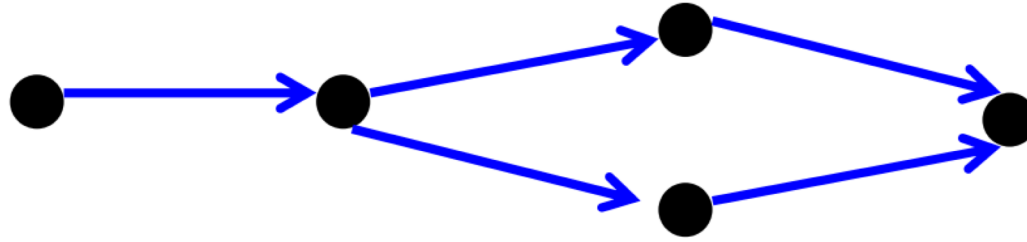# Cyber-Physical Systems

Laura Nenzi

Università degli Studi di Trieste
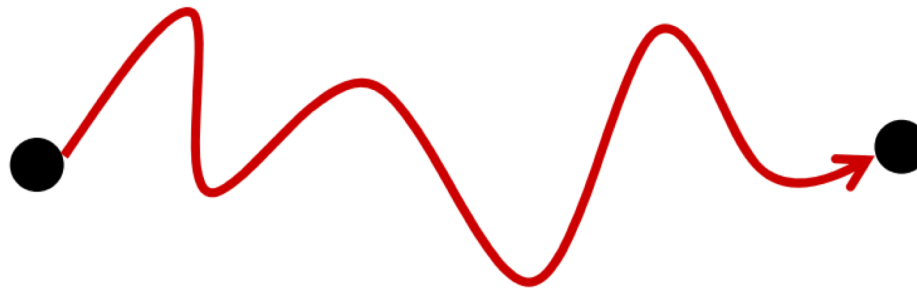II Semestre 2018

Lecture: Hybrid Models
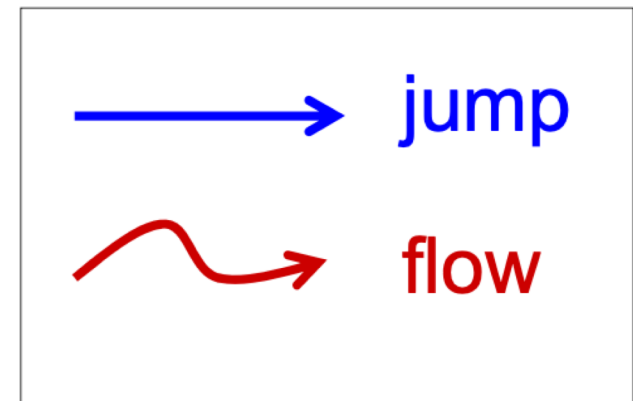
Discrete System (FSM)

Continuous System

Hybrid System

jump

flow
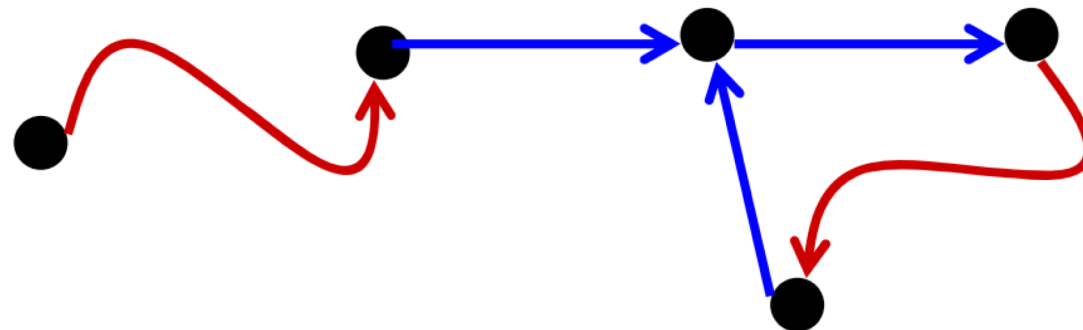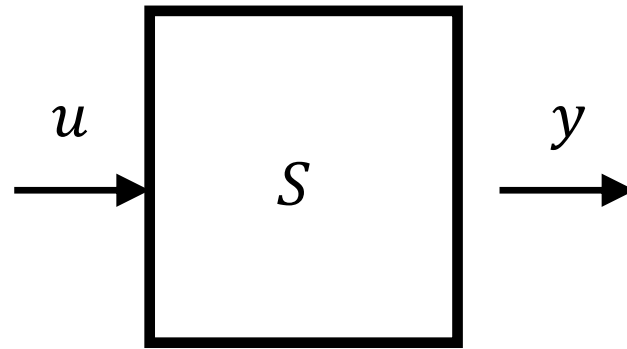
Actor Models

