Lab session: Uppaal

UPPAAL

- Networks of timed automata
- Integer variables: global and local
- Real-valued clocks
- Synchronisation by complementary actions

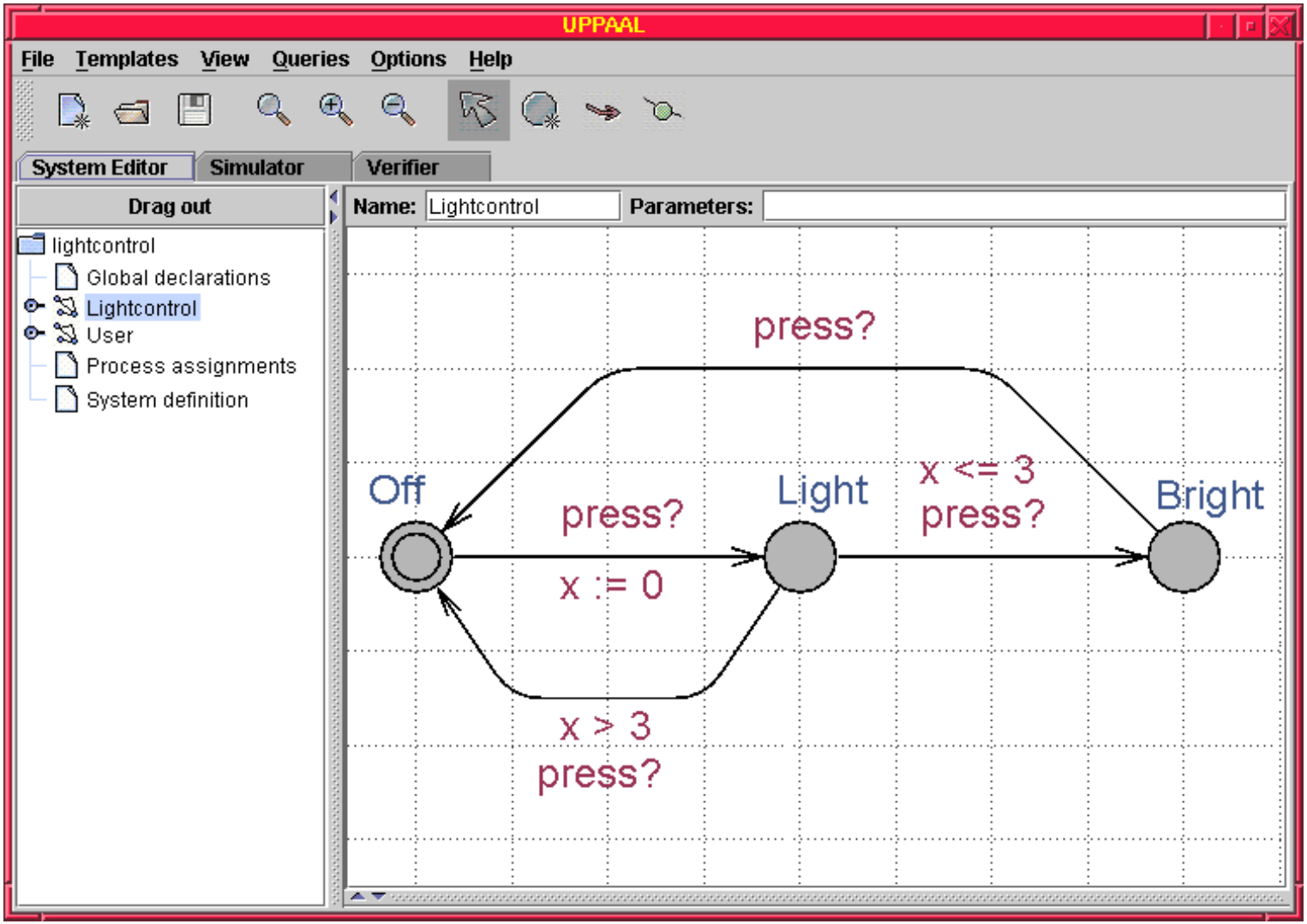
  emission « **a!** » synchronises with reception « **a?** »

UPPAAL: graphical interface

- Execute the command uppaal

  On the Ensimag server:
  /matieres/5MMMVSC7/UPPAAL/uppaal
- Three parts
  - System editor: networks of timed automata
  - Simulator
  - Verifier / Diagnostic generator
- Two types of files
  - System description: « `.xml` »
  - Set of properties: « `.q` »

