CS:4980
Foundations of Embedded Systems

Timed Model
Part I

Models of Reactive Computation

Synchronous model

- Components execute in a sequence of discrete rounds in lock-step
- Computation within a round: execute all tasks in an order consistent with task precedence constraints

Asynchronous model

- Speeds at which different components execute are independent
- Computation within a step: execute a single task that is enabled

Timed model (partially synchronous)

- Like asynchronous for communication of information
- But can rely on global time for coordination

Continuous-time model (for dynamical system)

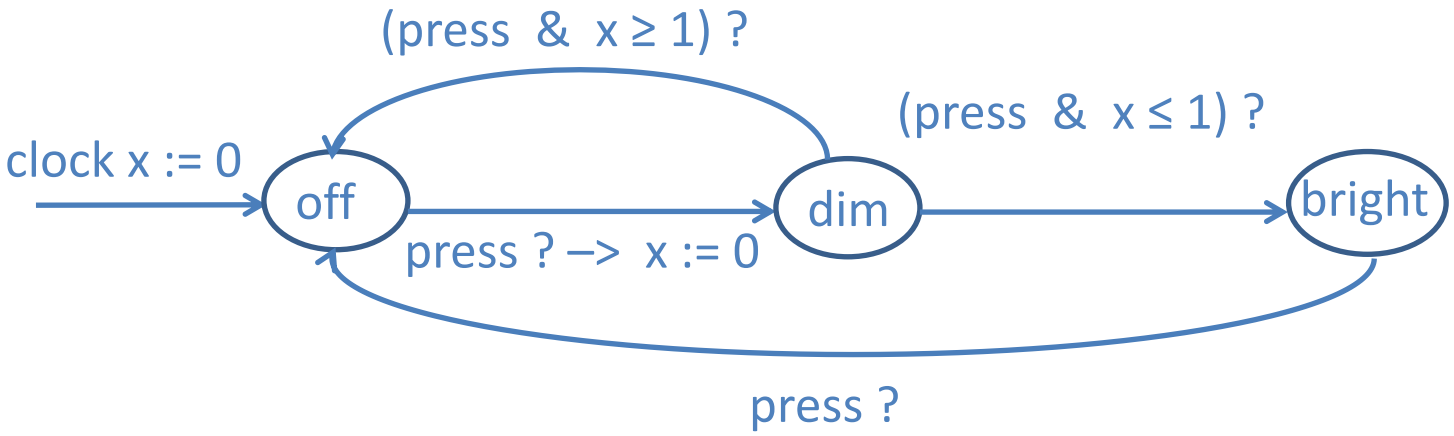
- Synchronous, but now time evolves continuously
- Execution of system: solution to differential equations

Timed Model

The timed model allows us to express phenomena such as:

- Task A is is executed every 5ms
- The 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 5ms, the message is resent

Timed Model Example



Initial state: (mode = off, x = 0)

Timed transition: (off, 0) –0.5–> (off, 0.5)

Input transition: (off, 0.5) –press?–> (dim, 0)

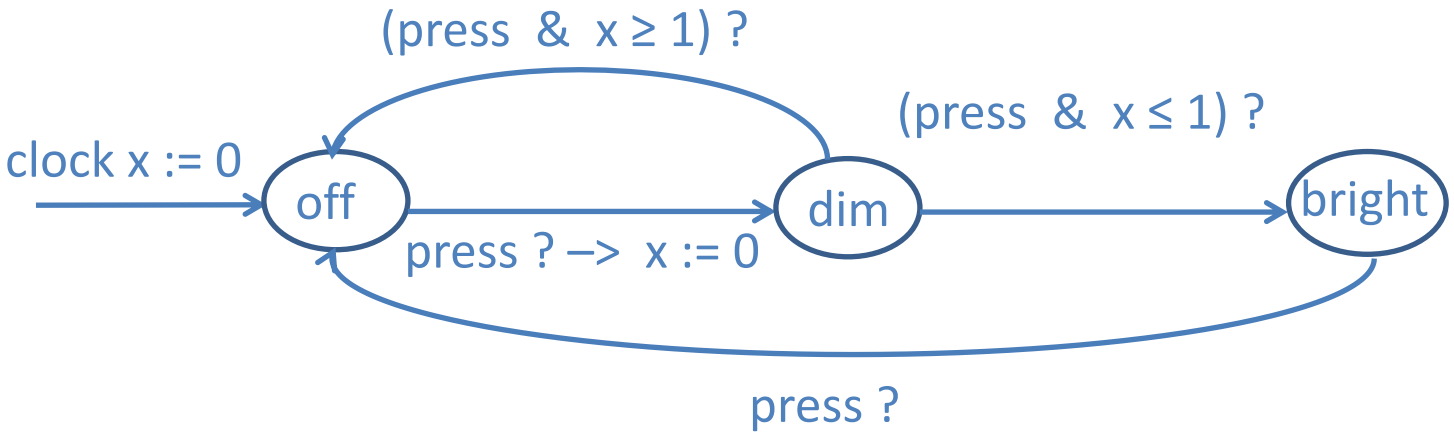
Timed transition: (dim, 0) –0.8–> (dim, 0.8)

