Embedded System Design and Modeling

III. Extended FSMs and Timed Automata

FSM example

▶ Recall the previous FSM example
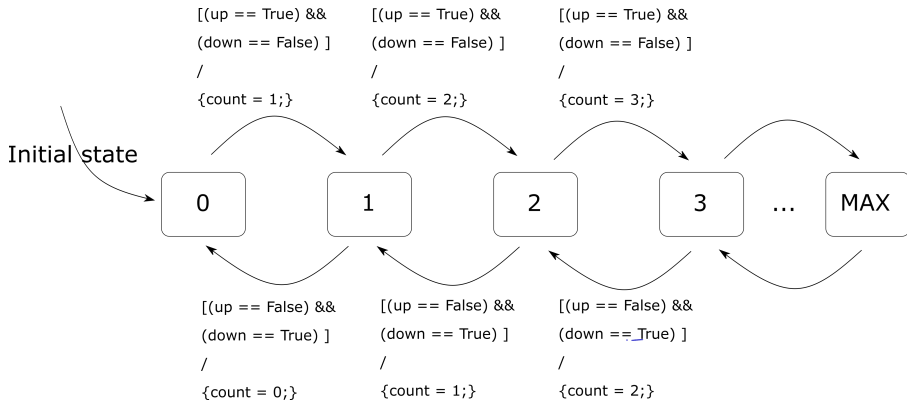

Figure 1: Parking system FSM

▶ Can we make it is simpler to draw?

▶ **Extended FSM** = FSM with **internal variables**

Inputs:
  up: bool
  down: bool
Outputs:
  count: integer (0, MAX)
Variables:
  count: integer (0, MAX)

States = 10 ↑ (handwritten)

[(up==False) && (down==True) && (count > 0)] /

{count = count - 1;}

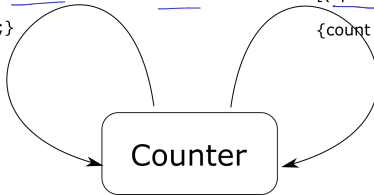[(up==True) && (down==False) && (count < MAX)] /

{count = count + 1;}

≅ 100 (handwritten)

Counter

Figure 2: Extended FSM with variable "count"

Extended FSM

- The <u>state of the model</u> = the current "bubble" and the values of **all the internal variables**

- Example: OS hibernation in Windows:
    - state of computer = all the RAM memory values
    - if all memory is written down on HDD, and reloaded tomorrow, the system effectively resumes operation from where it left off

- State is not anymore "the number of bubbles"
    - there is only one "bubble" in our FSM
    - but there are MAX+1 states (all possible values of the count variable)

- Always make explicit declaration of:
    - model inputs
    - model outputs
    - model internal variables
    - and their data types
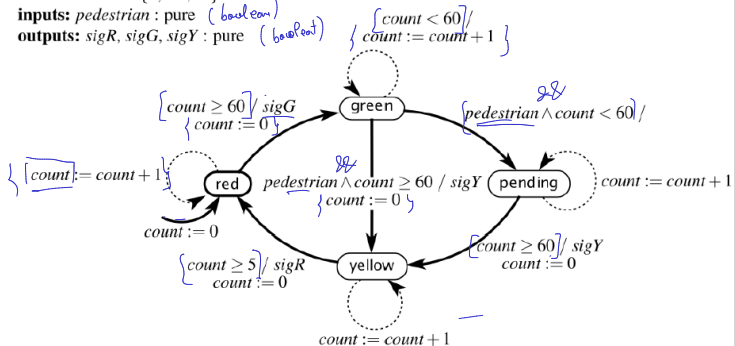
- Extended FSM are useful for modeling **time-based** conditions:
    - measure passage of time: increment a variable every *tick*
    - only works if the FSM is time-triggered

Example: pedestrian crossing light

- How is time measured in the model below?
- How many states does the model below have?