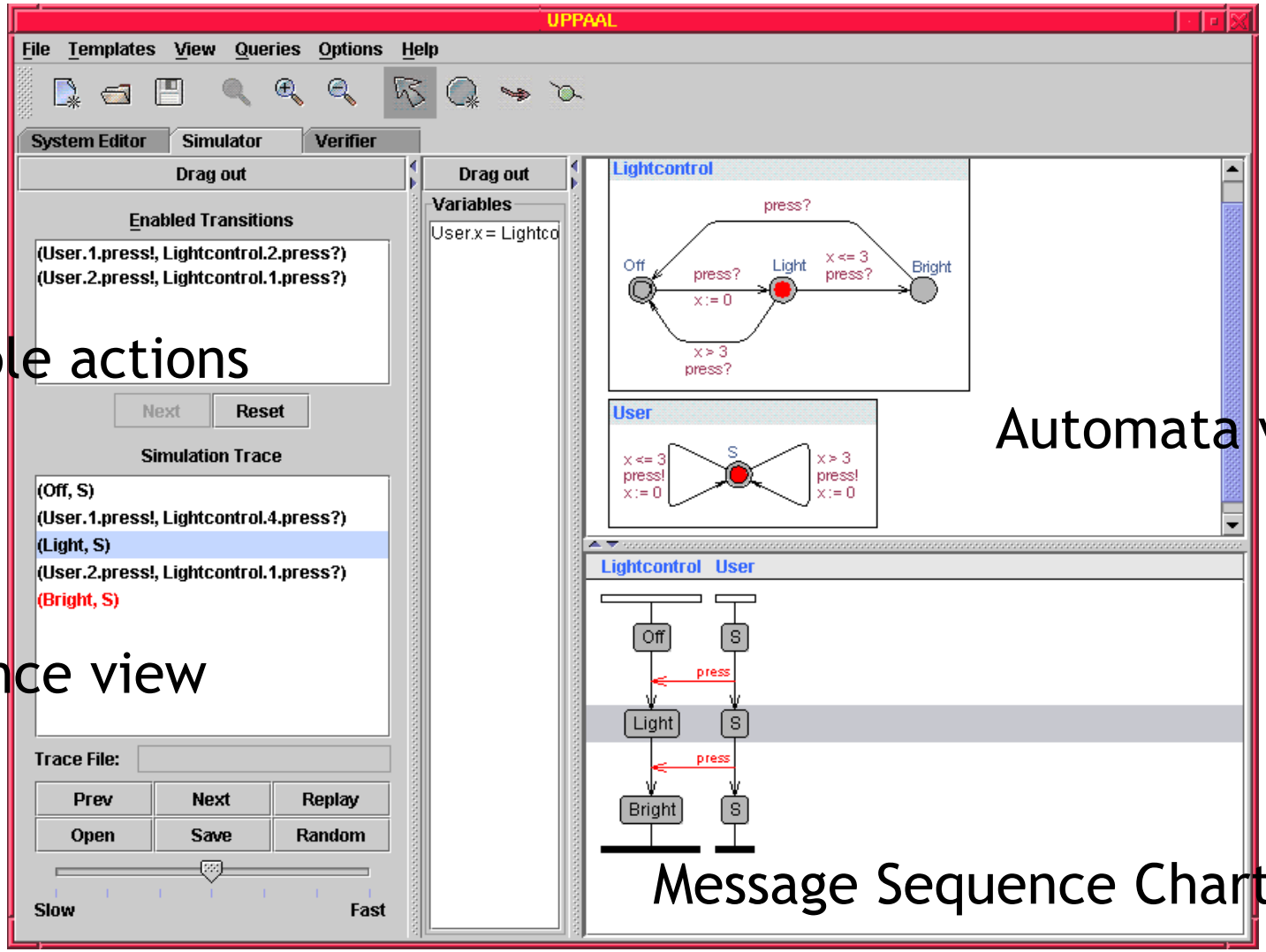
UPPAAL: system editor (1/2)



UPPAAL: system editor (2/2)

- Graphical representation of automata
  « templates » / processes
- Global and local declarations
  - Variables: « **`int`** »
  - Clocks: « **`clock`** »
  - Communation channels: « (**`urgent`**) **`chan`** »
- Process instantiation
  Assignment of automata parameters
- Description of the network
  Parallel composition of automata

