Homework assignment (model checking)

Peterson's mutual exclusion algorithm for *N* processes

Teaching assistant: **Tamás Tóth**

Problem description

Peterson's algorithm [1] (or Peterson's solution) is a concurrent programming algorithm for mutual exclusion that allows two or more processes to share a single-use resource without conflict, using only shared memory for communication [3]. The algorithm is implemented for *N* processes [2] as follows.

Global variables:

```
// level is an array of N integers form the range [-1..N-1]
// with each element initialized to -1
level : array [0..N-1] of [-1..N-1] := { -1 };

// waiting is an array of N-1 integers form the range [-1..N-1]
// with each element initialized to -1
waiting : array [0..N-2] of [-1..N-1] := { -1 };
```

Pseudocode for the *i*-th process:

```
// Entry protocol
for (l := 0; l < N-1; ++l) {
    level[i] := l;
    waiting[l] := i;

    // wait for condition to hold
    await (waiting[l] ≠ i ∨ ∀j ∈ [0..N-1] : (j ≠ i → level[j] < l));
}

// Critical section

// Exit protocol
level[i] := -1;
```

Prepare the model of the algorithm while conforming to the following constraints:

- Executing an assignment (e.g. `level[i] := l`) or evaluating an atomic formula (e.g. `waiting[l] ≠ i`) is an atomic operation. Any operation more complex than this should not be considered atomic.
- The processes are scheduled by a fair scheduler that steps one of the *N* processes in each time step. It should hold in every *M*-th time step for each process that the process has been stepped at least *K* times and at most *L* times (thus $L \cdot N \geq M \geq K \cdot N$) since the last check (M time steps before). Besides this constraint, the behavior of the scheduler should not be restricted.
- The model should be parametric in *K*, *L*, *M* and *N*.

Properties to check

Verify the correctness of the algorithm for at least three processes ($N \geq 3$) by formalizing the following properties as formulas in temporal logic. In case of a property violation, explain why the property fails based on a counterexample.

1. The system is deadlock free.
2. Mutual exclusion is enforced.
3. The critical section is reachable for some processes.
4. The system is starvation free.

Hints

For the modeling of the scheduler consider using the *select* construction.

The complete solution of the problem involves a functioning scheduler as described above. Nonetheless, a solution without scheduler is still admissible (with a lower grade).

References

[1] G. L. Peterson: Myths about the mutual exclusion problem. Information Processing Letters, Vol 12, pp. 115–116, 1981.

[2] M. Hofri: Proof of a mutual exclusion algorithm – a classic example. Operating Systems Review, Vol. 24, Nr. 1, pp 18-22, January 1990.

[3] https://en.wikipedia.org/wiki/Peterson%27s_algorithm