A box, where the inputs and the outputs are functions $\quad S: u \rightarrow y$

Actor models are composable. We can form a **cascade composition**

Having continuous inputs

Input u(t)

Output y
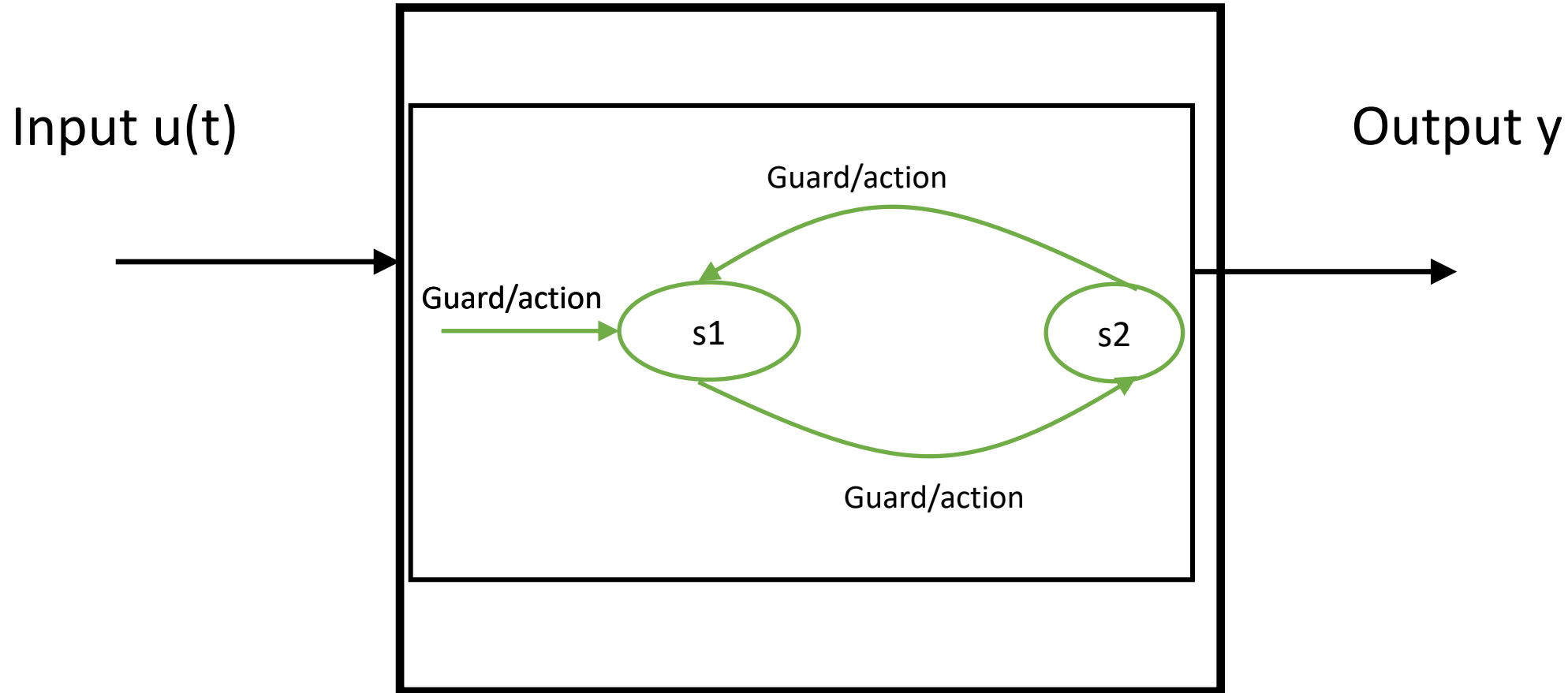
We will define a transition to occur when a guard on an outgoing transition from the current state becomes enabled

