Formal Methods in Software Engineering
Exercise 9 (January 26)

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

January 8, 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 the) counterexample run.
- 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 has a buffer in which it can hold a single message and a counter with a maximum value M.

Each process waits for messages from both directions. If such a message arrives, there are three options:

- If the counter is not zero, the message is immediately forwarded in the other direction and the counter is decreased.

- If the counter is zero and there is not yet a message in the buffer, the message is stored in the buffer.
- If the counter is zero and there is already a message in the buffer, both messages "bounce" (i.e. they are returned to their original senders) and the counter is reset to M.

Define a Promela model of above system for $N = 3, M = 1$ and channels with buffer size 1 such that initially (in the `init` clause) two messages are put into the two input channels of process 0 and the processes are started. Then perform the following tasks:

- Simulate the Promela model to validate (convince yourself about) its adequacy with respect to above specification.
- Formulate the property "at most two processes hold messages in their buffers" and verify it with Spin.
- Formulate the property "it is never the case that process 1 holds a message in its buffer" and demonstrate by a counterexample run that this property is violated (make the counterexample as short as possible).
- Demonstrate by a run (derived from a failed model check) that the system may run into a deadlock (i.e. that two processes hold messages in their buffers such that no further action can occur); again make this run as short as possible.

Bonus (20%) Change the system model such that a process with a message in its buffer may spontaneously (while waiting for another message) remove the message from the buffer, reset the counter to M, and forward the message appropriately.

Formulate and verify/falsify for this system the property: if process 0 has a message in its buffer, it will eventually have no message in its buffer.