Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 12: Timed Models

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)

Outline

- Timed Processes
  - Timing-based Light Switch
  - Buffer with a Bounded Delay
  - Multiple Clocks
  - Formal Model
  - Timed Process Composition
- Timing-based Protocols
  - Timing-based Distributed Coordination

- Next lecture: Hybrid systems
- Questions

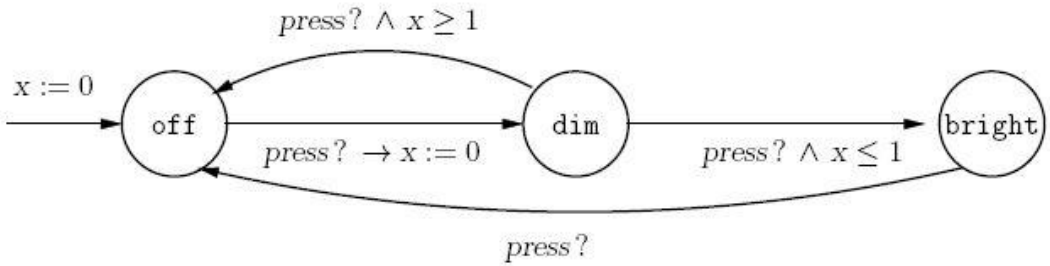
Timed Models

- *synchronous* model of computation
  - all components execute in lock-step
  - the production of outputs by a component is synchronized with the reception of inputs
- *asynchronous* model of computation
  - all processes execute at independent speeds
  - there is an unspecified delay between the reception of inputs and the production of outputs by a process
- a ***timed*** model of computation
  - processes are not synchronized, but rely on the global physical time to achieve a loose form of synchronization
  - to express phenomena such as
    - "execute the task corresponding to sensing of temperature every 5ms,"
    - "delay between the reception of an input value and the corresponding output response is between 2ms to 4ms,"
    - "if an acknowledgment is not received within 4ms, resend."

Timed Models
- Timed Processes

- formal model of computation for timed processes is a variation of the model of asynchronous processes

- a light switch - a single push-button, along with a built-in timer, to control a light bulb with two intensity levels. The switch is initially off.
  - when it is pressed once, it turns on the light at a low intensity.
  - if it is pressed twice in rapid succession, the light is turned on at a bright intensity. Here, “rapid" means that the duration between the successive press events is less than one second. If the delay between the successive press events is more than 1s, the second press event is interpreted as a command to switch the light off.

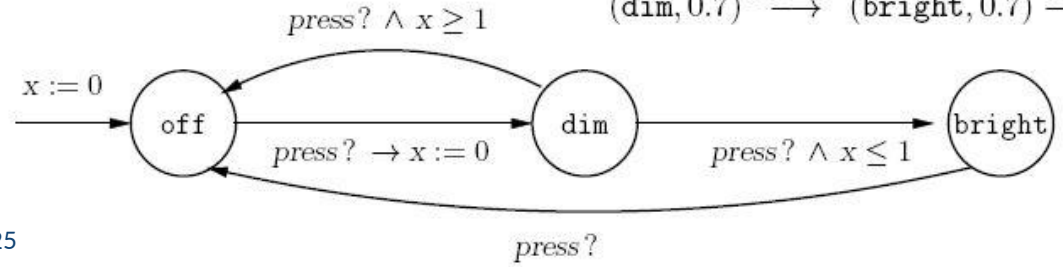


timed process LightSwitch

Timed Models
- Timed Processes (cont)

- A timed process has input, internal, and output actions just like an asynchronous process
  - an input action processes an input value received on an input channel,
  - an output action produces an output value on an output channel,
  - an internal action only updates state variables and does not involve any input or output channels (During each such action, clock variables are tested and updated in the same way as other state variables).
- a timed process also has ***timed actions*** that capture elapse of time.
  - During a timed action of duration, which may be any positive real number, the value of each clock variable is incremented by *delta*, and other state variables (including the mode) stay unchanged.
- One possible execution of the process is shown below, where each state is specified by listing the mode and the value of the clock variable x:

