Weryfikacja wspomagana komputerowo

2016-2017

Spin project – Yet another mutual exclusion algorithm

Vincent Penelle

Rules of the project

- Make a Promela model of the exposed problem
- Make a pdf document answering the questions asked (and explaining difficulties, if you had any)
- Send the project to `penelle@mimuw.edu.pl` before 12/03/2017. A .tar (or .tar.gz) archive containing everything with your name as title would be very appreciated. The detail of what I expect are found below.

Exercise 1: (Yet another mutual exclusion algorithm) The goal of the project is to model and verify another algorithm enforcing the mutual exclusion property, but this time, for an arbitrary number of processes.

Consider N identical processes (i.e. they all have the same algorithm). They possess a non-critical section and a critical section (i.e. the section in which they access to the global variables and need not to be disturbed while doing so). The goal of the algorithm is to ensure that at any time at most one of the processes is in its critical section.

We propose to implement the Szymanski's protocol to enforce mutual exclusion. There is a global array of length N that can take values between 0 and 4, referred to as *flag*. The idea is that *flag[i]* indicates the status of process i. The protocol is given in pseudo-code:

```
1   loop forever do
2   begin
3   l1: Noncritical section
4   l2: flag[i] := 1;
5   l3: wait until (flag[0] < 3 and flag[1] <3 and ··· and flag[N−1] <3)
6   l4: flag[i] := 3;
7   l5: if (flag[0] = 1 or flag[1] = 1 or ··· or flag[N−1] = 1)
8       then begin
9           l6: flag[i] := 2;
10          l7: wait until (flag[0] = 4 or flag[1] = 4 or ··· flag[N−1] = 4)
11      end
12  l8: flag[i] = 4;
13  l9: wait until (flag[0] <2 and flag[1] <2 and ··· flag[i−1] <2)
14  l10: Critical section
15  l11: wait until (flag[i+1] ∈ {0,1,4} and ··· flag[N−1] ∈ {0,1,4})
16  l12: flag[i] := 0;
17  end
```

1. Try to understand informally what the protocol is doing (make drawings). Make an informal explanation of what you understand (and what you don't). Don't cheat: make that question first and do not modify it after the rest of the project.

2. Model the protocol in Promela. You will assume that all tests on flag are atomic (bonus question: why should you assume that? what if you don't?). Make the protocol description easy to change when you change N (i.e. no long list of flag[1] ¡N && flag[2] ¡N, etc).

3. Check for several values of N (at least 2) that the protocol indeed ensures mutual exclusion. You are free to add some global variable, assert statements or LTL formula to do so (actually, I don't see how you do that without). Report your results for $N = 4$ (true or false, number of states, memory used)

4. The code a process has to go through before reaching the critical section can be divided into several segments. We refer to statement l4 as the doorway, to segments l5, l6, and l7 as the waiting room and to segments l8 through l12 (containing the critical section) as the inner sanctum. You are requested to check the following basic claims using assertions (but you may want to use a parallel process checking it). You will take care that if you change the value of N, the written model does not change (or not much). Give for each case the changes to your original Promela specification, and present the verification results. In case of negative results, simulate the counterexample by mean of guided simulation.

   (a) Whenever some process is in the inner sanctum, the doorway is locked, that is, no process is at location l4.

   (b) If a process i is at l10, l11 or l12, then it has the least index of all the processes in the waiting room and the inner sanctum.

   (c) If some process is at l12, then all processes in the waiting room and in the inner sanctum must have flag value 4.

In the end, you should send me:

- A pdf file detailing the answer to the questions, along the verification results when asked for. For the verification results, give the options you've used, the number of states and memory usage. It will also detail what the other files are and the links between them (you still should have clear names).
- A promela model for question 2.
- A promela model for question 3.
- A promela model for each element of question 4.
- For every failed claim, the error trail (that is a file in .trail that should appear in the folder you ran ispin). Name it appropriately (e.g. the number of the question – or the name of the claim).