CS4211 Assignment - II

October 25, 2021

Notes

- This assignment is due before 9:59 PM, Sunday, 14th November, 2021. No late submissions!
- Tool to be used: NuSMV model checker `https://nusmv.fbk.eu`
- This is an individual assignment. Acts of plagiarism are subjected to disciplinary action by the university. Please refer to `https://www.nus.edu.sg/celc/programmes/plagiarism.html` 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 three folders inside it:
       * **Problem 1:** Include your translated NuSMV source file needed for Question 1.
       * **Problem 2:** Include your implemented NuSMV source file needed for Question 2.
       * **Problem 3:** Include your implemented NuSMV source file needed for Question 3.
     - **report.pdf:**
       * For Question 1 (Part B), explain the LTL formula defined and describe how it can be used to show the desired property explained in the question.
       * For Question 2 (Part B), explain your answer. If you used any LTL properties to support your answer, explain the output of NuSMV.
       * For Question 3 (Part B), explain your answer. If you used any LTL properties to support your answer, explain the output of NuSMV.
     - **readme.txt:** It should contain all information required to reproduce any verification runs you have done using NuSMV.
  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 Luminus Workbin Folder **Lab2**.

Problem 1 [7 marks]

Problem Description

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
#define limit 3

bool busy[2];
int count = 0;

mtype = {start, stop, data, ack};
chan up[2] = [1] of { mtype };
chan down[2] = [1] of { mtype };

proctype operator(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[1-id]  -> busy[id] = true };
                out!start;
                in?ack;
                do
                        :: (count < limit) -> out!data; count++ ; in?data
                        :: (count <= limit) -> out!stop; count = 0; break
                od;
                in?stop;
                busy[id] = false
        od
}

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

Questions

A. Translate above Promela code into SMV code while adhering the protocol defined. **(5 marks)**

B. 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 and show whether it holds true using NuSMV. **(2 marks)**

Problem 2 [4 marks]

Problem Statement

The problem of ensuring mutually exclusive access to a shared resource has attracted the attention of computer scientists for many decades. Here is a candidate solution with two processes which uses atomic swap operations. The code for $process_1$ is shown below; it uses a local variable `key1` and a shared variable `lock`, both boolean. The code for $process_2$ is the same, except that `key1` is replaced by `key2`, a boolean variable local to $process_2$. You should assume asynchronous composition of processes and that in each time step, one line of the code gets atomically executed. The procedure swap simply swaps the value of its two arguments.

```
loop forever {
    /* noncritical section */
    key1 := false;
    while (key1 = false) do
        swap(key1, lock);
    /* critical section */
    lock := true
}
```

Questions

A. Using NuSMV model checker show that the above solution preserves mutual exclusion.

- Write a SMV model to implement above solution **(1 mark)**
- Using a LTL formula verify whether above solution preserves mutual exclusion **(1 mark)**

B. Using a Linear time temporal logic (LTL) property show that this is not a good solution to the mutual exclusion problem. Show your proof using NuSMV and explain your answer **(2 marks)**

Problem 3 [4 marks]

Problem Statement

The mutual exclusion problem is an extremely popular one. Here is a candidate solution to the problem for two processes. The process have `id` 0 or 1. The code for process `i` (`i` = `0` or `1`) is shown below. The global variables are `b[0], b[1]` is boolean and `k` is an integer. The composition of the processes is asynchronous and each simple statement of the below pseudo-code is executed atomically. (Assume `b[i]` = `false` initially)

```
C0: b[i] := true;
    k = 1 - i;
C1: if k != i and b[1-i])
                then go to C1;
    else /*critical section*/;
    b[i] := false;
    /* non-critical section */;
    go to C0;
```

Questions

A. Using NuSMV model checker show that the above solution preserves mutual exclusion.

- Write a SMV model to implement above pseudo-code **(1 mark)**
- Using a LTL formula verify whether above solution preserves mutual exclusion **(1 mark)**

B. The above candidate solution is starvation-free.

- Write a LTL formula to show the solution is starvation free **(1 mark)**
- Show proof using NuSMV and explain your answer **(1 mark)**