Input transition: (dim, 0.8) –press?–> (bright, 0.8)

Timed transition: (dim, 0.8) –1–> (dim, 1.8)

Input transition: (dim, 1.8) –press?–> (off, 1.8)

Timed Model Example



Clock variables

- Tests and updates in mode-switches like other variables
- New: During a timed transition of duration d, the value of clock variables increases by an amount equal to d

Timing constraints

Update x := 0 in off –> dim and guard x ≤ 1 in dim –> bright indicates that timing of these two transitions is ≤ 1 apart

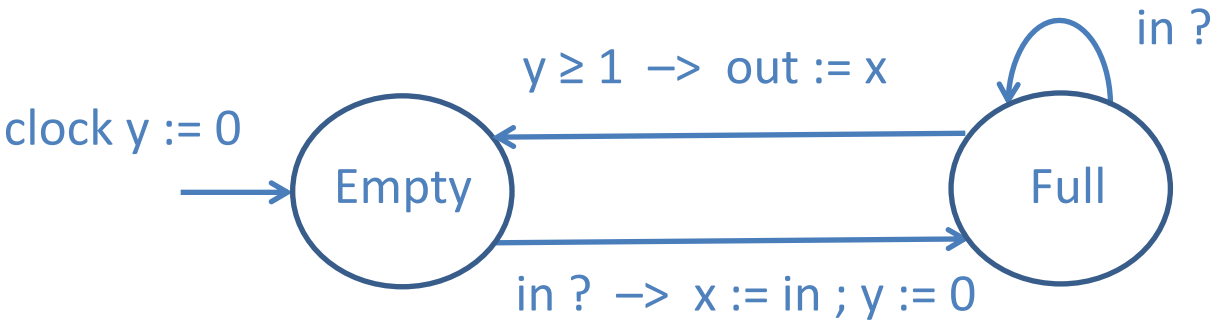
Timed Buffer Example

Buffer with a bounded delay

Behavior:

Input received on channel in is transmitted on output channel out after a delay of d time units, with LB ≤ d ≤ UB
(i.e. the delay has known lower and upper bounds)

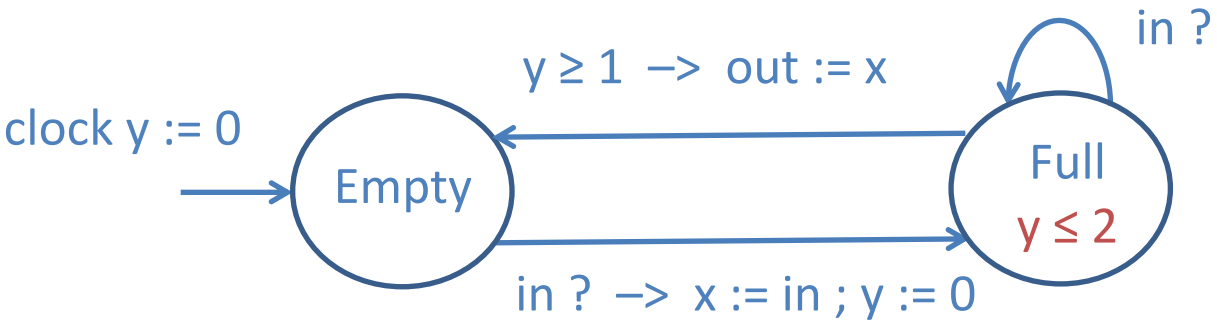
Modeling Timed Buffer



- Input channel: msg in Output channel: msg out
- Mode indicates whether the buffer is full or not
- State variable x remembers the last input value when buffer is full
- Clock variable y tracks the time elapsed since buffer filled up
- When buffer is full, input events are ignored
- Guard y ≥ 1 ensures that at least 1 time unit elapses in mode Full

