

MEC302: Embedded Computer Systems

Theme I: Modeling Dynamic Behaviors

Lecture 3 – Discrete Dynamics: Discrete and Hybrid Systems

Dr. Timur Saifutdinov

Assistant Professor at EEE, SAT

Email:

Timur.Saifutdinov@xjtlu.edu.cn

Lecture outline

- Discrete dynamics:
 - Discrete systems;
 - State machines;
 - Finite-state machines;
 - State machine properties;
 - Extended state machines;
- Hybrid system modelling:
 - Hybrid systems;
 - Timed automata;
 - Supervisory control.

Discrete dynamics

A **discrete system** operates in a sequence of discrete steps and is said to have **discrete dynamics**.

Discrete systems operate with **discrete signals** of the form $e: \mathbb{R} \rightarrow \{absent\} \cup X$:

- It is absent most of the time and we can count, in order, the times at which it is not *absent*;
- If $X = \{present\}$, the signal carries no value and it is called a **pure signal**;
- Otherwise, the signal is not pure and X can be any set of values ($X \subseteq \mathbb{R}$).



Examples of discrete systems:

- Calculators/computers/controllers;
- Traffic lights;
- Board games;
- Etc.

Discrete systems

Consider a counter system (e.g., at a parking garage):

- **Inputs:**

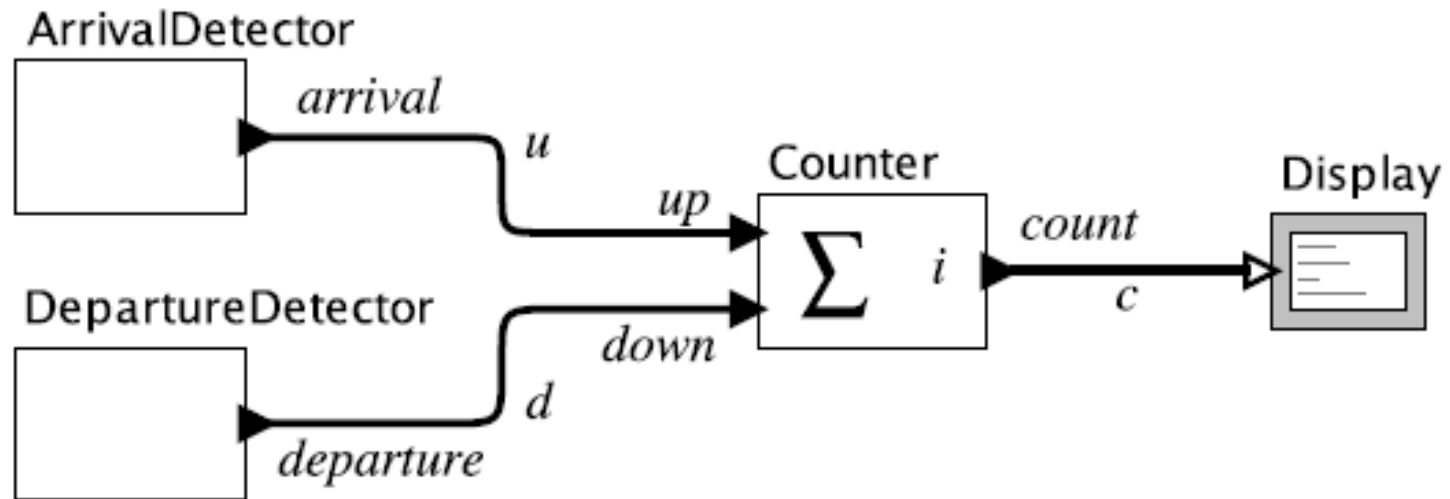
- $u: \mathbb{R} \rightarrow \{absent, present\};$
- $d: \mathbb{R} \rightarrow \{absent, present\};$

- **States*:**

- $s: \mathbb{R} \rightarrow \mathbb{Z};$

- **Output (two options):**

- $c: \mathbb{R} \rightarrow \{absent\} \cup \mathbb{Z}$ – **Mealy machine** (produces output during transition);
- $c: \mathbb{R} \rightarrow \mathbb{Z}$ – **Moore machine** (produces output when in a state).



* – Discrete models with finite state space are called finite-state machines.

George H. Mealy and Edward F. Moore are Bell Labs engineers in 1950s.