Thermostat FSM preventing chattering

$$\text{Temp} \leq 18 \;/\; heatON$$

cooling

heating

$$\text{Temp} \geq 22 \;/\; heatOFF$$
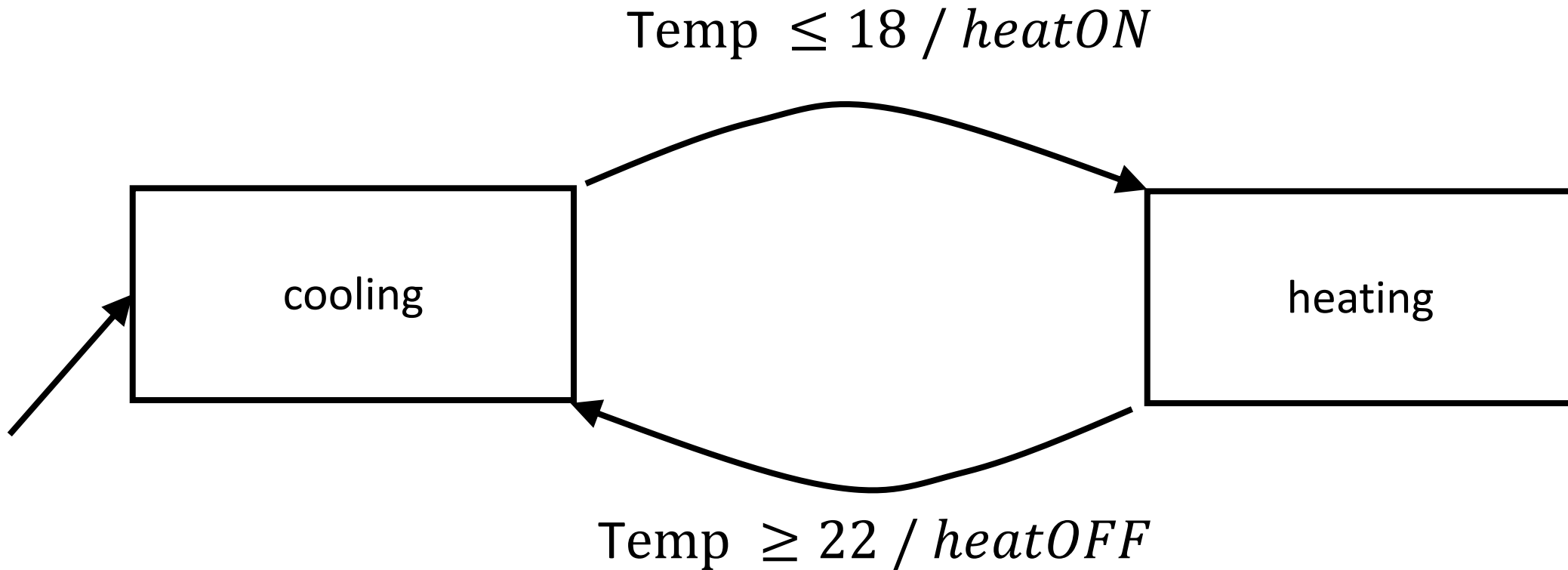
It could be event triggered, like the garage counter, in which case it will react whenever a *temperature* input is provided. Alternatively, it could be time triggered, meaning that it reacts at regular time intervals

Thermostat FSM with a continuous-time input signal
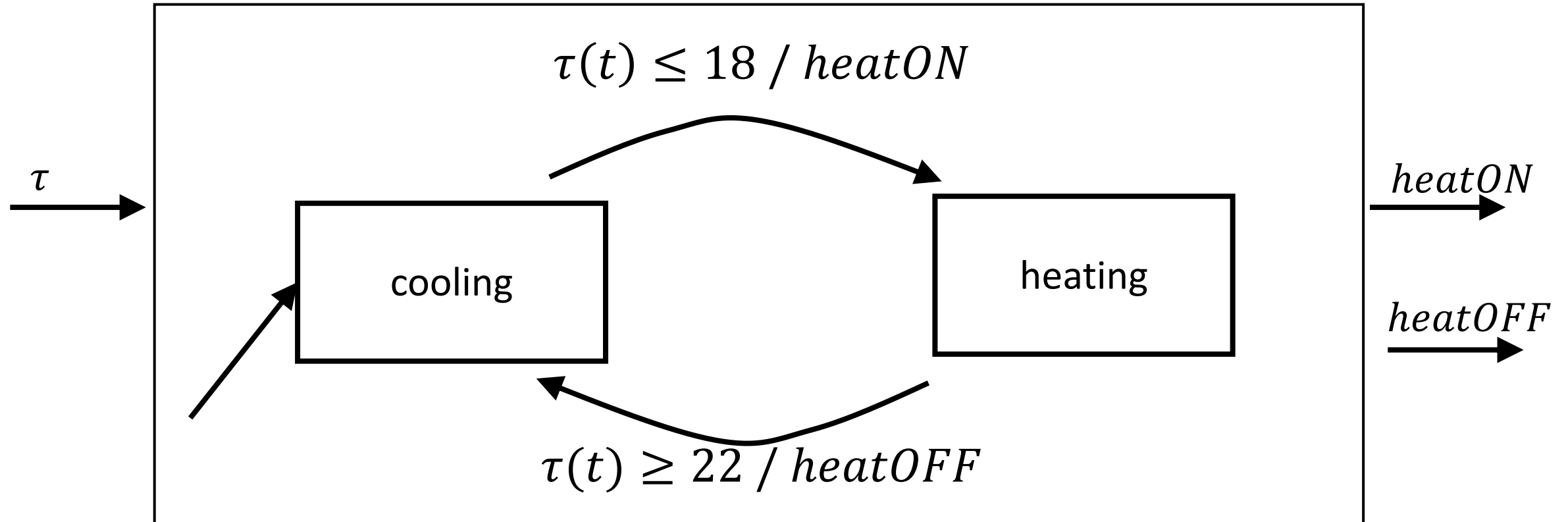
The outputs are present only at the times the transitions are taken