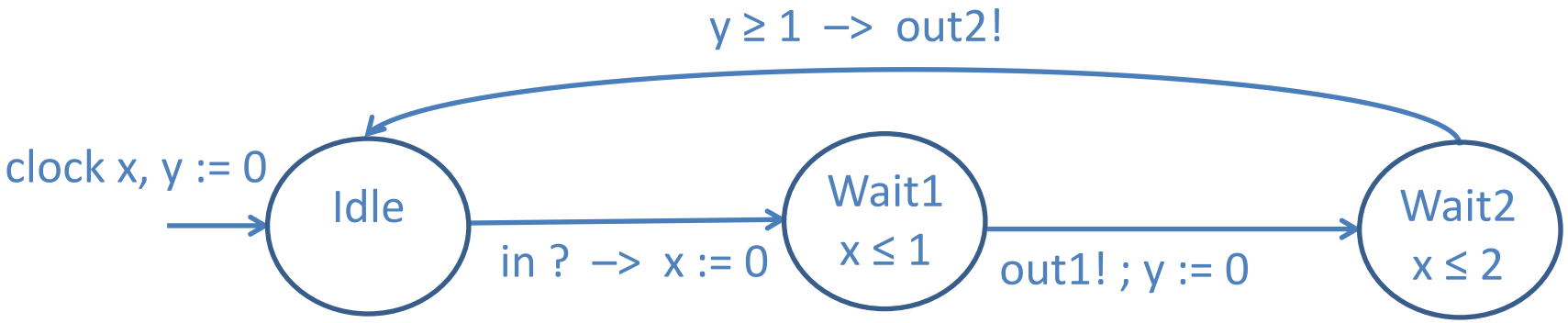
How to ensure that mode-switch from Full to Empty is executed before clock y exceeds the upper bound 2?

Clock Invariants



- The constraint y ≤ 2 associated with mode Full is a *clock invariant*
- A timed transition of duration d is allowed only if the clock invariant is satisfied for the entire duration of the transition
  - (Full, x, 0.8) –0.7–> (Full, x, 1.5) allowed
  - (Full, x, 0.8) –1.4–> (Full, x, 2.2) disallowed
- Clock invariants limit how long a process stays in a mode

Example with Two Clocks

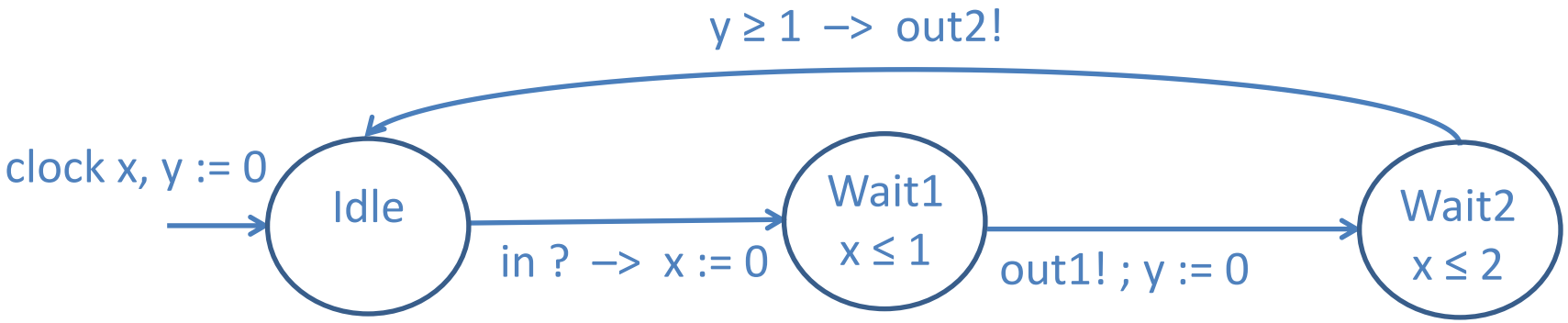


- Input channel: event in Output channels: event out1, out2
- Two clock variables: x, y
- As time passes, both clocks increase (and at the same rate)

