Prof. Dr. Oscar Nierstrasz
Pascal Gadient, Mohammadreza Hazhirpasand

Assignment 05 — 20.10.2021 – v1.0
Liveness and Guarded Methods

IMPORTANT:
The next time (27 October 2021) we will perform the hands-on lab from 10:15am to 12:00pm at a different location, because we need more space to assist you. It is mandatory to attend the lab in person! Please be there on time and bring your own charged notebook (don't forget the power adapter) that can connect to the university network to access the internet!

Address where the lab will take place:
Universität Bern
Schanzeneckstrasse 1
3012 Bern
Room S003

Exercise 1 (4 Points)

Answer the following questions:

a) When should a guarded method be preferred over balking?

b) What is in your opinion the best strategy to deal with an InterruptedException? Justify your answer!

c) How can you detect a deadlock?

d) How can you avoid a deadlock?

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 are the differences 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.

NB: You can search for deadlocks with the LTSA tool through the menubar at "Check" - "Safety" or "Check" - "Progress".

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.

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

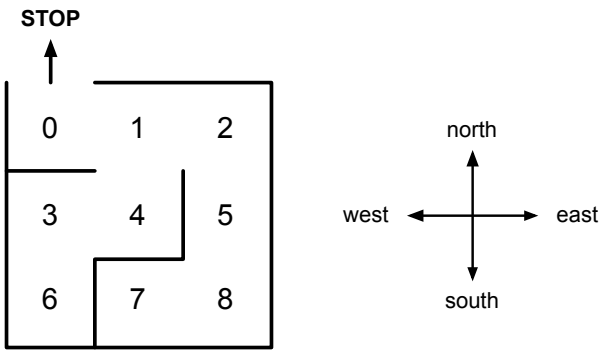

Figure 1: A maze.