Lecture 18
Concurrent Programming

2nd November 2003

Discussion of exam handback

Redo the exam as a takehome.

Linda Barriers revisited

Last time I asked you to think about how to make the Linda barrier reusable. Here are two approaches of varying success.

A attempt to use a coordinator process:

```
Initialize the barrier and execute the coordinator:
OUT("barrier", 0)

Coordinator will execute:
OUT("coord", 0)
IN("coord", n)
IN("barrier", n) ---> we need this
OUT("barrier", 0)

Each time a process arrives at the barrier it executes;
IN("barrier", ?counter)
OUT("barrier", counter+1)
RD("barrier", n)
IN("coord", ?counter)
OUT("coord", counter+1)
```

This “solution” doesn’t work. Why? After a process executes its RD(“barrier”, n) operation it can cycle around and add one to the barrier, breaking other processes that still need to read the value n in their own RD operations.

The problem here is that the reusable barrier *must* have code for waiting twice: once on arrival at the barrier and once before reusing the the barrier. The code in the processes here does not have any place that it would wait before reusing the barrier: there is essentially always a “coord” tuple present that can be read and updated.

An attempt using two barriers:

```
Init:
OUT ("Barrier", 0)

1 Process (Arbitrary):
IN("Barrier", ?ct)
OUT("Barrier", ct+1)
IN("Barrier", N)
OUT("Barrier2", N-1)
IN("Barrier2", 0) // Waits for count down on second barrier
OUT("Barrier", 0) // Restarts the first Barrier

Other Processes:
IN("Barrier", ?ct)
OUT("Barrier", ct+1)
IN("Barrier2", ct)
OUT("Barrier2", ct-1)
```

This attempt also does not work. The problem here is that if in one of the “other” processes the IN operation of “Barrier” happens to read 0, that process and process 1 will race for the Barrier2 tuple with value 0. One of them will win and the other will never make more progress. This can be turned into a solution however with a small change to the code:

```
Init:
OUT ("Barrier", 1)

1 Process (Arbitrary):
IN("Barrier", N) //Waits for all of the processes to reach the barrier
OUT("Barrier2", N-1) // Starts the count down (which releases all of the
processes
IN("Barrier2", 0) // Waits for count down on second barrier
OUT("Barrier", 1) // Restarts the first Barrier

Other Processes:
IN("Barrier", ?ct)
OUT("Barrier", ct+1)
IN("Barrier2", ct)
OUT("Barrier2", ct-1)
```