Sample timed transitions from state (mode, x, y) = (Wait2, 0.8, 0) :

(Wait2, 0.8, 0) –0.3–> (Wait2, 1.1, 0.3) –0.72–> (Wait2, 1.82, 1.02)

Two Clock Example



- Clock x tracks time elapsed since the last input event
- Clock y tracks time elapsed since the output event

What is the behavior of this model?

Suppose an input event occurs at time t,

- the process produces an output event on channel out1 at time $t' \in [t, t+1]$
- then on channel out2 at time $t'' \in [t'+1, t+2]$

Example Specification

Consider a timed process with

Input: event x **Output:** event y, event z

Desired behavior:

- For each input event, produce both output events
- Time delay between x? and y! is in interval [2, 4]
- Time delay between x? and z! is in interval [3, 5]
- Ignore later inputs received in these intervals

Definition of Timed Process

Definition: A *timed process* TP consists of

1. An asynchronous process P, where some of the state variables can be of type *clock* (non-negative reals)
2. A *clock invariant* CI, a Boolean expression over P's state variables

Note: Inputs, outputs, states, initial states, internal actions, input actions, and output actions are exactly as in the asynchronous model

Definition of Timed Process

Notation: For a state s and time t, let $s+t$ denote the state such that

- $(s+t)(x) = s(x) + t$ for every clock variable x
- $(s+t)(y) = s(y)$ for every non-clock variable y

Definition: Given a state s and a time $d > 0$, *timed action* $s \text{ –d–> } s+d$ is a transition of duration d where state $s+t$ satisfies invariant CI for all $t \in [0, d]$

Note: If CI is a convex constraint, it suffices that s and $s+d$ satisfy CI

Definition of Parallel Composition

Let $TP_1 = (P_1, CI_1)$ and $TP_2 = (P_2, CI_2)$ be timed processes

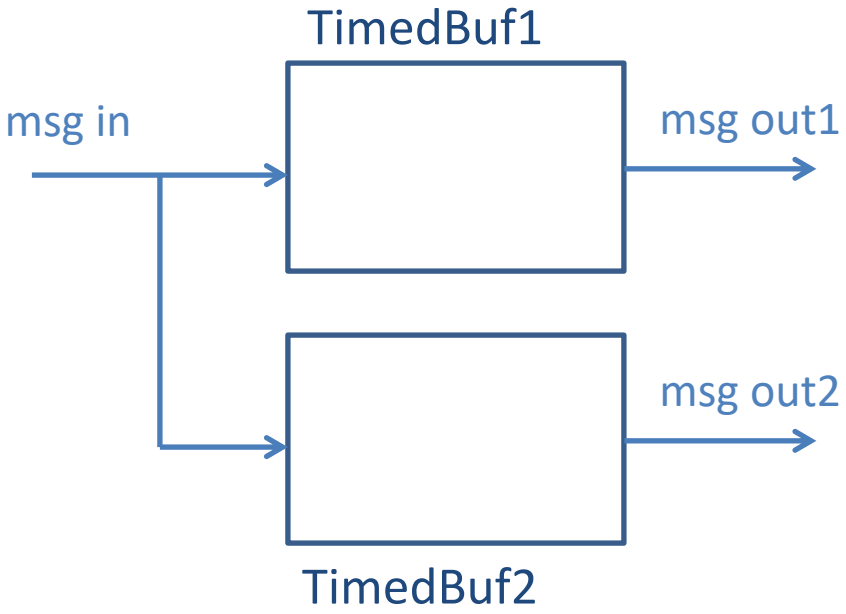
The parallel composition $TP_1 \mid TP_2$ is defined iff $P_1 \mid P_2$ is defined (that is, iff the outputs of P_1 and P_2 are disjoint)

