Correction of lab session on Uppaal

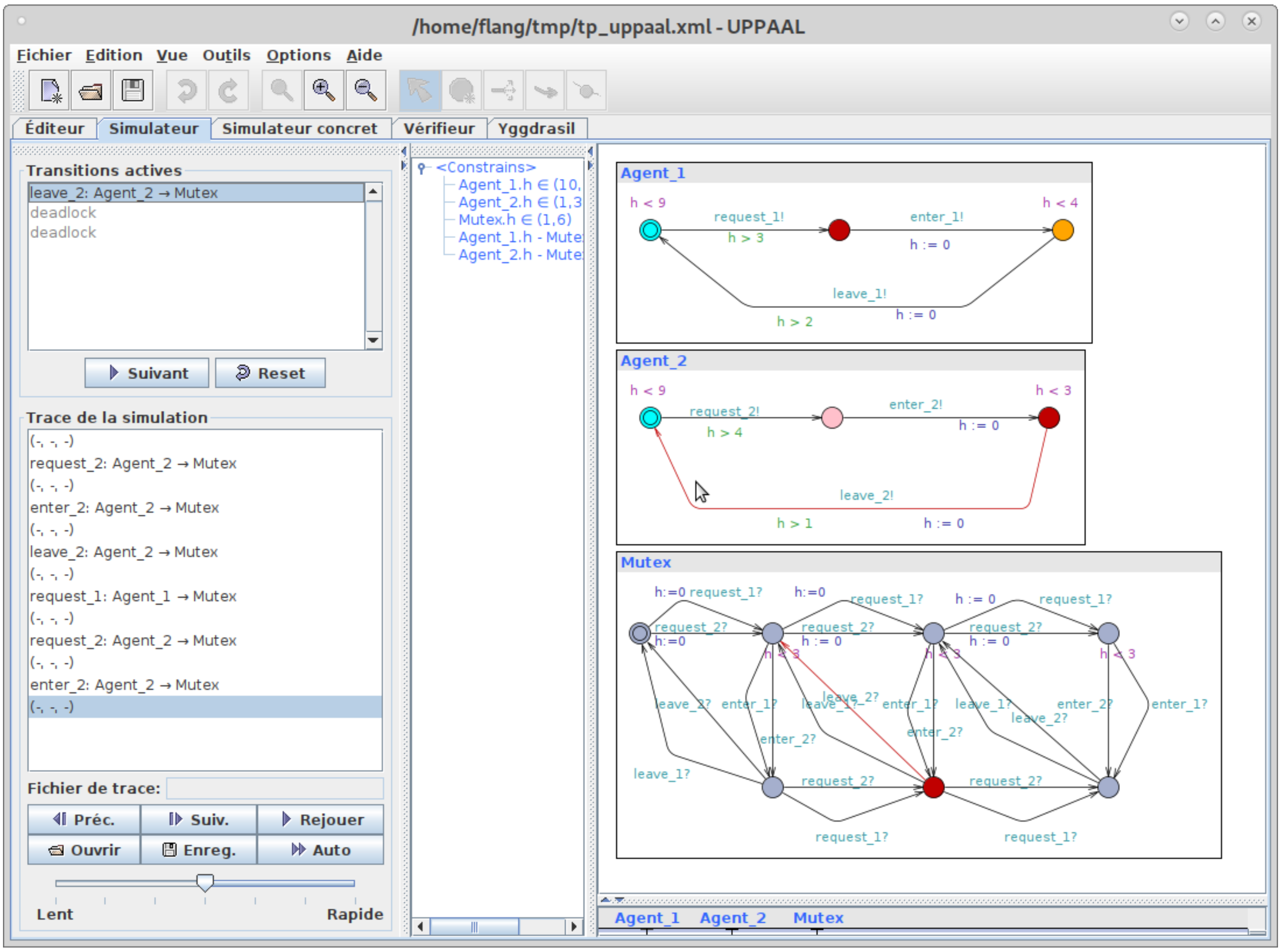
Question 1.

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?

Answer 1.

Yes, this sequence of synchronisations is possible, as shows the following screenshot :



/home/flang/tmp/tp_uppaal.xml - UPPAAL

Transitions actives

leave_2: Agent_2 → Mutex
deadlock
deadlock

Trace de la simulation

```
(-, -, -)
request_2: Agent_2 → Mutex
(-, -, -)
enter_2: Agent_2 → Mutex
(-, -, -)
leave_2: Agent_2 → Mutex
(-, -, -)
request_1: Agent_1 → Mutex
(-, -, -)
request_2: Agent_2 → Mutex
(-, -, -)
enter_2: Agent_2 → Mutex
(-, -, -)
```

Fichier de trace:

```
<Constrains>
  Agent_1.h ∈ (10,
  Agent_2.h ∈ (1,3
  Mutex.h ∈ (1,6)
  Agent_1.h - Mute
  Agent_2.h - Mute
```

The following constraints are satisfied (expand the « Constrains » tag in the middle window) :