UPPAAL: simulator



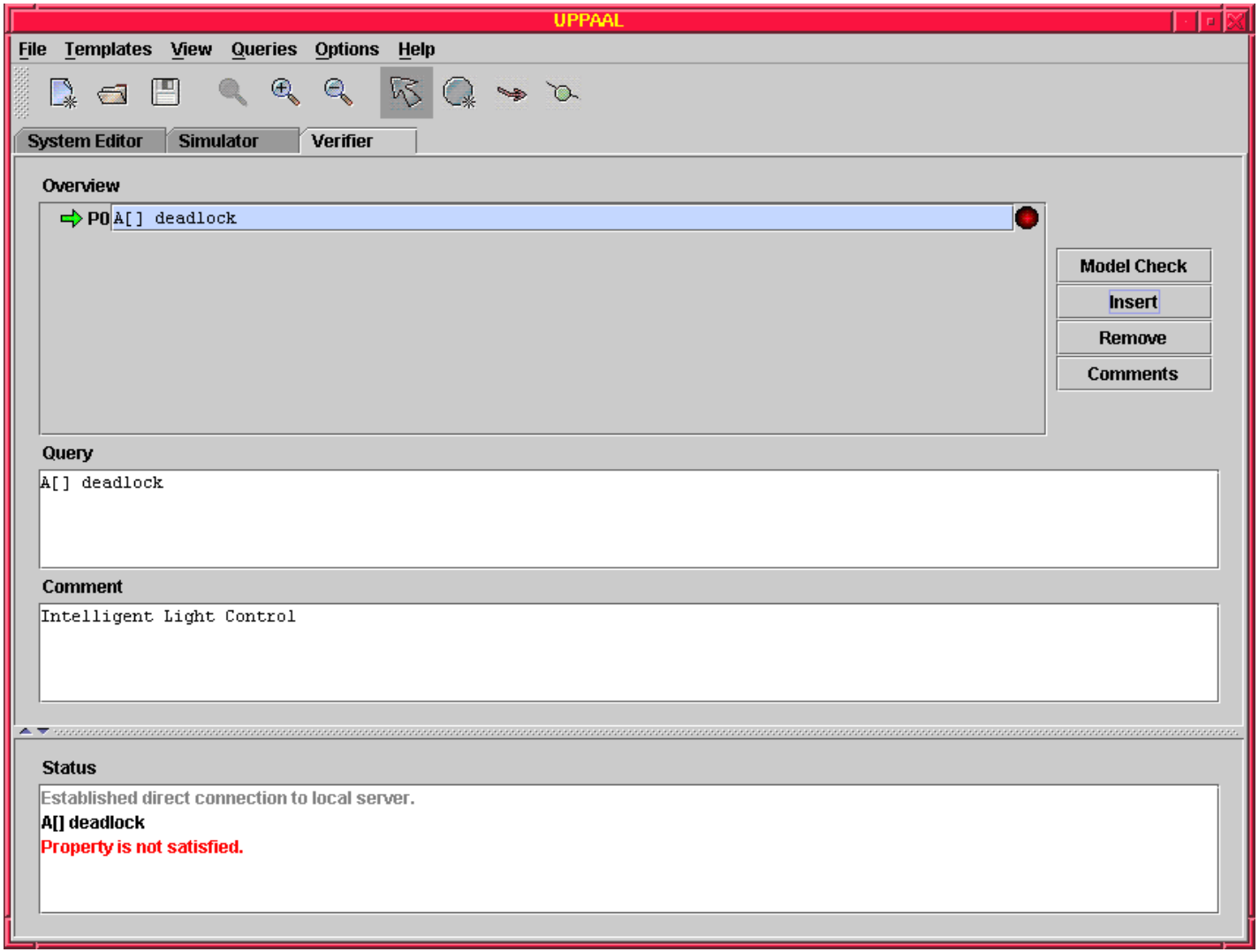
Possible actions

Automata view

Sequence view

Message Sequence Chart view

UPPAAL: verifier



UPPAAL: Language of properties

- Properties on states
  - Expressions without side effect: « x < y »
  - Automata state predicates: « proc.q »
  - System deadlock: « **`deadlock`** »
- Properties (subset of CTL)
  - « **`E<>`** p » : potentiality (« *Exists Finally* »)
    There exists a path leading to a state satisfying p
  - « **`E[]`** p » : trajectory (« *Exists Globally* »)
    There exists a path on which all states satisfy p
  - « **`A[]`** p » : invariance (« *Always Globally* »)
    On all paths, all states satisfy p
  - « **`A<>`** p » : inevitability (« *Always Finally* »)
    On all paths, a state satisfying p can be reached
  - « p_1 **`-->`** p_2 » ≡ **`A[]`** (p_1 **`imply`** (**`A<>`** p_2))

UPPAAL: Examples of properties

- 1 is invariably smaller than 2:

  A[] 1 < 2

- Deadlock freeness:

  A[] not deadlock

- If process « p1 » is in state « q1 », then process « p2 » cannot be in state « q2 »:

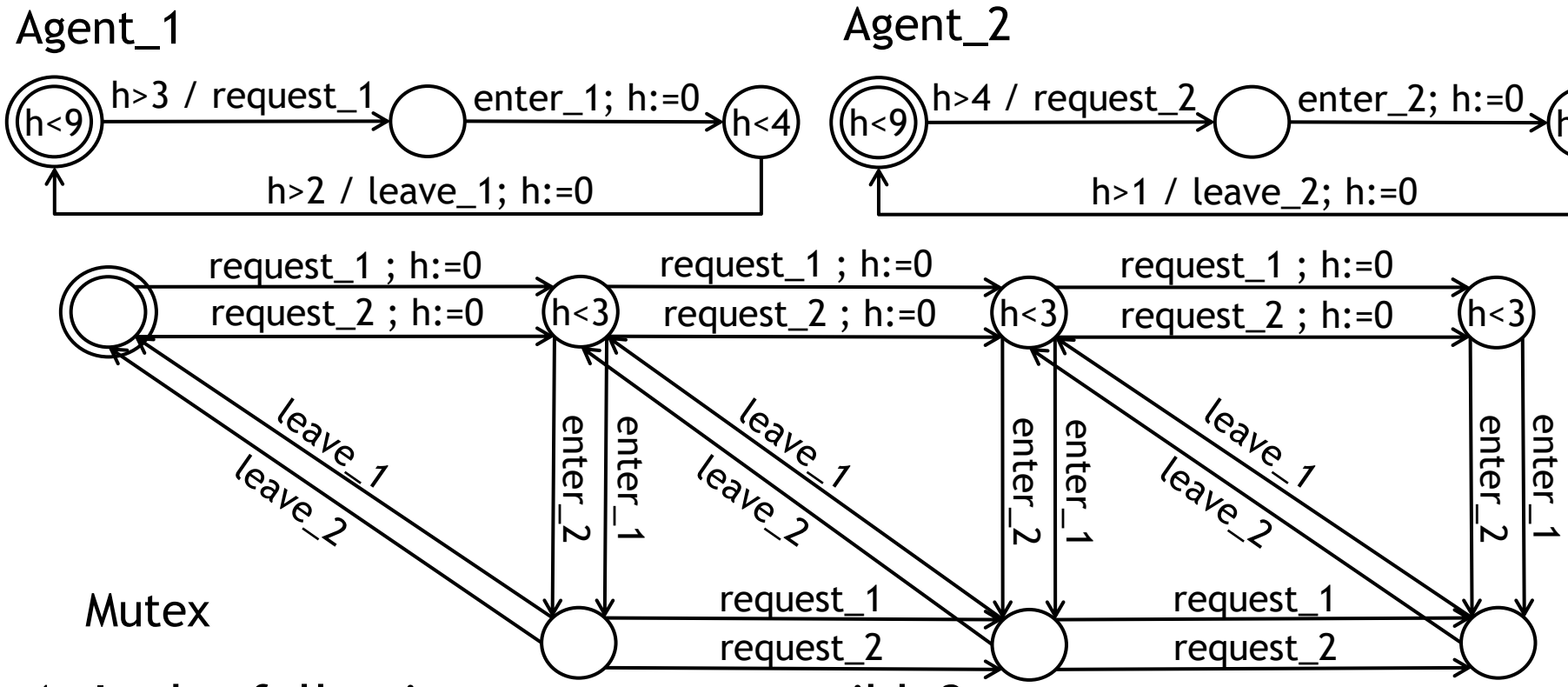
  A[] $p_1.q_1 + p_2.q_2 \leq 1$

- If process « p » is in state « q_1 », then « p » will inevitably reach state « q_2 »:

  $p.q_1 \dashrightarrow p.q_2$

/matieres/5MMMVSC7/uppaal-subject-english.pdf

(Agent_1 ||$_{\emptyset}$ Agent_2) ||$_{\{request\_i,\ enter\_i,\ leave\_i\}}$ Mutex



1. Is the following sequence possible?
   request_2, enter_2, leave_2, request_1, request_2, enter_2
2. What are the clock values?
3. Is there a risk of deadlock? If so, how to correct?

Uppaal: Important

- Actions (request_1, request_2, etc.) are named *channel* and must be declared in section Project/Declarations: **chan** request_1, request_2;
- Channel synchronisations are binary, identifying a sender and a receiver.
  Example : request1! request1?
- Clocks must be declared locally in processes: **clock** h;
- The parallel composition is to be described in section System declarations: **system** Agent_1, Agent_2, Mutex
- The Select field of transitions is useless in this lab session
- Full subject (also on Chamilo):
  /matières/5MMMVSC7/uppaal-subject-english.pdf