This model assumes one reaction per second
(a *time-triggered* model)

Figure 3: Extended FSM with time measuring [1]

---

[1] image from Seshia' slides

Figure 4: Hybrid systems [2]

---
[2]image from Seshia' slides

Hybrid systems
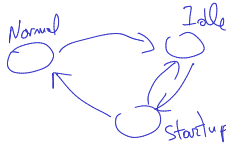
- **Hybrid systems** = system with mixes discrete and continuous behavior

- Example: a PID controller with different modes:
    - a set of distinct functioning model (e.g. Startup / Normal / Idle)
    - each state is a sub-system implemented with continuous dynamics

- State **refinement** = a lower-level implementation of a state

Types of hybrid systems

Automate temporale

▶ **Timed automata** = hybrid system where every state refinement just measures passage of time (differential equation of degree 1)

▶ **Higher-order systems** = hybrid system where every state refinement uses higher-order differential equation (2 or more)

▶ **Two-level control systems** = complex controllers with two levels of operation

  ▶ high-level discrete modes of operation (e.g. ECU Power Modes: Normal / Startup / Sleep Mode 1 / Sleep Mode 2)
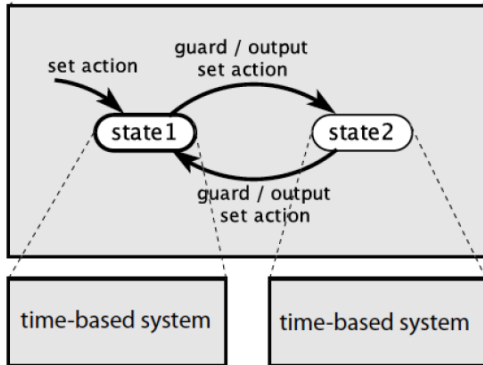 ▶ low-level refinements with continuous dynamics

Figure 5: Timed automaton example [3]

Example

continuous variable: $x(t) \in \mathbb{R}$
inputs: $click \in \{present, absent\}$
outputs: $single, double \in \{present, absent\}$

$x'(t) = 0$ f const

$x'(t) = 1 \Rightarrow$

x

$x(t)$

idle
$\dot{x}(t) = 0$

$[click]$
$x(t) := 0$

one
$\dot{x}(t) = 1$

$x(t) := 0$
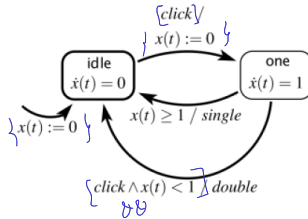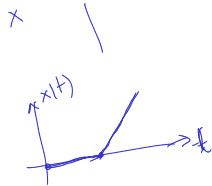
$x(t) \geq 1 / single$

$[click \wedge x(t) < 1] double$

Figure 6: Mouse Double-click detector model [4]

- Here $\dot{x}(t) = 1$ means "x(t) increases linearly with time", so it measures time
- How many states does this model have? ∞ infinitate

[4]image from Seshia' slides

Example: Another Thermostat

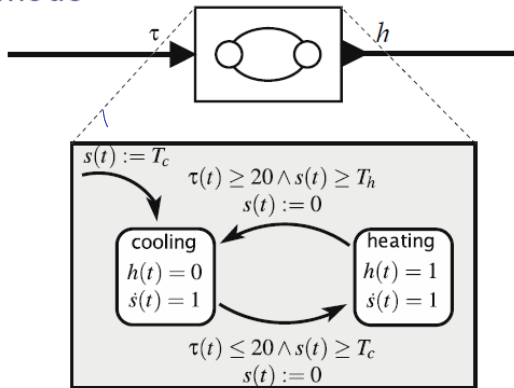Temperature threshold is 20 with minimum times T_c and T_h in each mode

Figure 7: - Another thermostat model as a Timed Automaton [5]

---

[5]image from Seshia' slides

Temperature threshold is 20 with minimum times T_c and T_h in each mode
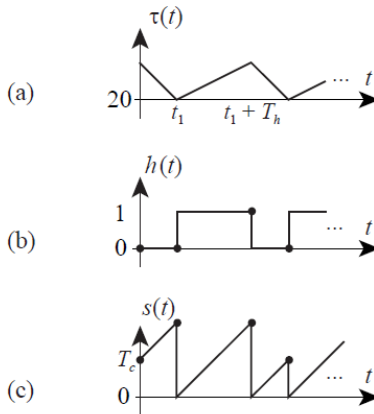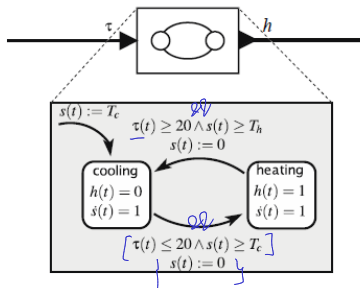
Figure 8: Input and output signals [6]