Composition: $TP_1 \mid TP_2 = (P_1 \mid P_2, CI_1 \wedge CI_2)$

- Asynchronous composition of P_1 and P_2 defines the internal, input and output actions of $TP_1 \mid TP_2$
- Conjunction of CI_1 and CI_2 defines the clock-invariant of $TP_1 \mid TP_2$

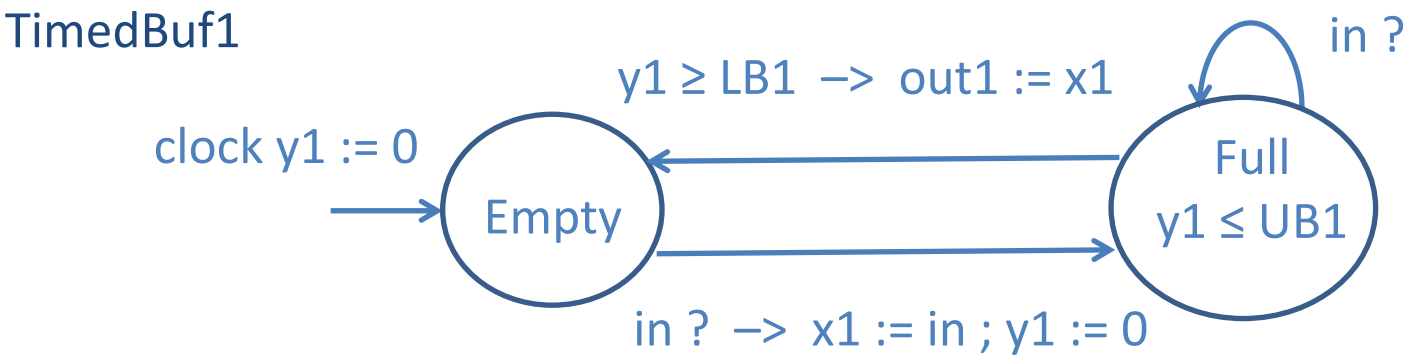
Consequence: The composite process allows a timed action of duration d exactly when both TP_1 and TP_2 can wait for time d

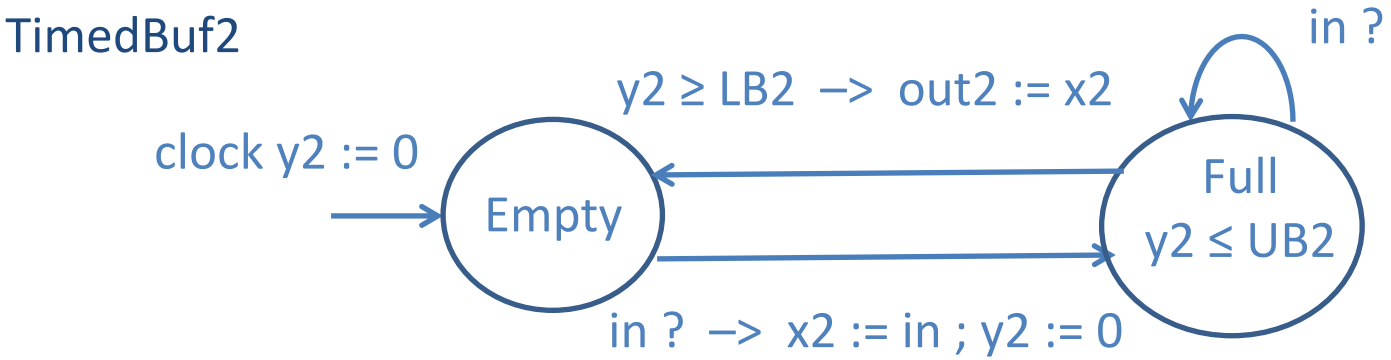
Composition of Processes



- How to construct timed process corresponding to the composition of the two processes?
- What are the possible behaviors of the composite process?

