Formal Models for Parallel and Distributed Systems Exercise 2 (June 1)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

May 5, 2011

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

1. a PDF file with a decent cover page (mentioning the title of the course, your full name and Matrikelnummer) with
   - listings of the model/configuration files and
   - the output of the TLC model checker runs,
2. all .tla and .cfg files used in the specification and model checking.

A Client/Server System: CCS/FSP

First develop a CCS specification (in the value passing calculus presented in the lecture) of the system of Exercise 1 with one server and N processes and $2N$ buffered channels of size B. You may use in your specification variables ranging over integers and can use the usual integer operations in conditional expressions and output actions.

Hint: since in the system no information is carried by messages, it suffices to model the buffer with a state variable i that describes the number of messages in the buffer; see also the Example `BoundedBuffer` in LTSA.

Next translate this specification as directly as possible to a FSP model with $N = 3$ and $B = 1$.

Construct drawings for the labeled transition system of the server process, one client process, one channel process, and (if possible) of the composed system.

Construct manually in the animator a trace of a (part of a) system run where Client 1 requests the resource, receives the resource, and releases the resource.

Check whether the system may run into a deadlock and give the output of the check.

Check whether the system maintains liveness for client 1 by defining a progress property that includes the client's actions for requesting the resource and entering the critical region, e.g.

```
  progress LIVENESS = { c[1].request, c[1].enter }
```

(see also example `Twocoin` in LTSA).

Hide from the model all action names except those for entering and exiting the critical region by the clients, perform minimization, and construct a drawing for the minimized system (see also example `User` in LTSA).

Explain whether/how the drawing illustrates that mutual exclusion is preserved.

Verify formally whether the system maintains mutual exclusion by defining a corresponding mutual exclusion property, e.g.

```
  progress MUTEX = (c[i:1..N].enter->p[i].exit->MUTEX).
```

which is composed with the system (see also Example `Mutex_property` in LTSA).