State machines

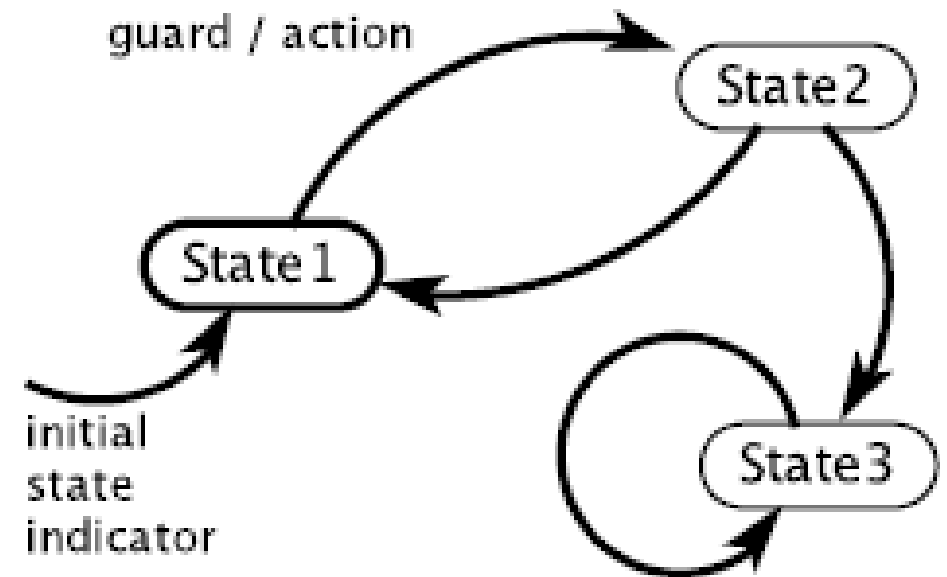
A **State Machine (SM)** is a model of a system with **discrete dynamics** that at each reaction maps valuations of the inputs to valuations of the outputs, where the map may depend on its current state.

inputs: name : type/set

outputs: name : type/set

It can be represented **graphically** with:

- *Inputs* – specified with names and types/sets;
- *Outputs* – specified with names and types/sets;
- *System states* – distinguished with balloons;
- *Transitions* – represented with arrows;
- *Predicate* (logical expression) – guards;
- *Reaction* (update) – action on a guard.



Or, it can be explicitly formulated with a Mathematical model of a **Finite-SM (FSM)**:

$FSM(States; Inputs; Outputs; update; initialState)$

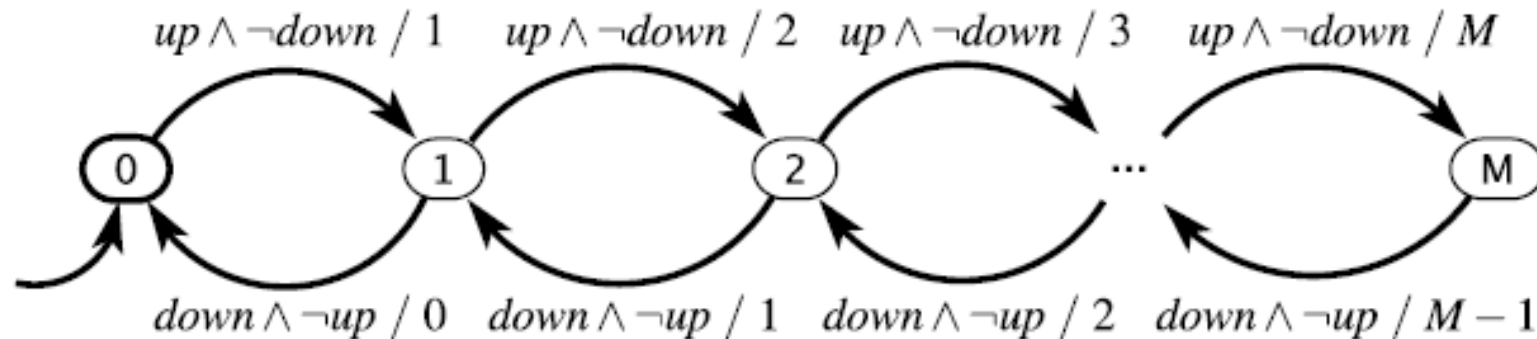
Finite-state machines

A **Finite-State Machine (FSM)** is an **SM** where the set *States* of possible states is finite.

FSM model for the garage counter (graphical representation):

inputs: *up*, *down* : pure

output: *count* : $\{0, \dots, M\}$



Predicate standard code (key):

true Transition is always enabled.

p_1 Transition is enabled if p_1 is *present*.

$\neg p_1$ Transition is enabled if p_1 is *absent*.

$p_1 \wedge p_2$ Transition is enabled if both p_1 and p_2 are *present*.

$p_1 \vee p_2$ Transition is enabled if either p_1 or p_2 is *present*.

$p_1 \wedge \neg p_2$ Transition is enabled if p_1 is *present* and p_2 is *absent*.

Finite-state machines

Mathematical model of an **FSM** is formulated with a five-tuple:

$$FSM(States; Inputs; Outputs; update; initialState) = \left(\begin{array}{l} States = \{0, 1, \dots, M\} \\ Inputs = (\{up, down\} \rightarrow \{present, absent\}) \\ Outputs = (\{count\} \rightarrow \{0, 1, \dots, M, absent\}) \\ initialState = 0 \\ update(s, i) = \begin{cases} (s + 1, s + 1) & \text{if } s < M \wedge i(up) = present \wedge i(down) = absent \\ (s - 1, s - 1) & \text{if } s > 0 \wedge i(up) = absent \wedge i(down) = present \\ (s, abs) & \text{otherwise} \end{cases} \end{array} \right)$$

State machine properties

Important properties of **State Machines (SM)**:

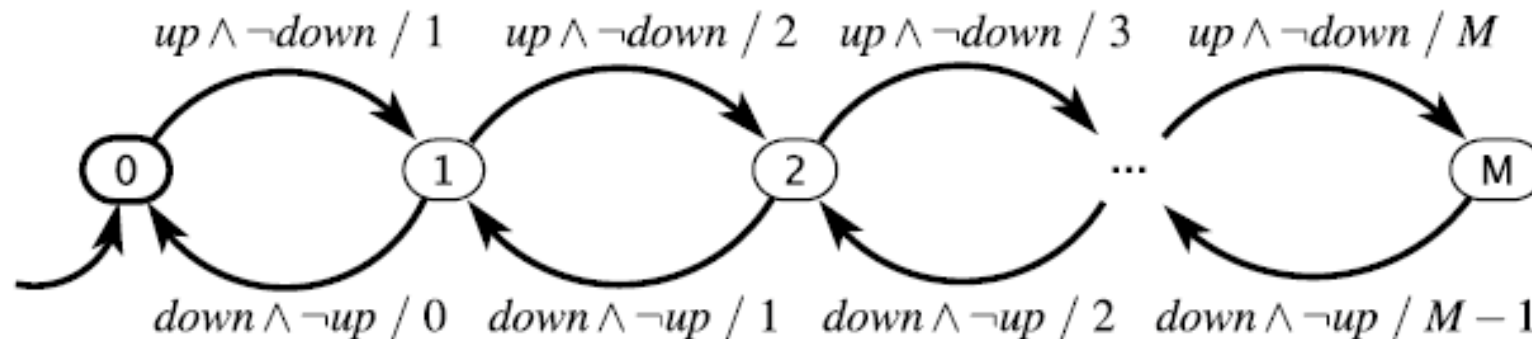
- **Determinacy** – An SM is said to be deterministic if, for each state, there is at most one transition enabled by each input value.
- **Receptiveness** – An SM is said to be receptive if, for each state, there is at least one transition possible on each input symbol.

As a result, if an SM is both deterministic and receptive, there is exactly one transition possible on each input value.

Is the FSM model of the garage counter deterministic, receptive, neither or both?

inputs: $up, down$: pure

output: $count : \{0, \dots, M\}$



Extended state machines

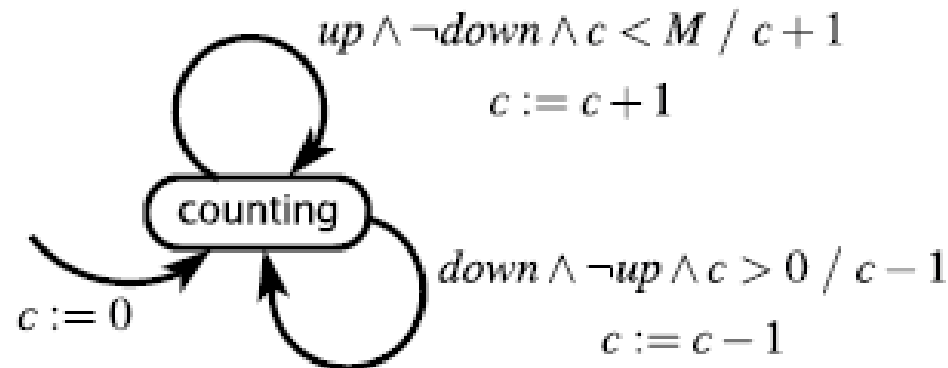
An **FSM** with large number of states can be generalized with an **Extended State Machine (ESM)**.

- Consider ESM model for garage counter:

variable: $c: \{0, \dots, M\}$

inputs: $up, down$: pure

output: $count: \{0, \dots, M\}$



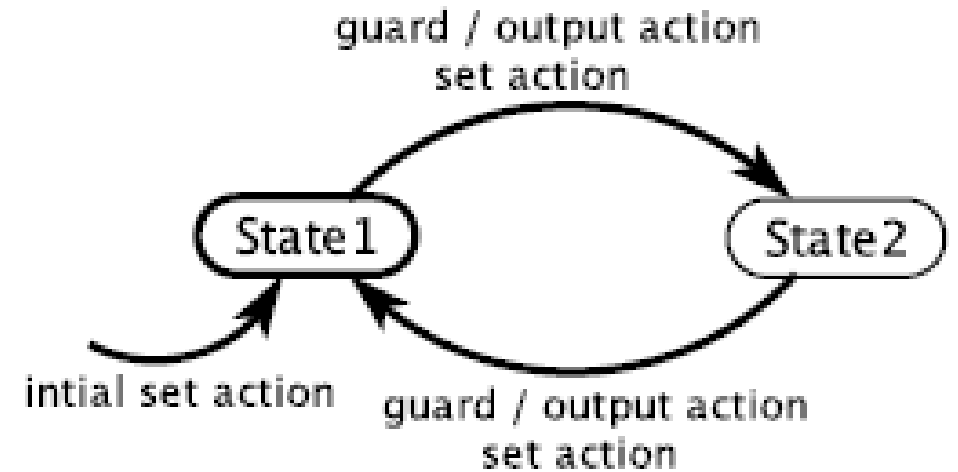
Extended state machines

The general notation for an **ESM** is as follows:

variable declaration(s)
input declaration(s)
output declaration(s)

There are three main differences:

- Variable declarations;
- Variables initialization – initial set action;
- Assignments to variables – set actions.