---

[6]image from Seshia' slides
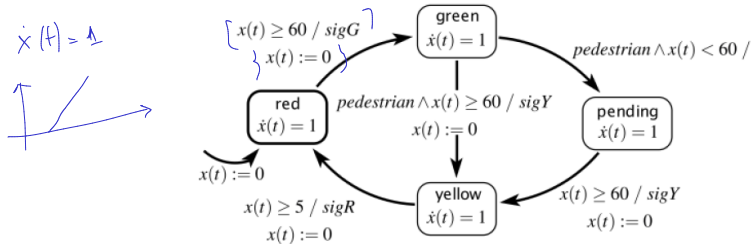
Example: Another Traffic Light

Timed automaton model of a traffic light controller

continuous variable: $x(t)$: $\mathbb{R}$
inputs: *pedestrian*: pure
outputs: *sigR, sigG, sigY*: pure

This light remains green at least 60 seconds, and then turns yellow if a pedestrian has requested a crossing. It then remains red for 60 seconds.

Figure 9: Traffic Light controller Timed Automaton [7]
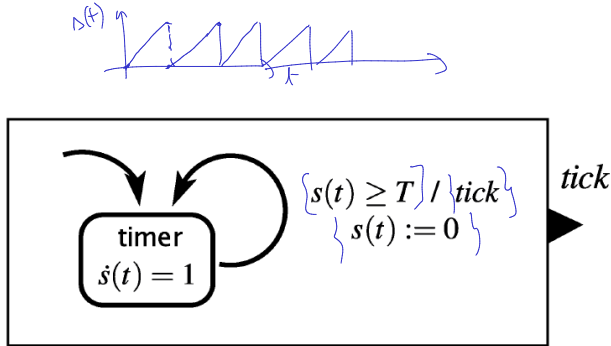
Figure 10: Timed Automaton to generate a *tick* event every T seconds [8]

---

[8]image from Seshia' slides

Example: Bouncing Ball

Hybrid Automaton for Bouncing Ball

y – vertical distance from ground (position)
a – coefficient of restitution, $0 < a < 1$
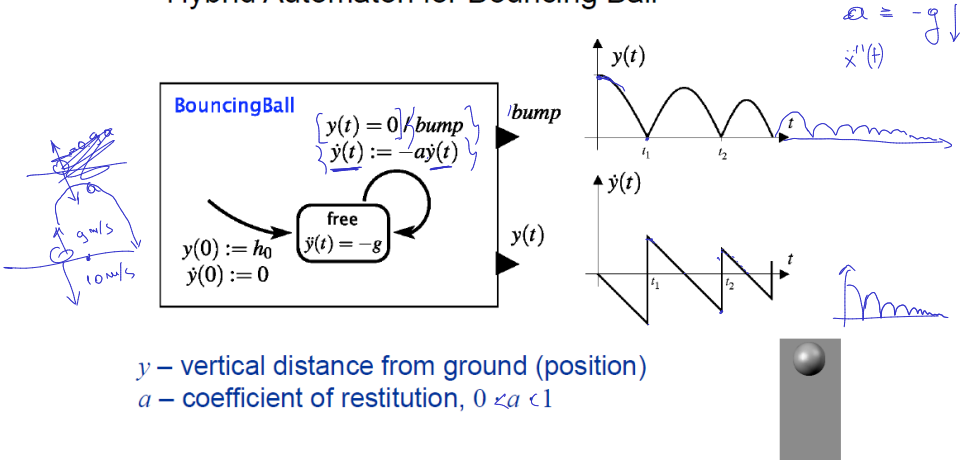
Figure 11: Timed Automaton to simulate a bouncing ball movements [9]

▶ FSM simulation software

▶ Used in this class: Stateflow (Simulink / Matlab)

▶ Features:
  ▶ State Actions
  ▶ Temporal Logic
  ▶ Other events
  ▶ . . . other . . .