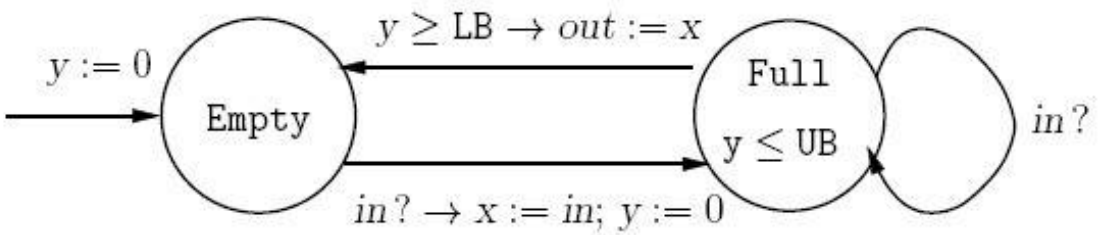
$$(\texttt{off},0) \xrightarrow{2.3} (\texttt{off},2.3) \xrightarrow{press?} (\texttt{dim},0) \xrightarrow{0.2} (\texttt{dim},0.2) \xrightarrow{0.5}$$
$$(\texttt{dim},0.7) \xrightarrow{press?} (\texttt{bright},0.7) \xrightarrow{3.0} (\texttt{bright},3.7) \xrightarrow{press?} (\texttt{off},3.7)$$



timed process LightSwitch

Timed Models
- Timed Processes
○Buffer with a Bounded Delay

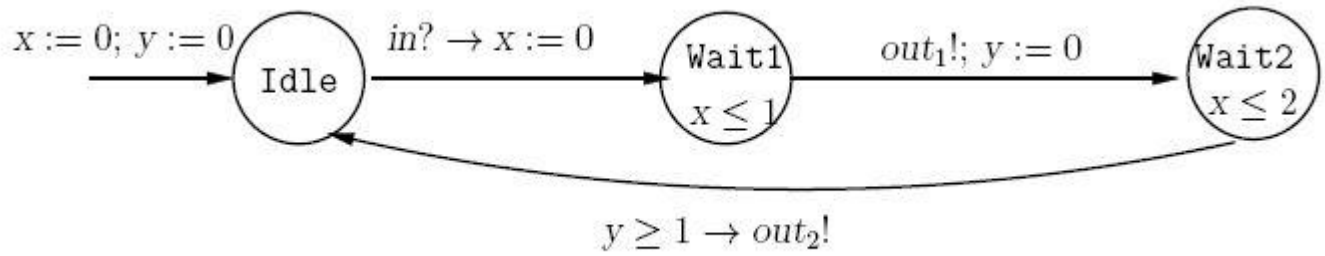
- a timed buffer of capacity 1 with input channel in and output channel out, of type msg.
  - •Whenever an input value v is supplied to the buffer, it is stored in the internal state variable x. Now the buffer becomes full, and it simply ignores (or loses) further inputs until it gets a chance to output the stored value on the output channel.
  - •The timing assumption is that the delay between the reception the input value and the transmission of the corresponding output value, is at **least LB and at most UB time units,** where the parameters LB and UB capture the lower and upper bound, respectively, on the delay. This form of lower and upper bounds on delays is a commonly occurring pattern for timed systems.



timed buffer with a bounded delay - TimedBuf-

Timed Models
- Timed Processes
○ Multiple Clocks

- a timed process that employs two clocks, consider the process shown in the next figure with input channel in and output channels out1 and out2.
  - If an input event on the channel in happens at time t, the process responds by producing an output event on the channel out1 at time t1 followed by an output event on the channel out2 at time t2 such that
    - (1) the delay (t1 –t) is at most 1
    - (2) the delay (t2-t) is at most 2
    - (3) the delay (t2-t1) is at least 1.
  - Thus, the output event on the channel out2 is constrained to occur within an interval that depends on the timing of the preceding input event on the channel in as well as the output event on the channel out1. The process does not accept inputs on channel in until it has issued both the output events.



