CS4211 Assignment using SPIN model checker

National University of Singapore, Year 2020

Given out by Abhik Roychoudhury

- This assignment is due before **9:59 PM, 23 October, 2020**. **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 student number YourStudentNumber, e.g. A123456M. Create the following files in this folder (name these files as instructed.):
     - **assignment**: Create three sub-folders inside it:
       i. **Question 1:** Include the modified spin source (if needed) and the LTL property checker file(s) needed for Question 1.
       ii. **Question 2:** Include your implemented spin source and the LTL property checker file(s) needed for Question 2.
       iii. **Question 3:** Include your implemented spin source and the LTL property checker file(s) needed for Question 3.
     - **report.pdf:** Please include your particulars (name, student number and NUS email address), assumptions you made in system modeling (if any). For each of the three questions:
       i. Explain the LTL property used.
       ii. 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.
     - **readme.txt:** It should contain all information required for me to reproduce any verification runs you have done using SPIN.
     - **ListofFiles.pdf:** A top-level explaining each file in the folder
  2. Zip the entire YourStudentNumber folder (including the folder itself and all files in it) into a file YourStudentNumber.zip.
  3. Submit YourStudentNumber.zip to the Luminus Folder **SPIN Submission**

Question 1 [5 marks]

Problem Statement

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]);
    }
}
```

Questions

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. **(1 mark)**

2. The above property is not true for our protocol, augment the provided model to make the above property true. 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. **(4 marks)**

Question 2: (courtesy Gerard Holzmann) [9 marks]

Problem Statement:

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."

Questions

1. Write a Promela model of a network node with three incoming streams of messages, and one outgoing stream. **(5 marks)**

- 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.)
- Each of the incoming data streams produces only messages of one specific type (say red, green, and blue for the three streams)
- Use handshake for the incoming messages and store them inside the node in local buffer

2. Explain how the reassembly deadlock can occur for the given model, provide a detailed description (textual explanation) for your answer. **(2 marks)**
3. Define the key property in Linear-time temporal logic (LTL), use SPIN to prove that it can be violated, and show the counter-example. **(2 marks)**

Question 3 [11 marks] (from Center TRACON Automation Systems)

Problem Statement

A weather update controller consists of a weather control panel (WCP), a number of weather-aware clients, and a communication manager (CM) which controls the interactions between the WCP and all connected clients. Two standard behaviors of this system are as follows:

- Client Initialization

    1. A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.
    2. If the CMs status is idle when the connecting request is received, it will set both its own status and the connecting client's status to pre-initializing, and disable the weather control panel so that no manual updates can be made by the user during the process of client initialization. Otherwise (CMs status is not idle), the CM will send a message to the client to refuse the connection, and the client remains disconnected.
    3. When the CM is pre-initializing, it will send a message to instruct the newly connected client to get the new weather information, and then set both its own status and the client's status to initializing.
    4. If the client reports success for getting the new weather, the CM will send another message to inform the client to use the weather information, and then set both its own status and the client's status to post-initializing.

Otherwise, if getting new weather fails, the CM will disconnect the client and set its own status back to idle.

5. If the client reports success for using the new weather, this initialization process is completed. the CM will set both its own status and the client's status to idle, and re-enable the WCP so that manual weather update is allowed again. Otherwise, if using new weather fails, the CM will disconnect the client, re-enable the WCP, and set its own status back to idle.

- Weather update

1. User can manually update new weather information only when the WCP is enabled. By clicking the update button on the WCP, an update message is sent to the CM.
2. When the CM is idle and receives update request from the WCP, it will set its own status and all the connected weather-aware client's status to pre-updating, and disable the WCP from any further updating requests before the completion of current update.
3. When CMs status is pre-updating, it will send messages to instruct all connected clients to get the new weather information, and then set its own status and the client's status to updating.
4. If all the clients report success for getting the new weather, the CM will send messages to inform the clients to use the new weather information, and then set its own status and the client's status to post-updating. Otherwise, if any of the connected clients reports failure for getting the new weather, the CM will send messages to all clients to use their old weather information, and then set its own status and the client's status to post-reverting.
5. When CMs status is post-updating, if all the clients report success for using the new weather, the updating is completed. The CM will set its own status and the client's status to idle, and re-enable the WCP. Otherwise, if any of the connected clients reports failure for using the new weather, the CM will disconnect all connected clients, re-enable the WCP, and set its own status back to idle.
6. When CMs status is post-reverting, if all the clients report success for using the old weather, the reverting is completed. The CM will set its own status and the client's status to idle, and re-enable the WCP. Otherwise, if any of the connected clients reports failure for using the old

weather, the CM will disconnect all connected clients, re-enable the WCP, and set its own status back to idle.

More details

- You can assume any fixed number of clients.
- Initially, the WCP is enabled for manually weather updating.
- The CM is at its idle status, and all the clients are disconnected.

Questions

1. Write a Promela model for the above controller.
   - Correct implementation of client initialization **(3 marks)**
   - Correct implementation of weather update **(3 marks)**
2. The key property of this system is to be able to propagate the latest weather update to all connected clients via the communication manager. Define the key property in Linear-time temporal logic (LTL) **(1 mark)**
3. Show that there exists a deadlock and provide a clear interpretation of the counter-example obtained from SPIN. Any additional problems you find in the protocol will of course distinguish your answer and earn more credit **(2 marks)**
4. Implement a solution which makes the model deadlock free and verify the deadlock free property. (**2 marks**)