The state of an **ESM** is defined by both the discrete state (a “bubble”) and states of variables (ie, their values).

Further reading on SM in the textbook (sections: **3.5 Nondeterminism**; **3.6 Behaviors and Traces**):

Lee, E.A. and Seshia, S.A., 2016. Introduction to embedded systems: A cyber-physical systems approach. Mit Press.

Hybrid system modelling

Modelling techniques considered previously were:

- **Continuous dynamics** uses differential/integral equations to model continuous and “smooth” processes;
- **Discrete dynamics** uses state machines to model **discrete systems**.

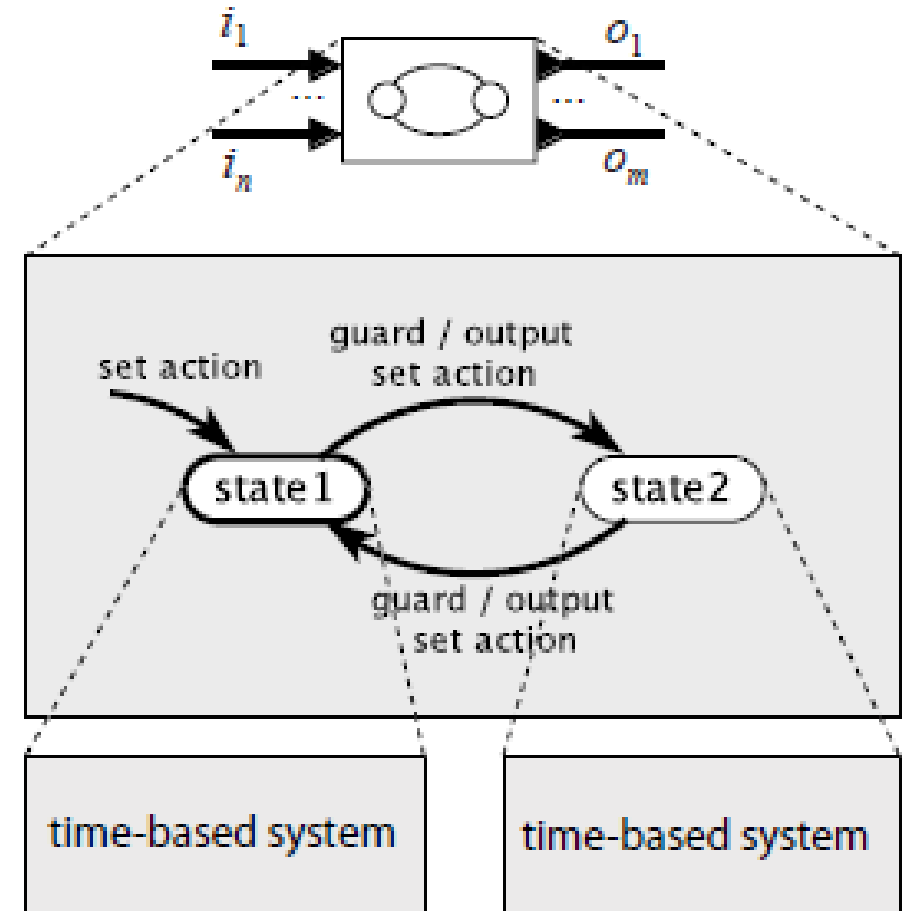
Hybrid system modelling is a formalism that incorporates both.

Hybrid systems can be very powerful to model cyber-physical systems by integrating their continuous physical dynamics with discrete computational systems.

Hybrid systems

Discrete **state machines** can be extended to **hybrid systems** by allowing any of the following:

- Continuous inputs (i.e., continuous-time signals);
- Continuous dynamics of the output(s);
- State refinement (i.e., continuous state variables).