Composition of Timed Processes

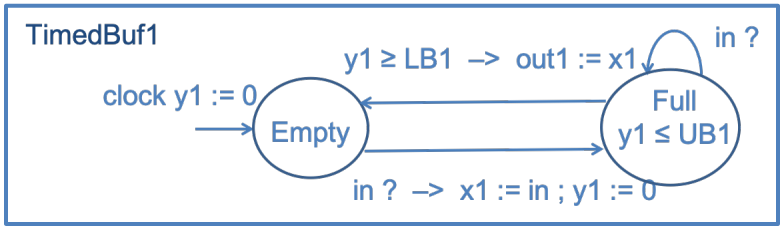


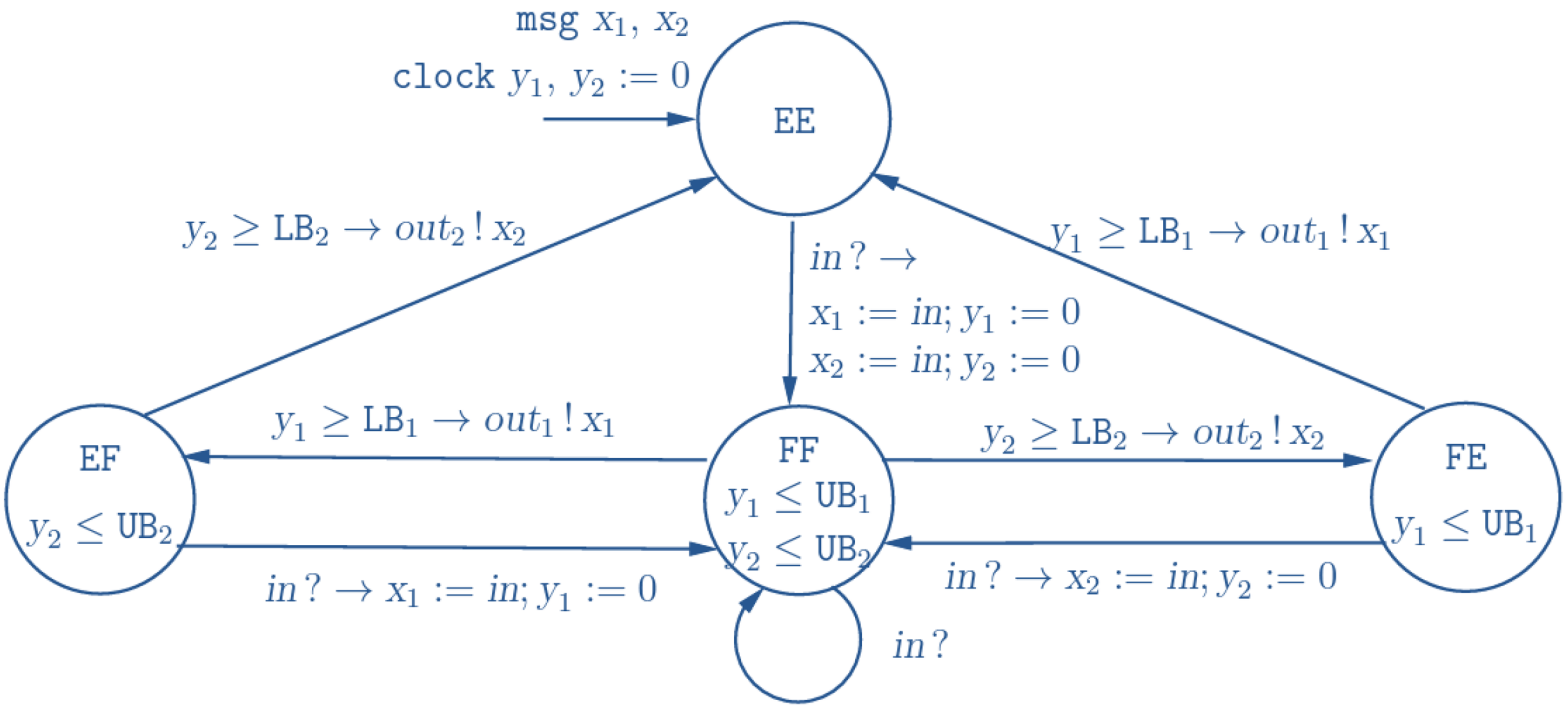


The composite process has four modes:

(Empty, Empty), (Empty, Full), (Full, Empty), (Full, Full)

