Formal modelling and verification

You have to create formal models and verify them in the UPPAAL tool. The assignment consists of two parts, at first you have to create an untimed model and then extend it with timing information.

Documents: you have to create the **formal models**, the **verification queries** and you also have to write a **summary** (the summary should be in *doc* or *pdf* format). All these documents have to be compressed into a zip file and be sent before the deadline! The summary has to contain the following information: the **detailed models** and their **explanation**, the **modelling decisions**, the **verification queries** and their **evaluation**, with also the **explanation of the results**! We are also evaluating the outline of the document, so please try to prepare nice documents!

The evaluation is the following. We can only deal with solutions containing both parts of the modelling phases and have all queries defined according to the specification! In the final evaluation you have to be able to defend your modelling and verification decisions! In addition, you might be asked to modify your specification/model!

Operating system

Engineers implemented the following protocol in an operating system to ensure the mutual exclusion so that no two threads are in their critical section at the same time, no two threads access to a shared resource simultaneously. In order to ensure this property, the designers implemented a controller, which handles the requests and lets the threads to enter their critical section.

The working of the protocol is the following. When the operating system is started, it creates two binary (Boolean) variables for each thread in the memory. Thread *i* will use these two variables r_i and g_i to communicate with the controller. These variables are initialized to "*FALSE*". Thread *i* sets the value of variable r_i to "*TRUE*" if it wants to enter the critical section. The controller can read the variable *r* of every thread. If the controller recognizes that thread *i* has set its variable r_i to "*TRUE*", it can grant the right for thread *i* to enter the critical section by setting the variable g_i of thread *i* to "*TRUE*". The thread periodically checks if its variable *g* had been set "*TRUE*", then the thread enters its critical section and uses the shared resources. When thread *i* leaves the critical section, it sets the variable r_i to "*FALSE*" and it proceeds with its normal working. The controller periodically checks the variable r_i, and if it is "*FALSE*", it drawbacks the grant from thread *i* by setting variable g_i to "*FALSE*".

The implementations are the following (pseudo) codes. Each line is an atomic operation in itself.

thread$_i$	**controller**
1 … other activity …	1 **wait until** at least one r_i is *TRUE*
2 r_i := *TRUE*	2 let *c* be such that r_c = *TRUE*
3 **wait until** g_i = *TRUE*	3 g_c := *TRUE*
4 critical section	4 **wait until** r_c = *FALSE*
5 r_i := *FALSE*	5 g_c := *FALSE*
6 **go to** 1	6 **go to** 1

If more than one thread set their variables r_i to "*TRUE*", the controller chooses nondeterministically from them (in line 2 of controller).

Let assume that our operating system runs 3 threads and one controller.

Phase1: Modelling and verification

Create the formal model of the system in UPPAAL! Does your model have the same behaviours as the pseudo codes?

Formalize the following specification properties with the help of temporal logic and verify your system (if the system satisfies the property, explain why, if the system violates the property, explain the counterexample, why did the model violate the specification)!

1. The model does not have deadlock!
2. Is it possible for every thread to enter their critical section?
3. Is it true that at most one of the threads can be in the critical section?
4. Is the system starvation and livelock free?

Phase2: Modelling and verification the timed system

Modify your (system) model! We know that each operation of the thread requires 1 time unit and it stays in the critical section at least 8 and at most 10 time units. After leaving the critical section, the thread needs at least 8 time units to process the results, so it cannot request access in the next 8 time units.

The controller requires 2 time units for each operation (so the operation "**wait until**" checks the given variable at every 2 units).

Create the formal model of the system in UPPAAL!

Check the requirements on the timed model:

1. The model does not have deadlock!
2. Is it possible for every thread to enter their critical section?
3. Is it true that at most one of the threads can be in the critical section?
4. Is the system starvation and livelock free?