State actions

- Actions can exist not only on transitions, but also **inside states**
- Three main types of **State Actions**:
    - **entry (en)**: executed only when a **state is entered**
    - **exit (ex)**: executed only when a **state is exited**
    - **during (du)**: executed when we are in state which is neither entered, not exited
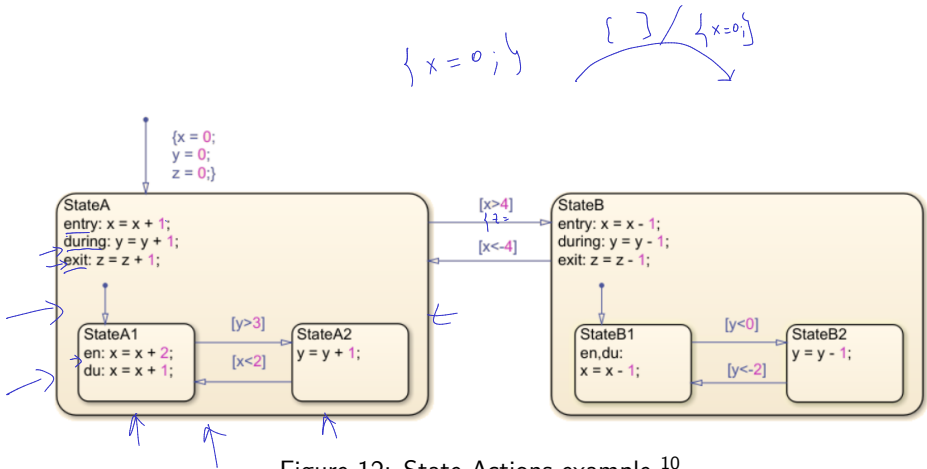
State actions

Figure 12: State Actions example [10]

---

[10]image from Matlab docs

- ▶ State actions can be avoided (use only transitions actions), but sometimes one or the other are more convenient

Temporal logic

- For time-based conditions, states certain predefined variables, which can be used to **measure time spent in a state**
  - **_tick_**: measures time steps
    - is incremented at **every time step**
    - is reset to 0 every time a state is exited or entered
    - actual duration **depends** on model step size
  - **_sec / msec_**: measures seconds or miliseconds
    - is incremented every second / milisecond
    - is reset to 0 every time a state is exited or entered
    - actual duration is **independent** on model step size

Temporal logic

- Temporal operators **after()**, **on()**, **every()** can generate events which can be used in conditions

- Examples:

  - **after(10, tick)**:
    - event is fired after 10 time steps spent in a state
    - evaluates to FALSE for the first 9 steps, is TRUE every time after that

  - **on(x, tick)**:
    - event is fired only **once**, exactly after x time steps spent in a state
    - evaluates to FALSE for the first $x - 1$ time moments, is TRUE only once at the x-th moment, is FALSE after that

  - **every(x, tick)**:
    - event is fired pariodically after x time steps
    - evaluates to FALSE for the first $x - 1$ time moments, is TRUE once at the x-th moment, then FALSE for the next $x - 1$ time moments, then TRUE again, and so on
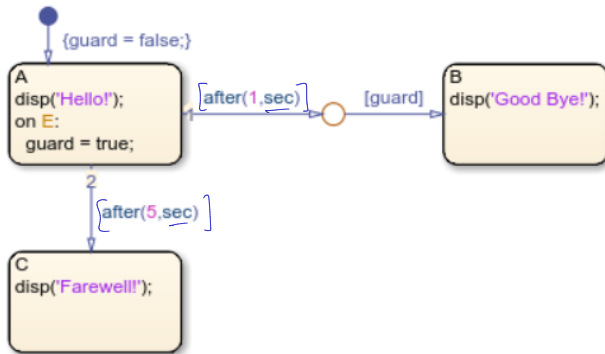
Temporal logic

Figure 13: Temporal Logic example [11]