Formal Methods in Software Engineering
Exercise 10 (February 9)

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

January 12, 2009

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

- A PDF file with
  - a cover page with the title of the course, your name, Matrikelnummer, and email-address,
  - the Promela model,
  - a screenshot of (part of) a simulation run,
  - for each property to be checked, the PLTL formula, the definition of the atomic predicates, a screenshot of the verification window with the message "valid/not valid", and (if not valid) a screenshot of (the end of) a counterexample run (which should be as simple as possible).
- the sources of the Promela model and of the properties verified (as saved from the verification window in a text file).

Model-Checking with Spin

Take a system with N processes P_i ($i = 0 \ldots N-1$) that are organized in a ring such that messages can flow in both directions of the ring i.e. each process P_i is connected by two buffered channels with its neighbor processes P_{i+1} and P_{i-1} (where $+/-$ denotes arithmetic modulo N).

Each process is one of two states "active" or "idle" such that:

- an active process may spontaneously generate messages and send them in any direction, but it also may become idle,
- an idle process does not generate any message and remains idle until it is awaked by receiving a message.

If all processes are idle and there are no messages pending in the channels, we call the system "terminated"; a terminated system remains in this state.

To detect termination, P_0 may in idle state start the following activity:

- P_0 sends in both directions of the ring a special "termination detection" token that carries initially a value "true";
- if an idle process receives this token, it forwards it without change (and remains idle);
- if an active process receives this token, it sets the token value to "false" before forwarding it.

P_0 waits until it receives both tokens (during this time, it forwards other messages but does not start a new termination detection). P_0 then checks the values of both tokens; if they are both still "true" and P_0 is still idle, P_0 concludes that the system is terminated.

Implement a Promela model of this system for $N = 4$ and channels with buffer size 2 (to avoid deadlocks, a message is only generated and submitted, if the channel is empty, while a termination detection token is unconditionally forwarded).

- Simulate the Promela model to validate (convince yourself about) its adequacy with respect to above specification.
- Formulate the property "if the system is terminated, it remains so forever" and verify/falsify it with Spin.
- Formulate the property "the system will eventually terminate" and verify/falsify it with Spin.
- Formulate the property "if process 0 concludes termination, the system is indeed terminated" and verify/falsify it with Spin.
- Formulate the property "if the system is terminated and process 0 starts termination detection, it will eventually conclude termination" and verify/falsify it with Spin.

In case of liveness properties, you may introduce additional fairness constraints in your specification, if they are required to make the property true.

For each verification/falsification explicitly state the result and give a verbal interpretation (do you think it is okay or not? why?). So do you think above termination detection algorithm is correct or not?