Informatics and Mathematical Modelling *HHL 2008–10–24*
Technical University of Denmark
Building 321
DK–2800 Lyngby
Denmark

02152 CONCURRENT SYSTEMS FALL 2008

Exercise Class 6

Monday October 27

Deadlocks

1. Solve CP Exam December 1998, Problem 4 (In [Aux] page 9).

Synchronous Message Passing (CSP)

2. Write a CSP process *Merge* that interleaves two streams of integers coming from processes A and B respectively and passing the interleaved stream on to a process C.

3. Write a CSP process *Sum* that repeatedly receives an integer value from each of two processes A and B, and passes their sum to a process C.

4. Solve CP Exam June 1994, Problem 3. (In [Aux] page 8.)

5. Show how to implement a barrier for three processes using CSP communication (with void messages).

Asynchronous Message Passing

6. Do Andrews Ex. 7.7. The solution should be similar to Andrews Figure 7.7. You may postulate any abstract data type you need to keep track of the time.

To conveniently express that various kinds of messages can be sent to the server, you may use (SML-like) data types like:

type $Id = 1..n$;
type $Time = integer$;
type $Op =$ GetClock(Id) | Delay($Id, integer$) | Tick;

which can be explored using a case construction like:

case *op* :
  GetClock(id): ...
  Delay(id, d): ...
  Tick: ...
end case

7. *OPTIONAL* Do Andrews Ex. 7.16. You should generalize the problem to a number of processes $P_1, P_2, \ldots, P_n$. You need not worry to much about data structures and syntax, just try to sketch your idea. Hint: Use a ring communcation structure and let one of the processes send out a *probe* message.

Note: This problem is the essence of the *distributed meeting planning* problem — to find the first common free time-slot in the individual calendars of a group of participants.