qquad \textbf{while}$$

Rule of consequence

$$\frac{\phi' \rightarrow \phi,\ \{\phi\}\, S\, \{\psi\}\,,\ \psi \rightarrow \psi'}{\{\phi'\}\, S\, \{\psi'\}} \qquad \textbf{cons}$$

Await statement rule

$$\frac{\{\phi \wedge b\}\, S\, \{\psi\}}{\{\phi\}\, \texttt{<await (}b\texttt{)}S\texttt{>}\, \{\psi\}} \qquad \textbf{await}$$

Co statement rule

$$\frac{\{\phi_i\}\, S_i\, \{\psi_i\} \text{ hold and are interference free}}{\{\bigwedge_i \phi_i\}\, \texttt{co } S_1 \texttt{ //} \ldots \texttt{// } S_n \texttt{ oc}\, \{\bigwedge_i \psi_i\}} \qquad \textbf{co}$$

Semaphore wait rule

$$\frac{\phi \wedge s > 0 \rightarrow \psi[^{s-1}/_{s}]}{\{\phi\}\, \texttt{P(}s\texttt{)}\, \{\psi\}} \qquad \textbf{P}$$

Semaphore signal rule

$$\frac{\phi \rightarrow \psi[^{s+1}/_{s}]}{\{\phi\}\, \texttt{V(}s\texttt{)}\, \{\psi\}} \qquad \textbf{V}$$

Simplifying assumption: arithmetic on bounded types such as `int` does not wrap around silently. Overflow and underflow errors lead to abnormal termination which renders program behaviours irrelevant to partial correctness arguments such as proofs in *PL*.