Assignment 2 of Automated Software Validation Course

Due on: June 12, 2007

Tool to be used: SPIN model checker
http://spinroot.com/spin/whatispin.html

Submit all of the following as a single tar file to abhik@csa.iisc.ernet.in

The assignment should done individually. No late submissions please.

- A README file explaining all contents of your submission. It should also contain all information required for me to reproduce any verification runs you have done using SPIN.
- Your report (pdf file) containing details of bugs found, how you analyzed the protocol, sample outputs from SPIN (e.g. Message Sequence Charts) summary of results obtained, and your experience in using SPIN for protocol validation (any suggestions/improvements you might want as an user etc).
- The modified Promela code. If you have several modified versions, please submit ALL of them, and explain what each version achieves in the README file.

Consider two stations connected in a ring. Each station can both send and receive data. Consequently, we need to assign two operators to each station. Let the stations be 1 and 2, and the operators be 1a,1b,2a,2b. Then operator 1a handles data communication from station 1 to station 2, while operator 1b handles data communication from station 2 to station 1. Similarly for operators 2a, 2b.

Note that depending on the protocol for data communication, we may not always perform communication from station 1 to station 2 via a single channel (from 1 to 2). For example, if the protocol involves transferring data (from 1 to 2) followed by acknowledgment (from 2 to 1), then we might want to have separate channels for the actual data items and the acknowledgment signal. The following Promela code implements such a protocol for two stations. Since each station has two operators, our Promela code has four processes running concurrently.

```
#define true 1
#define false 0

bool busy[2];

mtype = {start, stop, data, ack};

chan up[2] = [1] of { mtype };
chan down[2] = [1] of { mtype };

proctype station(byte id; chan in, out)
{ do
          :: in?start ->
            atomic { !busy[id] -> busy[id] = true };
            out!ack;
            do
                 :: in?data -> out!data
                 :: in?stop -> break
             od;
            out!stop;
            busy[id] = false
         :: atomic { !busy[id] -> busy[id] = true };
           out!start;
           in?ack;
          do
               :: out!data -> in?data
               :: out!stop -> break
          od;
          in?stop;
          busy[id] = false
   od
}

init{
   atomic {
        run station(0, up[1], down[1]);
        run station(1, up[0], down[0]);
        run station(0, down[0], up[0]);
        run station(1, down[1], up[1]);
    }
}
```

Try to formalize the property that no communication between stations is aborted in the above protocol, i.e. any communication that is started (with a `start` signal) is finished (with a `stop` signal). Use Linear time temporal logic (LTL) to express your property description. You need to be careful about your choice of atomic propositions.

The above property is not true for our protocol. Use the SPIN toolkit to find out why it is not true. You can use any of the features of SPIN: random simulation, user guided simulation, model checking. Feel free to augment the Promela code to introduce auxiliary variables etc. if you need it (of course your additions should not change the meaning of the protocol).

Your answer will be graded based on your understanding of the protocol. You should clearly explain how you gained this understanding from the output of your experiments with SPIN. So, any additional problems you find in the protocol will of course distinguish your answer and earn more credit.