Timed automata

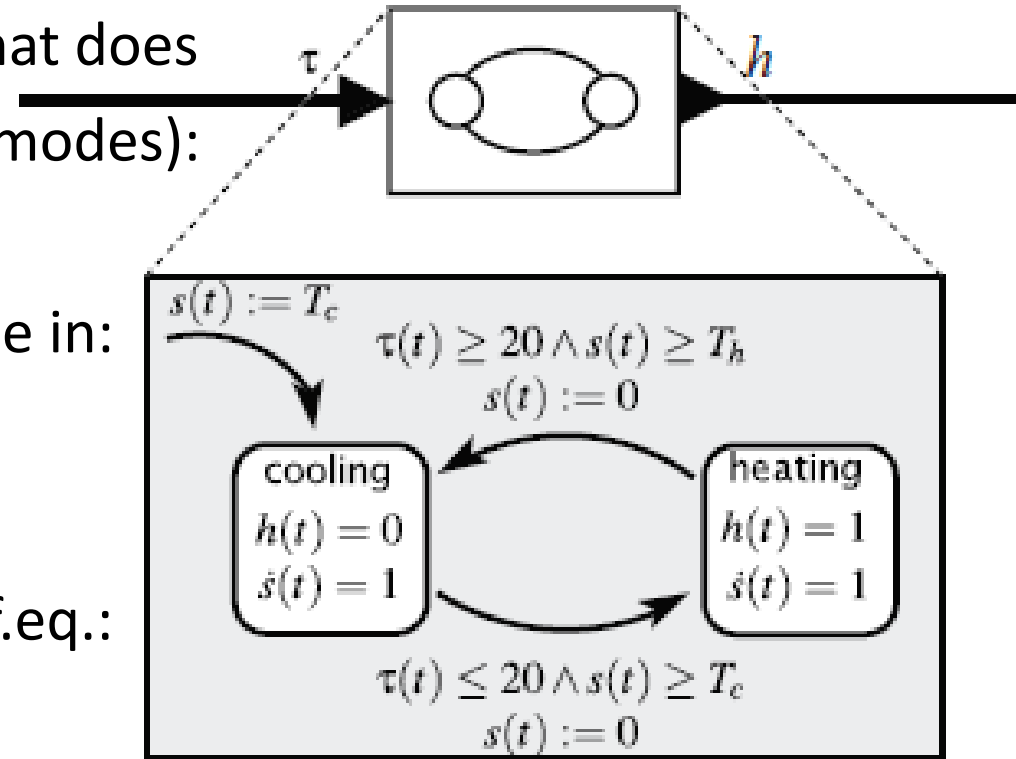
Timed Automata (TA) is a modal model with time-based refinement of states.

For example, let us consider a thermostat model that does not allow chattering (frequent transition between modes):

The main differences from **discrete machines** reside in:

- “Bubbles” (States) become **modes**;
- In each **mode**, state is defined by its variables;
- The model tracks the **clock** using a first-order diff.eq.:

$$\dot{s}(t) = 1.$$



Timed automata

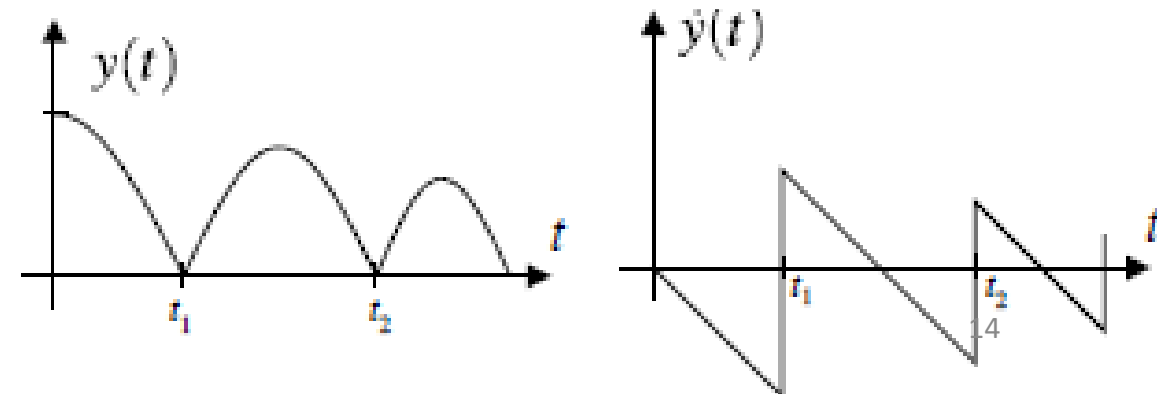
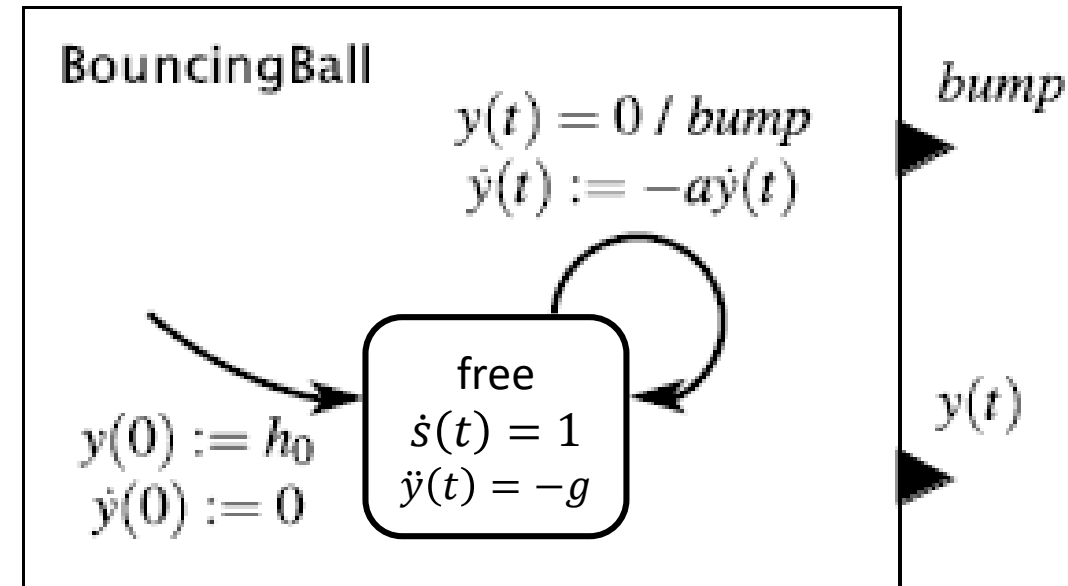
Let us consider another example of an **TA**, which models both continuous and discrete dynamics of a bouncing ball:

Problem description:

- Initial parameters at $t = 0$:
 - Ball is released at high h_0 ($y(0) := h_0$);
 - Initial speed is zero ($\dot{y}(0) := 0$);
- State refinement:
 - Passage of time ($\dot{s}(t) = 1$);
 - Ball position in space ($\ddot{y}(t) = -g$);
- Predicate (guard):
 - Ball hits the ground ($y(t) = 0$)
- Set action (on the predicate):
 - Change ball speed ($\dot{y}(t) = -a\dot{y}(t)$), where $a \in R_{(0,1)}$ – inelastic collision.
 - Pure signal *bump* is produced.

TA characteristics:

- Two outputs (and no inputs);
- Only one mode “free” (instantaneous collision).



Supervisory control

A control system includes four main components:

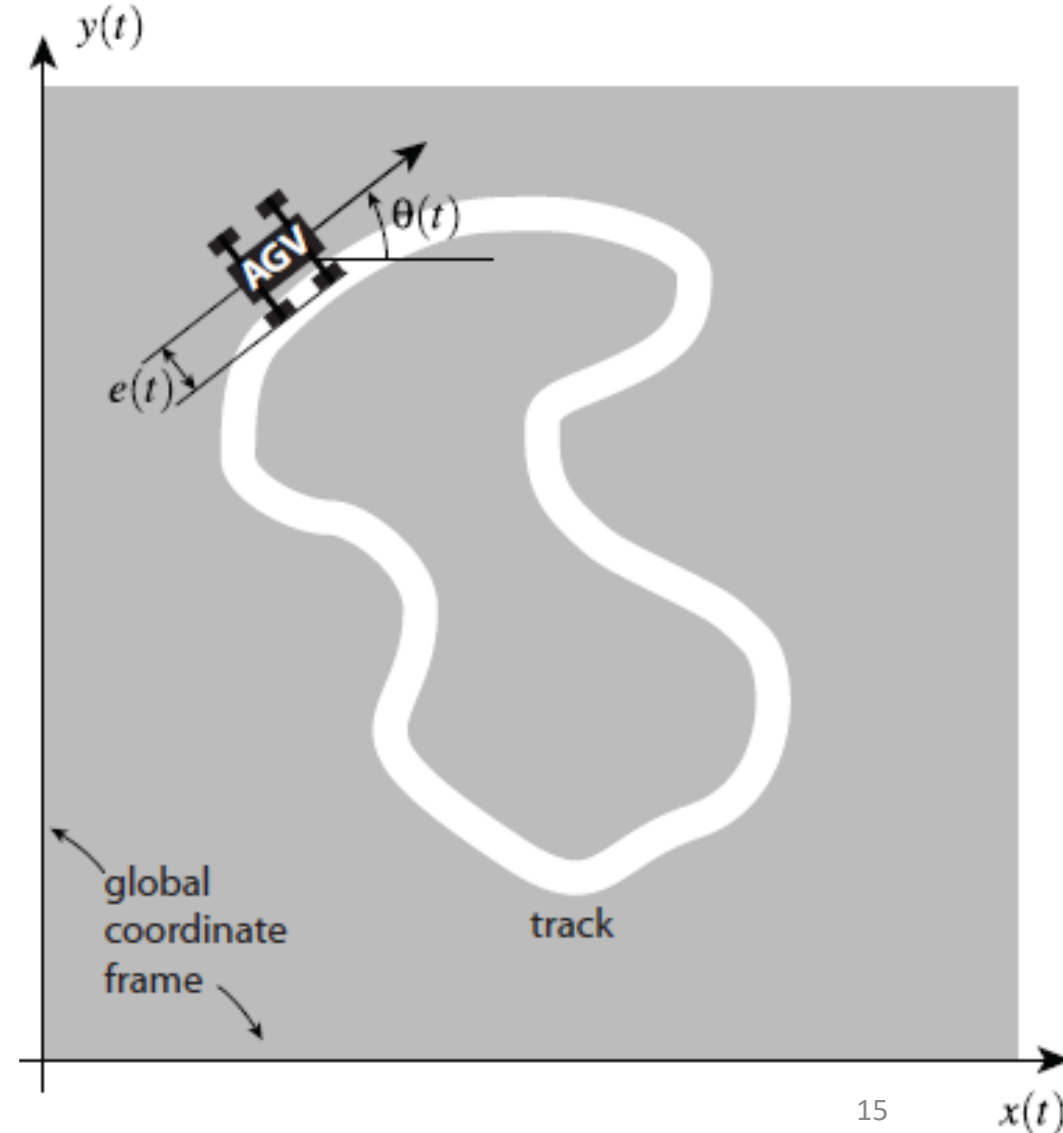
- A system to be controlled (ie, **plant**);
- **Environment** (where system operates);
- **Sensors** (measure some variables of the plant and environment);
- **Controller** (provide control signals to the plant).

