CS4271 Homework 2

March 5, 2010

1 Notes

- This assignment is due before 11:59 PM, Monday, 22nd March, 2010. No late submissions!
- Tool to be used: SPIN model checker `http://spinroot.com/spin/whatispin.html`
- This is an individual assignment. Acts of plagiarism are subjected to disciplinary action by the university. Please refer to `http://www.comp.nus.edu.sg/students/plagiarism/` for details on plagiarism and its associated penalties.
- *Submission Instructions*: (Failure to follow these instructions may result in deduction of marks)
  1. Create a folder named your matriculation number YourMatricNumber, e.g. U123456M. Create the following files in this folder (name these files exactly as instructed.):
     - **assignment2**: Create two folders inside it:
       - **Question 1:** Include the modified spin source (if needed) and the LTL property checker file(s) needed for Question 1 (Part B).
       - **Question 2:** Include your implemented spin source and the LTL property checker file(s) needed for Question 2.
     - **report.pdf:**
       - For Question 1 (Part A), explain the LTL property used.
       - For Question 1 (Part B), describe the counter example you get from SPIN and explain how it can violate the property. Include the screenshot of the counter example (*e.g.* message sequence chart) in the report.
       - For Question 2, do the same things as in Question 1 (Part B).
     - **readme.txt:** It should contain all information required for me to reproduce any verification runs you have done using SPIN.
  2. Zip (using WinZip) the entire YourMatricNumber folder (including the folder itself and all files in it) into a file YourMatricNumber.zip.
  3. Submit YourMatricNumber.zip to the IVLE Workbin Folder **HW2-Submission**.

2 Question 1

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]);
    }
}
```

- **Part (A):** (2 marks) 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.

- **Part (B):** (3 marks) 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.

3 Question 2 (courtesy Gerard Holzmann) [5 marks]

One of the pioneers of the internet is Leonard Kleinrock. In an interview he described a famed deadlock that paralyzed an early version of the Arpanet in the 70s as follows:

"Here was a really interesting problem, what we referred to as reassembly deadlock. I published the first book about the Arpanet in 1976 (Queueing Systems: Computer Applications, John Wiley and Sons) and described reassembly deadlock there, along with many other deadlocks. Let's assume that a number of messages are in the process of arriving at some destination IMP, which is reassembling these messages. But there's only so much space for reassembly. When a packet for a new message comes in, it might be eight packets long (that was the max), and so eight packet buffers are put aside for reassembling that message. So you have a bunch of eight-buffer segments put aside, partially filled, none of them complete; let's refer to the missing packets as "pink" packets.

Let's further assume that all of the reassembly buffer space is currently in the process of reassembling these messages. Now if you look at all the switches in the IMPs attached to this IMP, they're also sending packets to this same destination IMP. Let's further assume that all of the storage in these IMPs is full of packets, but none of them are the missing pink packets. They're all from newer messages (say "green" packets). Where's the missing pink stuff? One layer behind that. Now these missing pink guys can't get to the destination until these green guys get through; and the green can't get through until the pink get through, complete the reassembly of some messages, and release the reassembly buffer space! Bang! That's your re-assembly deadlock. It forced BBN to change the whole operating system for the IMP. That was deadly."

Write a Promela model of a network node with three incoming streams of messages, and one outgoing stream. Use handshake for the incoming messages, and store them inside the node in a local buffer (an array). Each of the incoming data streams produces only messages of one specific type (say red, green, and blue for the three streams). Require that each message to be sent on the outgoing stream contains a structure of 3 fields, one field of type red, one of type green and of type blue. (i.e., you need one of each type of incoming message to assemble one outgoing message.) Show how the reassembly deadlock can happen. Define the key property in Linear-time temporal logic (LTL), use SPIN to prove that it can be violated, and show the counter-example.

END OF ASSIGNMENT 2