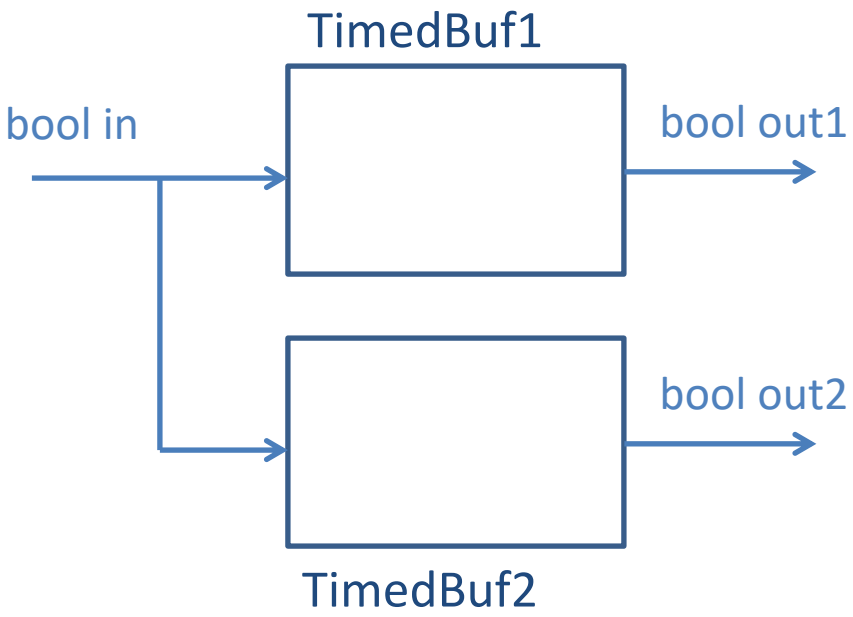
Composition of Timed Processes





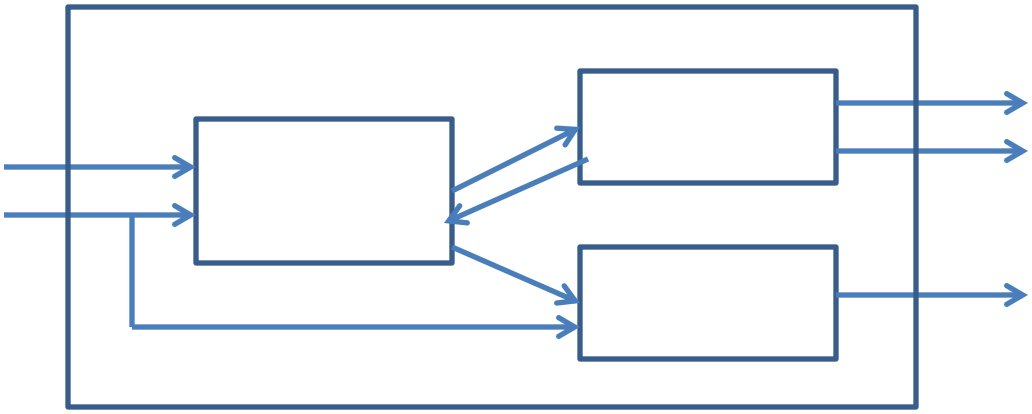
$$CI = (\text{mode} = EF \Rightarrow y_2 \leq UB_2) \wedge (\text{mode} = FF \Rightarrow y_1 \leq UB_1 \wedge y_2 \leq UB_2) \wedge (\text{mode} = FE \Rightarrow y_1 \leq UB_1)$$

Composition of Processes



- If UB1 < LB2 then out1 is guaranteed to occur before out2
  - Implicit coordination based on bounds on delays
- Is it possible to observe two out1 events without an intervening out2 event?
  - Depends on relative magnitudes of bounds (need timing analysis!)

Block Diagrams



Components can be timed processes now

Operations: instantiation (I/O variable renaming), parallel composition, and variable hiding

Composite system step:

1. an internal step of one of the components,
2. a communication (I/O) step involving relevant sender and receivers, or
3. a timed step involving all the components

Credits

Notes based on Chapter 7 of

Principles of Cyber-Physical Systems
by Rajeev Alur
MIT Press, 2015