Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Leonel Merino

Series 02 — 26.09.2017 – v1.0
Concurrency and Java

Exercise 1 (2 Points)

Answer the following questions (0.5 point each):

a. What states can a Java thread be in?

b. How can you turn a Java class into a monitor?

c. What is the *Runnable* interface good for?

d. Specify an FSP that repeatedly performs hello, but may stop at any time.

Exercise 2 (2 Points)

Consider the following Java implementation of a Singleton:

```
Public class Singleton {
   private static Singleton instance = null;
   private Singleton() {}
   public static Singleton getInstance() {
      if(instance == null) {
         instance = new Singleton();
      }
      return instance;
   }
}
```

This implementation assumes a single-threaded application.

a. What happens if the application is multithreaded?

b. How to implement a thread-safe singleton in Java?

c. Suppose there is 1000 requests/second from different threads to this Singleton. Does your implementation introduce a bottleneck? If yes, how can you improve it?

Exercise 3 (3.5 Points)

Download LTSA from http://www.doc.ic.ac.uk/~jnm/book/ltsa/download.html. For each of the following processes shown in Figure 1, give the Finite State Process (FSP) description of the corresponding Labeled Transition System (LTS) graph. You may verify the FSP descriptions by generating the corresponding state machines using the analysis tool.

Exercise 4 (2.5 Points)

Consider the full Race5K FSP from the lecture.

a. How many states and how many possible traces does it have if the number of steps is 5 (as in the lecture)?

b. What is the number of states and traces in the general case (i.e. for n steps)?

c. Check your solution using the LTSA tool.

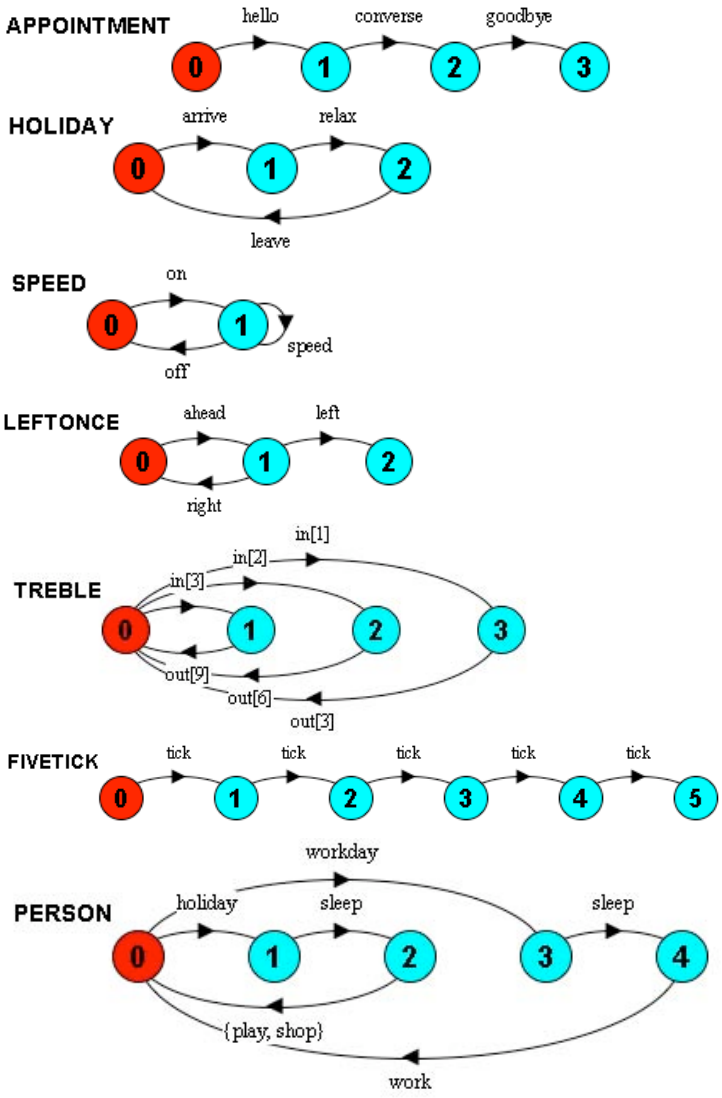

Figure 1: LTSA graphs.