Homework assignment

Eisenberg & McGuire mutual exclusion algorithm

Teaching assistant: **Tamás Tóth**

Problem description

The Eisenberg & McGuire algorithm [1] 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. The algorithm can be implemented as follows.

Global variables

```
// flags is an array of enum literals
// each element is initialized to IDLE
{IDLE, WAITING, ACTIVE} flags[0..N - 1] = { IDLE };

// turn is an integer between 0 and N - 1
int[0..N - 1] turn := 0;
```

Pseudocode for the *i*-th process

```
// Local variables
[0..N] j := 0;

// Entry protocol
repeat {
    flags[i] := WAITING;

    j := turn;
    while (j ≠ i) {
        if (flags[j] ≠ IDLE) {
            j := turn;
        } else {
            j := (j + 1) mod N;
        }
    }

    flags[i] := ACTIVE;

    j := 0;
    while (j < N ∧ (flags[j] = ACTIVE → j = i)) {
        j = j + 1;
    }

} until (j ≥ N ∧(flags[turn] ≠ IDLE → turn = i));

turn := i;
```

```
// Critical section

// Exit protocol
j := (turn + 1) mod N;
while (flags[j] = IDLE) {
    j := (j + 1) mod N;
}

turn := j;
flags[i] := IDLE;
```

Note that ∧ and → are the Boolean operators AND and IMPLY.

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

- Executing an assignment (e.g. `x[i] := false`) or evaluating of an atomic formula (e.g. $j \neq i$) is an atomic operation. **Any operation more complex than this should not be considered atomic**.
- The model should be parametric in *N*, that is, it should be possible to scale up the specification for more processes by modifying a single constant.

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. Make sure to understand the output of the verifier in each case, and correct your model if necessary. 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. **Note**: In order to obtain meaningful results from the verifier, you might have to extend your model with a scheduler that enforces progress. Moreover, in order to avoid starvation, the scheduler might have to ensure some sort of fairness.

Note that it is allowed to submit the homework without the successful verification of property 4, but this is not a solution of full value (i.e., the grade of the homework will be less than 5).

References

[1] M. A. Eisenberg, M. R. McGuire: Further comments on Dijkstra's concurrent programming control problem. Communications of the ACM, Vol 15 Issue 11, pp. 999, 1972.