A timed process with two clocks

Timed Models
- Timed Processes

- Formal Model

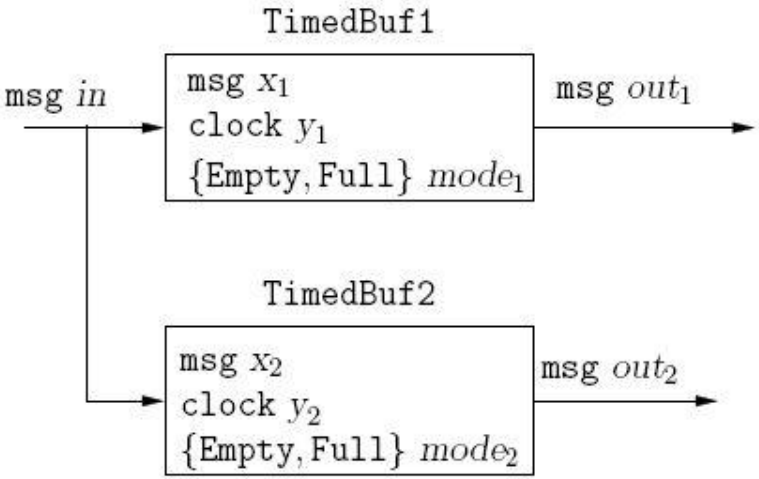
Timed Process

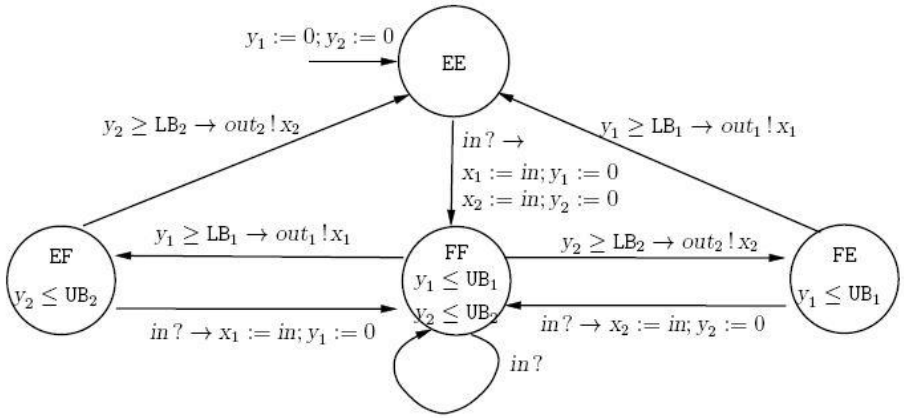
A *timed process* TP consists of (1) an asynchronous process P with some of the state variables of type `clock`; and (2) a *clock invariant* CI which is a Boolean expression over the state variables S. Inputs, outputs, states, initial states, internal actions, input actions, and output actions of the timed process TP are the same as that of the asynchronous process P. Given a state s and a real-valued time $\delta > 0$, $s \xrightarrow{\delta} s + \delta$ is a *timed action* of TP if the state $s + t$ satisfies the expression CI for all values $0 \leq t \leq \delta$.

Timed Models
- Timed Processes
o Timed Process Composition

TIMED PROCESS COMPOSITION

For two timed processes $TP_1 = (P_1, CI_1)$ and $TP_2 = (P_2, CI_2)$ such that the output channels of the two processes are disjoint, the *parallel composition* $TP_1 \mid TP_2$ is the timed process whose asynchronous process is $P_1 \mid P_2$ and whose clock invariant is $CI_1 \wedge CI_2$.





- Composition of two instances of TimedBuf processes and the State machine for composition

Collaborative learning Timed Process Composition

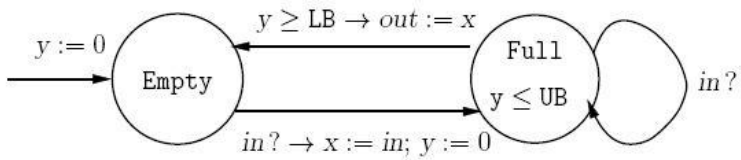
Consider:

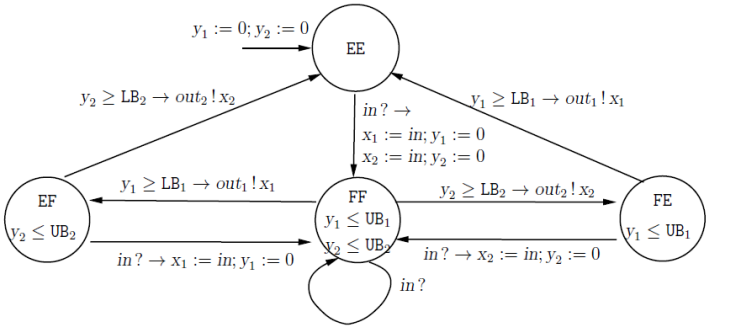
1. Composition of two instances of TimedBuf processes
2. State machine for composition

