Formal Methods in Software Development
Exercise 9 (March 17, 2008)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

January 29, 2008

The result is to be submitted by the deadline stated above via the Moodle interface as a .zip or .tgz file with

- a PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - the formal model of the system and the formulation of each property in LTL,
  - the PROMELA model of the system and the formulation of each property in Spin's version of PLTL (including the definitions of the atomic predicates),
  - for each property, the output of the Spin model checker,
  - for each violated property, (a significant part of) a screenshot of the sequence chart for a counterexample run and an ample explanation of the relevance of this counterexample;
  - an answer to the question of the correctness of the termination detection algorithm.
- files (.promela and .pltl) with the PROMELA model of the system and the Spin formulations of the properties that were model checked.

After submission, send me a notification per email.

9: Modeling and Verifying a Concurrent System

Create a formal model (state space, initial state condition, transition relation) of the following concurrent system (where shared variables are assumed for modeling communication among the components):

1. We have N components called *nodes* organized in a ring, i.e., every node has two neighbors with which it can communicate.

2. Every node is in one of the states *active* or *idle*; an active node can send signals to its left respectively right neighbor; however, every node may generate not more than S such signals.

3. An active node can by itself at any time become *idle*; however, an *idle* node becomes active only by a signal from one of its neighbors.

4. Every node has the task to find out when the system has *terminated*, i.e., when all nodes (including itself) are idle (since no idle node can generate a signal, no node can then be awaked again); if it detects this, it sets a local variable t to "true".

For solving the task, every idle node continually starts termination detection by sending a "probe" (a special signal) in one direction of the ring. This signal carries the identity (number) of the node that started the probe. Every idle node forwards the probe to the next neighbor (but remains idle); an active node drops the probe. If the probe eventually returns to its originator, and the originator is still idle, it sets t to "true".

The overall idea is that, if the system is terminated, all local variables t should eventually become "true". More specifically, we are interested in the following properties (where the meaning of "termination" is as explained above):

1. If some local t is "true", then the system is indeed terminated.

2. If the system is terminated, then eventually all local t become "true".

3. The system eventually terminates.

Formulate these questions in LTL.

Next, create a PROMELA model of the system (using channels and message passing rather than shared variables for communication among the components). Formulate and verify/falsify above properties with the help of Spin for $N = 4$ nodes and $S = 3$ signals per node (or smaller values, if necessary). For verifying lifeness properties, introduce (if necessary) sufficiently strong fairness conditions in order to avoid "trivial" counterexamples. If you find a significant counterexample, give an ample explanation and interpretation of its relevance.

Based on your findings, is above algorithm for termination detection correct? If not, can you give additional assumptions or modifications under which it becomes correct?