Lab session: Timed automata / Uppaal

To answer these questions, use the `uppaal` tool installed in `/matieres/5MMMVSC7/UPPAAL`; the command `/matieres/5MMMVSC7/UPPAAL/uppaal` launches the graphical interface. You may also install `uppaal` on your own computer.

We consider the system consisting of the three timed automata (Agent_1, Agent_2, and Mutex) depicted in Figure 1. Each transition is labeled by a 3-tuple of the form "g / a ; r", where g is the guard (**tt** denotes the guard that is always satisfied), a is the synchronisation, and r is the statement to be executed if the synchronisation occurs (**null** means nothing to do). State invariants are written inside states. Initial states are depicted with a double line.

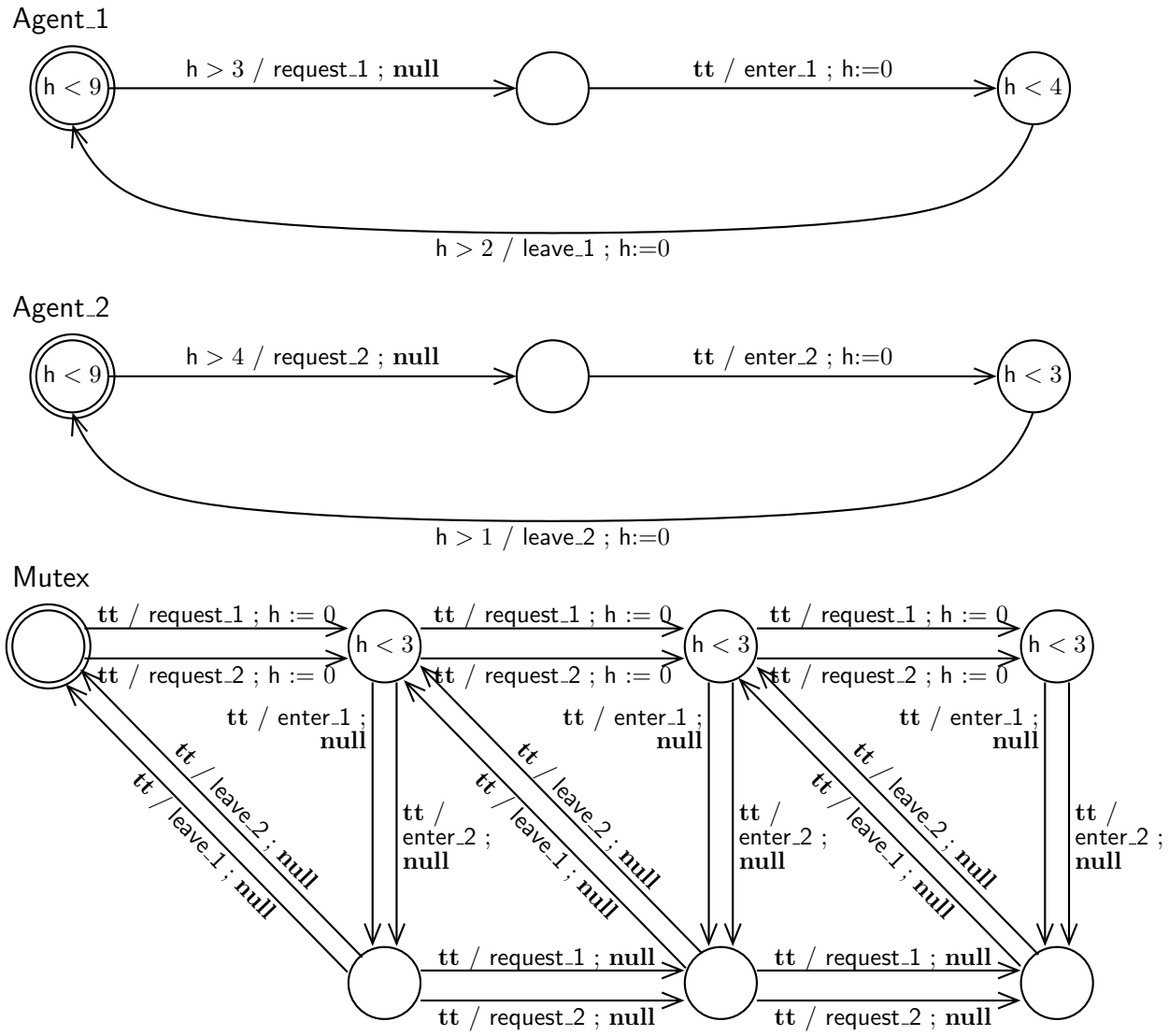

Figure 1: Timed automata network

Each of these automata has a local clock h. The three automata synchronise two by two on actions request_1, request_2, enter_1, enter_2, leave_1 and leave_2, i.e., the parallel composition could be described by the following expression:

$$\left(\mathsf{Agent\_1} \otimes \mathsf{Agent\_2} \right) \otimes_{\{\mathsf{request\_1},\mathsf{request\_2},\mathsf{enter\_1},\mathsf{enter\_2},\mathsf{leave\_1},\mathsf{leave\_2}\}} \mathsf{Mutex}$$

The guards and invariants express the following constraints:

- Agent_1 stays between 3 and 9 time units in its non-critical section and between 2 and 4 time units in its critical section.
- Agent_2 stays between 4 and 9 time units in its non-critical section and between 1 and 3 time units in its critical section.
- Mutex stays at most 3 time units without interacting with other processes before sending a response.

You will have to model this system using the Uppaal editor ("Editor" tab). An automaton can be described as a template. Use "Edit/Insert template" to create a new automaton. The name of each template can be changed in the "Name" field showing in the rightmost frame when you select the template.

There are global declarations ("Declarations" item just below the "Project" item in the leftmost frame) and declarations that are local to a template ("Declarations" item just below each "Template" item — you may have to expand by clicking on the node next to the "Template" item to see the corresponding "Declarations" item). Clocks must be declared locally to each automaton, whereas channels (request_1, request_2, enter_1, enter_2, etc.) must be declared globally. The following is a valid channel declaration:

```
chan request_1, request_2;
```

The following is a valid clock declaration:

```
clock h;
```

To edit an automaton:

- In every template, there is a predefined initial state.
- You can add new states by clicking the button showing a plain circle in the top menu, and then clicking in the rightmost frame where you want to insert the new state.
- You can add transitions by clicking the button showing a small red arrow, then clicking on the source state, and then draw the arrow towards the target state while keeping the mouse button pressed.
- Transitions can be curved using the rightmost button showing a small green circle on a line (called a nail).

You can attach contents to states and transitions by clicking the button showing a big blue arrow (which you can also use to move states), then right clicking on a state or on a transition, and finally selecting "Edit location" or "Edit edge":

- In states, only the "invariant" and "initial" fields are relevant.
- In transitions, the "select" field is not relevant.

- In the "sync" field of transitions, you have to decide whether the aumaton is sender or receiver. Suggestion is that agents be senders, whereas the mutex be receiver. If so, you will have to write actions of the form "leave_1!" in agents and "leave_1?" in the mutex (and similarly for the other actions).

At last, in the "System declarations" item, you must declare the automata composition as follows:

```
system Agent_1, Agent_2, Mutex;
```

Once your model is described, you can click the "Simulator" tab (answer Yes if Uppaal asks you whether it has to upload the model). If there are errors, you will see them in the lower frame of the "Editor" tab.

Question 1 Symbolic execution

Is the following sequence of synchronisations possible in this system?

request_2, enter_2, leave_2, request_1, request_2, enter_2

If so, what are the possible values of the clocks Mutex.h, Agent_1.h and Agent_2.h after executing this sequence?

To answer this question, click on the "Simulator" tab. You can select a synchronisation in the "Enabled transitions" frame and then click "Next". Just follow the above sequence. Constraints on clock values can be seen in the middle frame (you may have to expand the "<Constrains>" item by clicking on the node next to it).

Question 2 Verification

Can the system deadlock?

If so, explain why and find a way to correct the system to avoid any deadlock, without changing the time constraints described above.

You can check for the absence of deadlocks by clicking on the "Verifier" tab. In the "Query" frame, type the property "A[] not deadlock" and then click on the "check" button. To get a counterexample, go to the "Options" menu, click on "Diagnostic trace/shortest", and check the property again. Then go to the simulator to see the counterexample trace. The deadlock state appears in red. Synchronization is impossible on transitions in red. Try to understand why by looking at the clock constraints, transition guards and state invariants (source and target). Finally find a way to enable this transition, without changing guards and invariants.