Lecture 12 – in class
(in the room or online)

Task in class

- Group = 3 members (25XP activity +25XP questionnaire)
- Time:
  - 30 minutes work
  - 10 minutes – debriefing
  - 10 minutes - Questionnaire
- Create a poster (on paper or online canva.com or other) with information about the following questions:
  - Q1: What is the invariant of FF? Describe in natural language the meaning of it.
  - Q2: Which are the 2 possible actions from mode FE and under what conditions? State the actions and the condition for each of them.
  - Q3. What is the condition such that first component produces the first output before the second component produces its output (starting from EE)? Justify your answer.
- Poster:
  - Response to the Q1,Q2,Q3 questions
  - Add also the TimeBuf process and the State machine for composition:
- Add on assignment - Bonus_Lecture_12_CollaborativeLearning
  - Created Poster with answers + Team members
  - Questionnaire + Informed agreement signed
  - 19 December 2025 – h:20:00 (only students attending the lecture)





Timed Models
- Timing-based Protocols

- In the asynchronous model of computation – studied how to solve problems that require coordination among distributed processes, that is, without making any assumptions about the relative speeds of the participating processes.

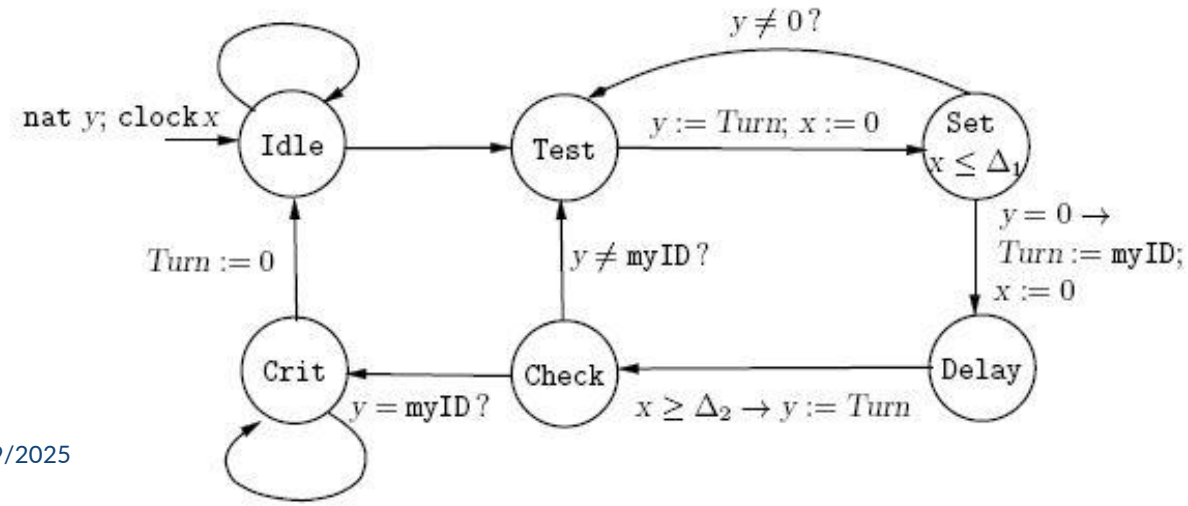
In the timed model of computation – how to solve? (where timing assumptions about the relative speeds of concurrent processes can be relied upon to design solutions.)

- Below we describe a timing-based solution to the classical coordination problem of mutual exclusion, and the same ideas can be used to solve consensus.

Timed Models
- Timing-based Protocols
Timing-based Distributed Coordination

• ***the mutual exclusion problem*** - allows asynchronous processes to access to a critical shared resource.

