Technische Universität München
Institut für Informatik
PD Dr. Ralf-Peter Mundani
Dipl.-Inf. Christoph Riesinger

SoSe 2015

High Performance Computing - Programming Paradigms and Scalability

Exercise Sheet 3: Dependency Analysis, Synchronisation, Parallel Structures

1 Dependency Analysis

For the successful parallel execution of processes, a dependency analysis is necessary. Here, *Bernstein's* conditions, for instance, help to identify instruction-level parallelism as to be observed in loops. It is required that an iteration of a loop is independent from all other iterations, i.e. read/write dependencies do not exist.

Examine the following code fragment and decide whether instruction-level parallelism might occur or not.

```
1 for (i = 1; i <= n; ++i) {
2     a[i] = 2 * a[i+m] + b[i];
3 }
```

For which values of $-1 \leq m \leq 2$ is a parallel execution of the loop iterations possible?

2 Semaphores

The Hamburg airport has two runways that cross in some point E (see figure **??**). Arriving airplanes either choose runway AB or runway CD for landing.

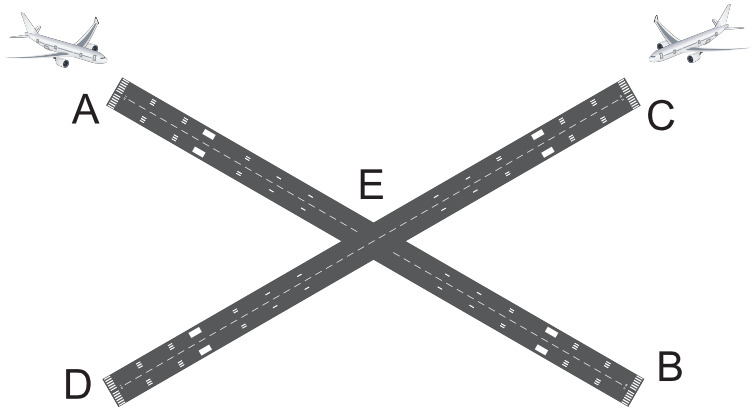

Figure 1: Crossing airplanes runways.

In order to prevent two planes from crashing, synchronisation is inevitable. Therefore, the airport administration designs the following rules:

a) No two airplanes can use the same runway at the same time.

b) If an airplane passed point E on its runway, the other runway is open for landing.

You are allowed to use as many binary semaphores as you like, in any point A, B, C, D, E as you like. Give a valid solution for both AB airplanes and CD airplanes with the correct usage of P(σ) and V(σ) operations for your semaphores σ! Discuss if your solution is a fair solution!

3 Deadlocks

In a multicore system 4 processes (threads) P_i are competing for 2 resources R_1 and R_2. Both resources are available two times. In order to finish their computations, the processes have the following request (in that order).

$$P_1 \to R_1, P_3 \to R_2, P_4 \to R_2, P_2 \to R_1, P_1 \to R_2, P_3 \to R_1, P_2 \to R_2$$

Assuming that all resources are only exclusively useable, resources cannot be withdrawn from a process, and processes do not release assigned resources while waiting for the allocation of other resources, the above situation might lead to a deadlock.

Draw an allocation/request graph for this scenario and discuss if the above situation is deadlock-free and in what ordering of execution – if possible – processes P_1 to P_4 might finish their computations!