Weryfikacja wspomagana komputerowo

2016-2017

Lab 5 – Goats, wolves, and more leaders

Vincent Penelle

Topics of this Lab

- A complete program,
- A more clever election leader algorithm.

Exercise 1: (Goats and wolves)

Let us conceive a program to solve a consequent problem.

G goats and W wolves travelling together come across the Vistula and want to cross it. There is no bridge, but a boat is on the shores. It can welcome L animals at the same time (who can drive the boat, yes. You are not require to explain how they do that). The trouble is that if at any time in the boat or one of the banks there are strictly more wolves than goats, the predator nature of these otherwise nice animals will take over. And we don't want any casualty.

- Create a promela program modelling this problem. You are strongly encouraged to use several process, for example one to load animals to the boat, one to unload animals, one to move the boat, etc. And leave the non-determinism take care of the rest. You should as well use *atomic* to prevent too much strange and useless behaviours.
- Use LTL formulæ to check the behaviour of your program and to test if there is a solution or not. Hint 1: With G=W=3 and L=2, there is a solution (but search for other). Hint 2: You want to check the existence of a successful path, but Spin is able to check if a formula is satisfied for all path. How could you do?
- Use a never claim to find a solution to the program. Hint: you want a claim that fails when you have a solution.

Exercise 2: Another leader election algorithm Last week, we had a big number of maximum message exchanged. We will try to lower that number.

The idea is that now, each process will have two states: active and passive. At the beginning, everybody is active.

If a process is active, he takes two messages of its two predecessors (p_1 then p_2) and compare them with his number n. If p_1 is bigger than the two others, it updates its number to p_1 and stays active. Otherwise, it becomes passive.

If a process is passive, he just passes down whatever he receives.

If an active process receives a message equal to its number as its first message, he concludes that he is the only left active process, send a special message of victory, and then every process passes it down, determining on the way if he is the leader or not (by comparing with its initial number), and when the victory message reaches the sender, he shuts down.

1. Write an algorithm (or a drawing, or both) explaining this algorithm.
2. How many messages are passed at worst? At best?
3. Here[1] is an implementation of this algorithm. Test it.
4. Verify the previous claims (and compare with the results of complexity of it).
5. Verify with a LTL property than every process always receives a alternation of messages with one and two, until it receives a victory message.
6. What else can you check?

[1] `./leader.pml`