5.2 The Uppaal Model Checker
What can we do with UPPAAL?

- Analyze networks of timed automata
  - Model timed processes with timed automata
  - Simulate
  - Verify reachability properties
What can we do with UPPAAL?

- Appropriate for systems that can be modeled
  - as a collection of non-deterministic processes with
    - finite control structure and
    - real-valued clocks,
    - communicating through channels or shared variables
- Typical application areas include
  - real-time controllers and
  - communication protocols
  particular, those where timing aspects are critical
- Heavily used to formally prove properties for various types of protocols
  - No deadlocks
  - Mutual exclusion
  - ...

Modellbasierte Softwareentwicklung
Main idea

1. Model the system
2. Verify properties of the system
3. View diagnostic trace in simulator if property is not satisfied

$A$: System Description (Timed Automata)

$F$: Requirement Specification

Model Checker

A satisfies F?

Yes!!

No!
Diagnostic information viewed in simulator
UPPAAL tool box

- **Modeling**
  - Timed Automata

- **Verification**
  - Via automatic model checking
  - Enables formal proofs of system properties
  - Exhaustive search that covers every possible dynamic behaviour

- **Validation**
  - Via graphical simulation
  - Enables examination of possible dynamic executions of a system during early design (or modeling)

- **Joint research project**
  - **Uppsala University** and **Aalborg University**
What is it, that constitutes a system?

- Based on timed automata
  - Finite state machine with clocks
  - Clocks measure (continuous) time
    - Clocks can be tested for values or reset
- A system in UPPAAL consists of processes
  - A process is a timed automaton, i.e. a state machine with time
  - Processes can communicate via synchronisation (channels)
  - A system in UPPAAL consists of concurrent processes
How do we specify, what we want to verify in Uppaal? 1/3

- Simplified Version of CTL, does not allow nesting of path formulae. Allowed formulae are:
  - **State Formulae:**
    - Expression that can be evaluated for a state without looking at the behaviour of the model
    - Example: $i == 7$, is true in a state whenever $i$ equals 7
  - **Reachability Properties:**
    - $E<>\varphi$: there exists a path on which eventually $\varphi$ holds
How do we specify, what we want to verify in Uppaal? 2/3

- **Liveness Properties:**
  - $A<>\varphi$: for all paths eventually $\varphi$ holds
  - $\psi - ->\varphi$: whenever $\psi$ is satisfied, then eventually $\varphi$ will be satisfied
How do we specify, what we want to verify in Uppaal? 3/3

- **Safety Properties:**
  - \( E[\varphi] \): there exists a maximal path such that \( \varphi \) is always true
  - \( A[\varphi] \): for all paths holds globally (in every state) \( \varphi \)
**TCTL in UPPAAL**

\[
S ::= E<> P | A<> P | A[] P | E[] P | P - -> P
\]

\[
P ::= M.l | g_c | g_d | not P | P or P | P and P | P imply P | ( P )
\]

- Clock constraints allow formulation of *quantitative time constraints*
  - i.e. how much time elapses between two system states (not possible in CTL)
- Nesting of path formulae are not supported
  - Because of simplified Version of CTL in UPPAAL
Examples in UPPAAL

- Example that makes use of an observer
- Normally an observer is an add-on automaton in charge of detecting events without changing the observed system
- Here the clock reset (x:=0) is delegated to the observer for illustration purposes.

(a) Test.  (b) Observer.
Properties for the Observer Example

- All resets of $x$ will happen when $x$ is above 2: $A[] \text{Obs.taken} \implies x \geq 2$ (Obs is instance of Observer)
- It is possible to reach a state where the observer is in the location idle and $x$ is bigger than 3: $E<> \text{Obs.idle} \text{ and } x > 3$
- For more examples: see UPPAAL-tutorial
Examples in UPPAAL

- Client server protocol
- Client sends requests, awaits acknowledge from server
  - Acks have to arrive within 5 time units
  - Client might receive ack already after 3 time units

Is the protocol correct?
Specification for the client server protocol

- Formulate specification as TCTL-formulae in Uppaal
- Ack-arrival within 5 time units:

  \[ \text{client.reqSent} \rightarrow (\text{client.ackReceived} \text{ and } \text{client.x} \leq 5) \]

  "Whenever a request has been sent indicated by the active state reqSent the client switches into state ackReceived by receiving an acknowledge within 5 time units."

- Ack after 3 time units:

  \[ E<> (\text{client.ackReceived} \text{ and } \text{client.x} == 3) \]

→ Use Uppaal for verification