- The allocation of the resource is not governed by a central coordinator, but processes need to coordinate among themselves using atomic registers.
- The safety requirement is mutual exclusion: no two processes should be in the critical section simultaneously.
- The liveness requirement is deadlock freedom: if some process wants to enter the critical section, then some process should be allowed to

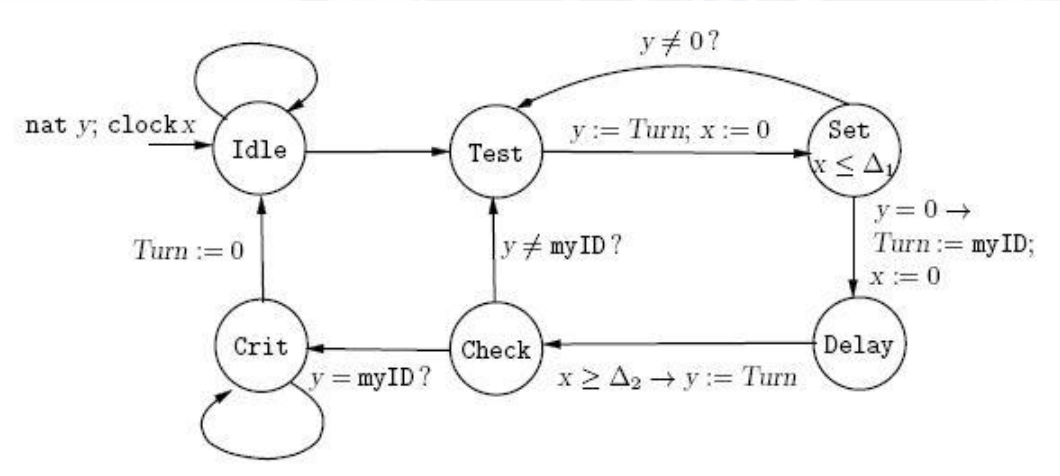


Timing-based Mutual Exclusion

Timed Models
- Timing-based Protocols

Timing-based Distributed Coordination (cont)

• Initially, Turn is 0 and the mode is Idle. There is no clock invariant associated with the initial mode Idle, and the mode has a self-loop. Thus, the process may spend an arbitrary amount of time in this mode.
• When the process wants to enter the critical section, it switches to the mode Test. Then, it reads the shared register Turn into its local variable y, and switches to the location Set.
• It then tests whether the read value is 0 or not. If it finds the value to be 0, it updates the value of the shared register Turn to its own identifier, and switches to the mode Delay; and otherwise, it switches back to the mode Test in order to read the shared register Turn again.



Timing-based Mutual Exclusion

References

• **Principles of Embedded Computation,** Rajeev Alur
http://www.seas.upenn.edu/~cis540/

CMES – Today

Bring it All Together

Timed Models

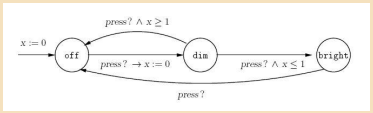
- a ***timed*** model of computation

processes are not synchronized, but rely on the global physical time to achieve a loose form of synchronization

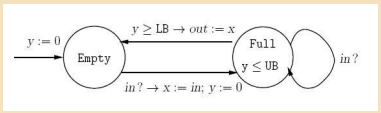
- to express phenomena such as
  - "execute the task corresponding to sensing of temperature every 5ms,"
  - "delay between the reception of an input value and the corresponding output response is between 2ms to 4ms,"
  - "if an acknowledgment is not received within 4ms, resend."

Timed models - examples

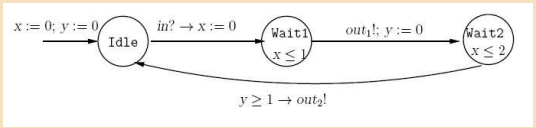
Timing-based Light Switch



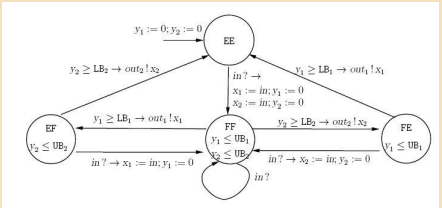
Buffer with a Bounded Delay



Multiple Clocks



Timed Process Composition



Thank You For Your Attention!

- ExitTicket
- Mentimeter
  - menti.com
  - Code:

Next Lecture

- Hybrid systems

Happy Christmas and Happy and Healthy New Year!

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)