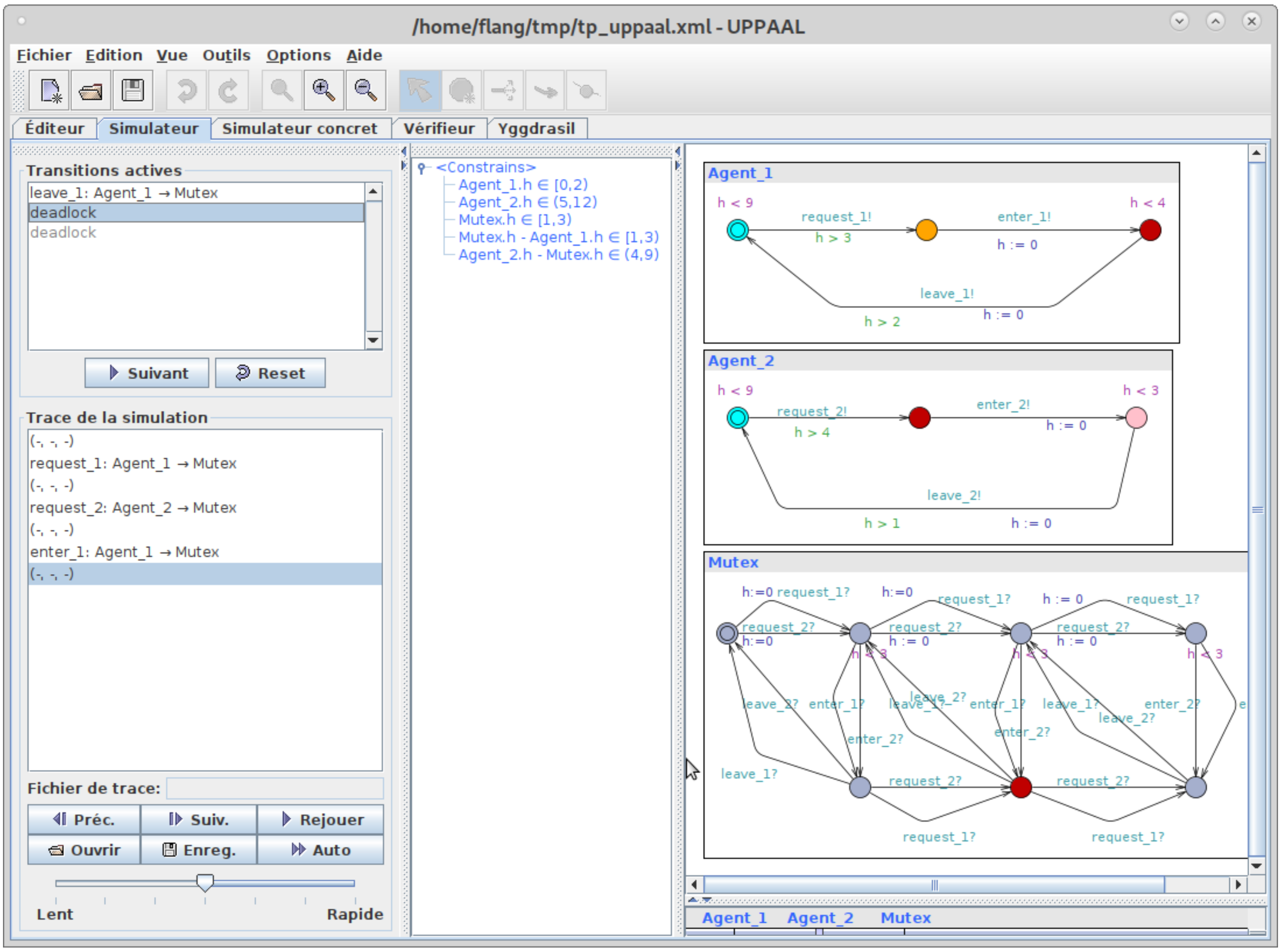
$10 < Agent\_1.h < 18$
$1 < Agent\_2.h < 3$
$1 < Mutex.h < 6$
$9 < Mutex.h - Agent\_1.h < 12$
$-3 < Agent\_2.h - Mutex.h \leq 0$

Question 2.

Can the system deadlock ?

Answer 2.

Yes it can. The following screenshot of the simulator window, after having checked the property shows possible deadlocks. Note that I selected « Options / trace de diagnostic (diagnostic trace) / la plus courte (shortest) ».



/home/flang/tmp/tp_uppaal.xml - UPPAAL

Transitions actives

leave_1: Agent_1 → Mutex
deadlock
deadlock

<Constrains>
- Agent_1.h ∈ [0,2)
- Agent_2.h ∈ (5,12)
- Mutex.h ∈ [1,3)
- Mutex.h - Agent_1.h ∈ [1,3)
- Agent_2.h - Mutex.h ∈ (4,9)

Trace de la simulation

(-, -, -)
request_1: Agent_1 → Mutex
(-, -, -)
request_2: Agent_2 → Mutex
(-, -, -)
enter_1: Agent_1 → Mutex
(-, -, -)

Fichier de trace:

The first deadlock appears when the system is in the red states and the following constraints are satisfied (select the first deadlock) :

$0 \leq$ Agent_1.h < 2
$5 <$ Agent_2.h < 12
$1 \leq$ Mutex.h < 3
$1 \leq$ Mutex.h - Agent_1.h < 3
$4 <$ Agent_2.h - Mutex.h < 9

This is indeed a deadlock state since:

- actions leave_2, enter_1, enter_2, request_1 and request_2 cannot take place because they are offered by at most one of the components ; therefore, no synchronisqtion can take place
- action leave_1 cannot take place because Agent_1.h < 2, whereas the guard in Agent_1 requires Agent_1.h > 2
- if time elapses beyond Agent_1.h == 2 (i.e., Agent_1.h ≥ 2), then leave_1 cannot take place either in the future, because the invariant of the target state of the transition labeled by

leave_1 in Mutex requires that $Mutex.h < 3$; by summing the constraint $Mutex.h – Agent\_1.h \geq 1$ with $Agent\_1.h \geq 2$, we obtain $Mutex.h \geq 3$, which contradicts this invariant

I do not detail the second deadlock.

A way to eliminate all deadlocks is to add a missing reset of h on the transitions labelled by leave_1 and leave_2 In Mutex. You can verify that the system is now deadlock-free.