The **controller** has two levels:

- **Low-level control** – determines time-based inputs to the plant;
- **Supervisory control** – determines control strategy to follow.

监督控制，决定控制策略

#Let us consider an Automated Guided Vehicle (AGV) that moves along a closed track painted on a warehouse floor.



Supervisory control

Problem description:

- Two control variables (of AGV):
 - Translational speed $u(t) \in R_{[0,10]}$ [m/s];
 - Rotational speed $\omega(t) \in R_{[-\pi,\pi]}$ [rad/s];

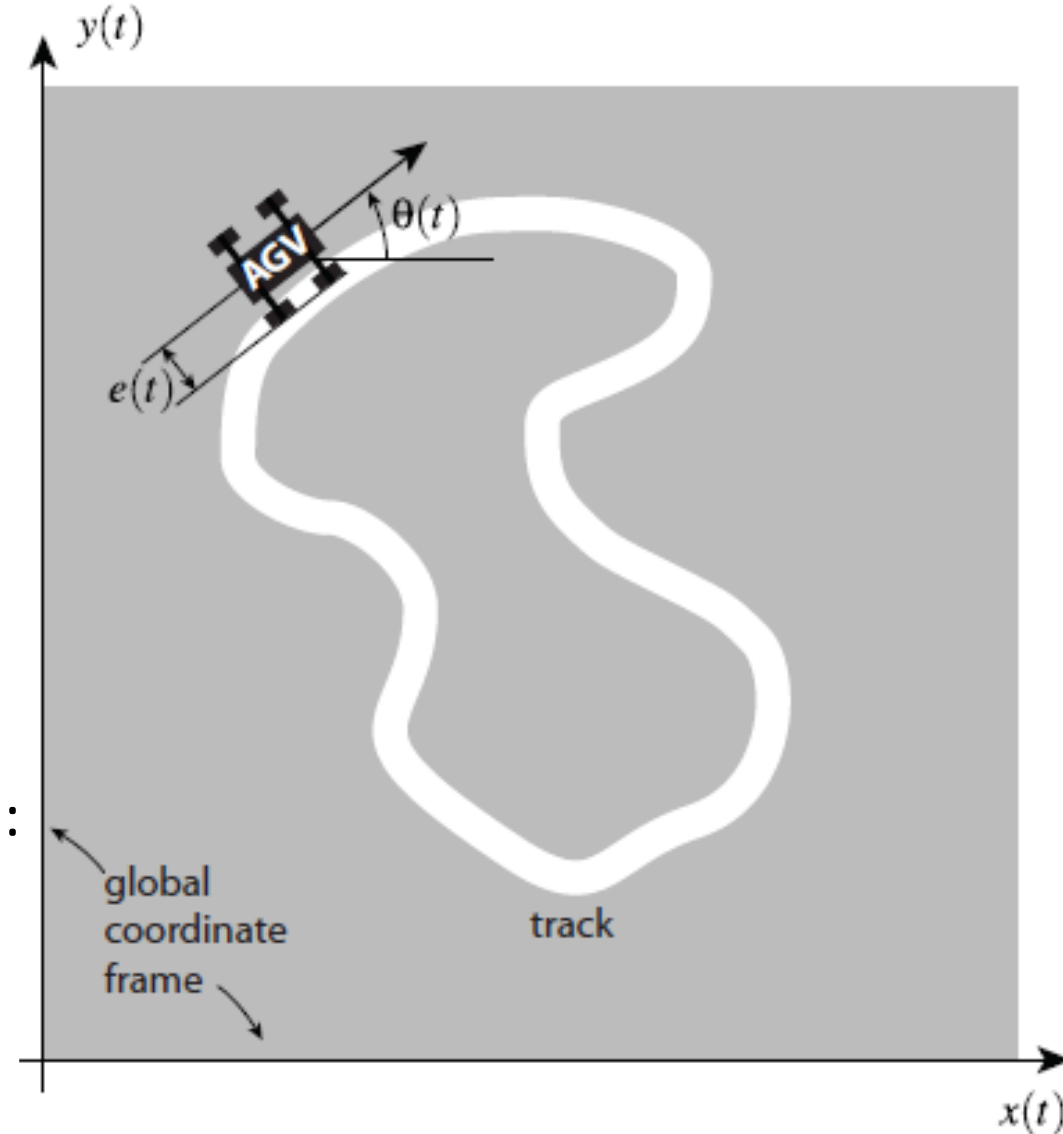
AGV position can be defined with:

- Coordinates $(x(t), y(t)) \in R^2$;
- Orientation $\theta(t) \in R_{(-\pi,\pi]}$.

AGV position can be modelled by the **ODE** system:

$$\begin{cases} \dot{x}(t) = u(t) \cos \theta(t), \\ \dot{y}(t) = u(t) \sin \theta(t), \\ \dot{\theta}(t) = \omega(t). \end{cases}$$

#Let us consider an Automated Guided Vehicle (AGV) that moves along a closed track painted on a warehouse floor.



Supervisory control

Suppose AGV has a sensor that measures distance from the track:

$$e(t) = f(x(t), y(t)),$$

where

$e(t) < 0$ - to the right of the track;

$e(t) > 0$ - to the left of the track.

