Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Leonel Merino

Series 05 — 18.10.2017 – v1.0b
Liveness and Guarded Methods

Exercise 1 (4 Points)

Answer the following questions:

a. What is a guarded method and when should it be preferred over balking?

b. Why must you re-establish the class invariant before calling wait()?

c. What is, in your opinion, the best strategy to deal with an InterruptedException? Justify your answer!

d. How can you detect deadlock? How can you avoid it?

e. Why it is generally a good idea to avoid deadlocks from the beginning instead of relying on deadlock detection techniques?

f. Why is progress a liveness rather than a safety issue?

g. Why should you usually prefer notifyAll() to notify()?

h. What is the difference and what are the similarities between a nested monitor lockout (a.k.a. nested deadlock) and a classical deadlock?

Exercise 2 (2 Points)

You have seen the dining philosophers problem during lecture. Please describe *four different solutions* to this deadlock problem and explain the *concept* and *fairness characteristics* of each.

Exercise 3 (2 Points)

Consider the FSP model for the dining philosphers:

```
PHIL = ( sitdown
-> right.get -> left.get -> eat
-> left.put -> right.put -> arise -> PHIL ).

FORK = ( get -> put -> FORK).

||DINERS(N=5) =
forall [i:0..N-1]( phil[i]:PHIL
|| {phil[i].left,phil[((i-1)+N)%N].right}::FORK ).
```

Modify the FSP specification, in a different manner than seen in the lecture, to remove the deadlock. Fairness is not mandatory.
Hint: You can search for deadlocks with the LTSA tool through the menubar at "Check" - "Safety" or "Check" - "Progress".

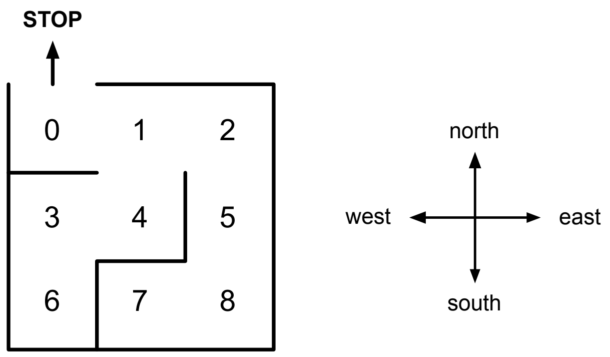

Figure 1: A maze.

Exercise 4 (2 Points)

In a FSP model of a process a deadlocked state is a state with no outgoing transitions, a terminal state so to say. A process in such a state can engage in no further actions. In FSP, this deadlocked state is represented by the local process STOP. By performing a breadth-first search of the LTS graph (in CHECK PROGRESS), the LTSA tool guarantees that a sample trace is the shortest trace to the deadlock state.

Consider the maze depicted in Figure 1. Write a description of the maze as FSP process which, using deadlock analysis, finds the shortest path out of the maze starting at any square in the maze.

You may check your solution by inspecting the *trace to deadlock* from specific squares.

Hint: At each numbered square in the maze, a directional action can be used to indicate an allowed path to another square.