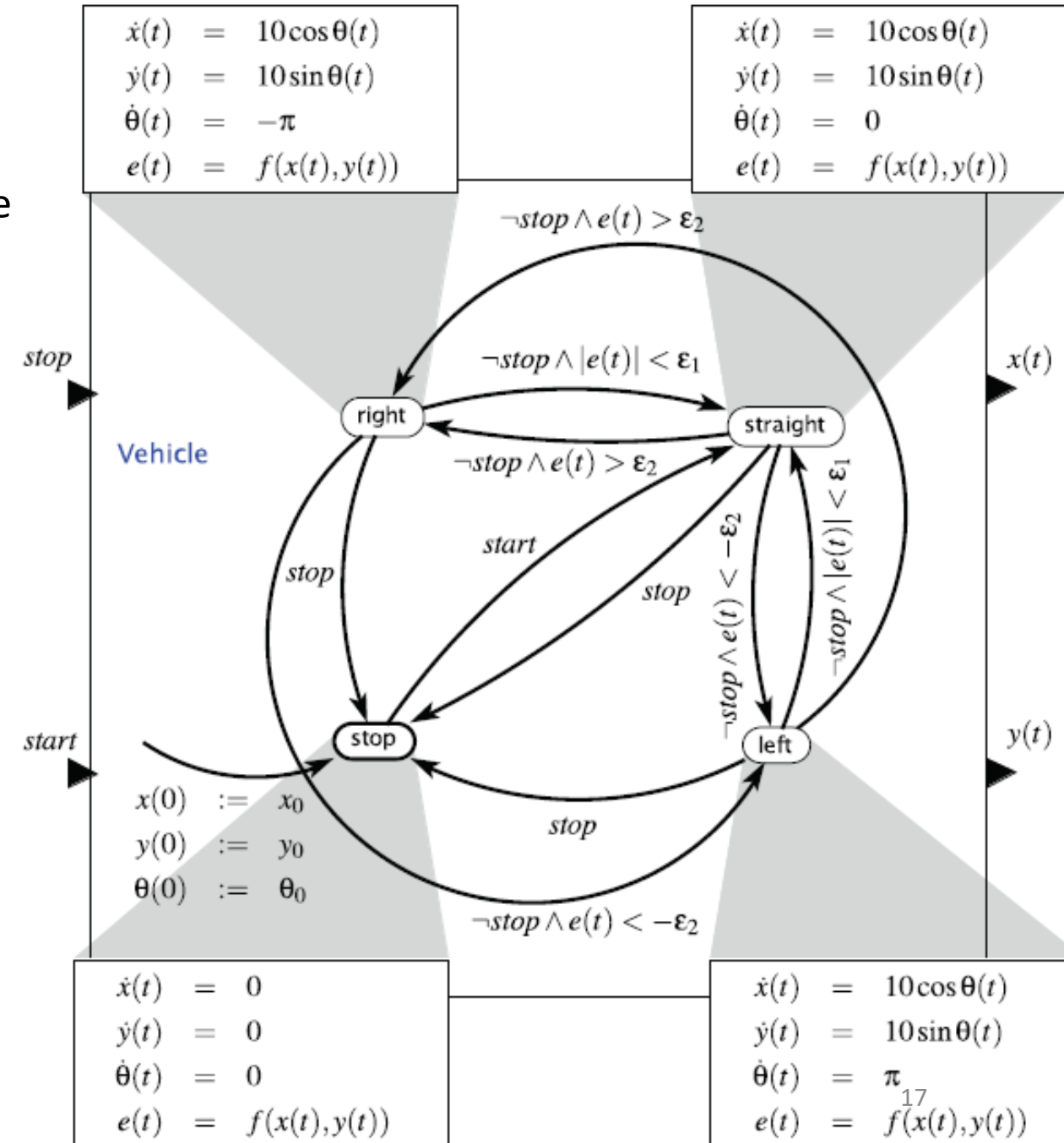
Specify two thresholds $0 < \varepsilon_1 < \varepsilon_2$:

- ε_1 - AGV is close enough;
- ε_2 - AGV far away.

Let's assume the simplest low-level control:

- $u(t) \in \{0, 10\}$ m/s;
- $\omega(t) \in \{-\pi, 0, \pi\}$ rad/s.

AGV Supervisory control model:



To sum up

- A **discrete system** operates in a sequence of discrete steps and is said to have **discrete dynamics**;
- **Discrete systems** can be modelled with **state machines**:
 - Graphical notation useful for FSM and ESM;
 - Mathematical notation to ensure precise model formulation;
- Two main properties of **state machines** are **determinacy** and **receptiveness**, determining if the system can be effectively controlled;
- **Hybrid systems** provide a bridge between continuous and discrete dynamics:
 - **State machines** are used to reflect discrete dynamics (i.e., mode transition);
 - **State refinement** is used to model continuous dynamics within a mode;
- **Timed automata** is the particular example of a hybrid system, where **time-based refinement of states** is modelled;
- **Supervisory control** model has been considered to illustrate the use of **hybrid systems** when designing multi-level control.

The End

See you next time (March 13)