State Refinements
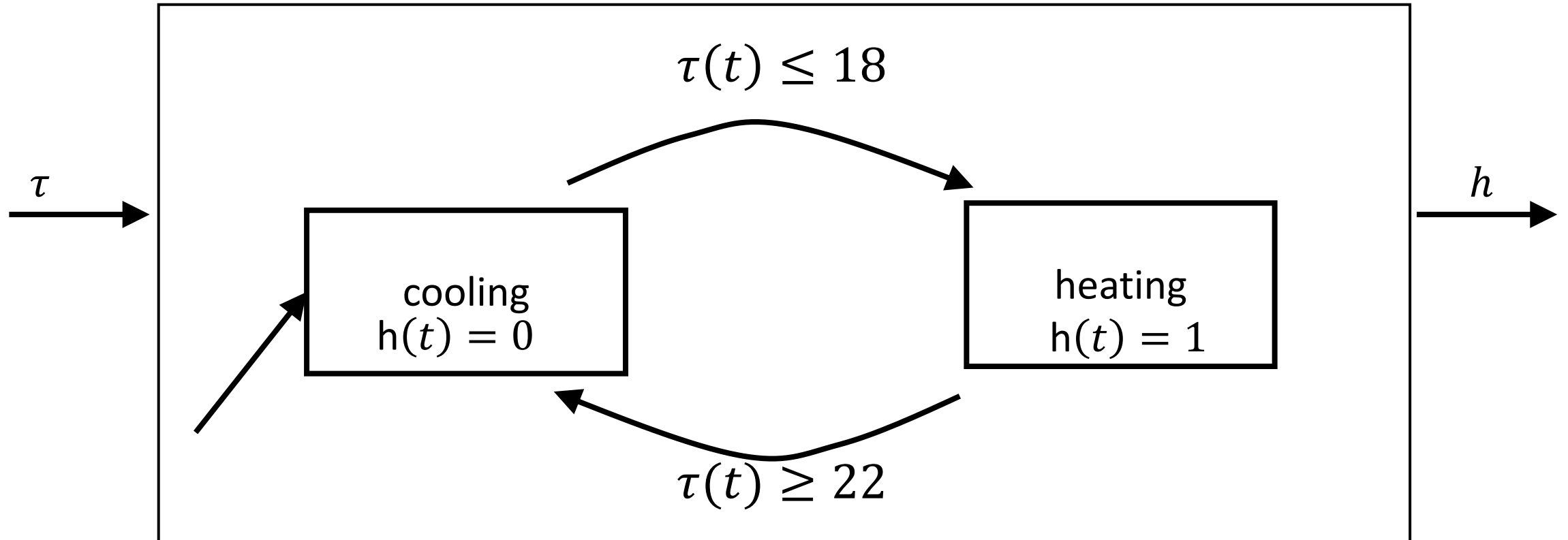
The current state of the state machine has a state refinement that gives the dynamic behavior of the output as a function of the input

Modal Models

A hybrid system is sometimes called a modal model because it has a finite number of modes, one for each state of the FSM, and when it is in a mode, it has dynamics specified by the state refinement.

Timed Automata

➢ Introduced by Alur and Dill (A theory of timed Automata, TCS,1994)

➢ They are the simplest non-trivial hybrid systems

➢ All they do is measure the passage of time

➢ A clock $s(t)$ is modeled by a first-ODE: $\dot{s} = a$ $\qquad \forall t \in T_m$

Timed Automata

$\tau(t) < 20 \wedge s(t) \geq T_c$
$s(t) := 0$

τ →

cooling
$h(t) = 0$
$\dot{s} = 1$

heating
$h(t) = 1$
$\dot{s} = 1$

→ h

$s(t) := T_c$

$\tau(t) \geq 20 \wedge s(t) \geq T_h$
$s(t) := 0$

This is an assignment, not a predicate
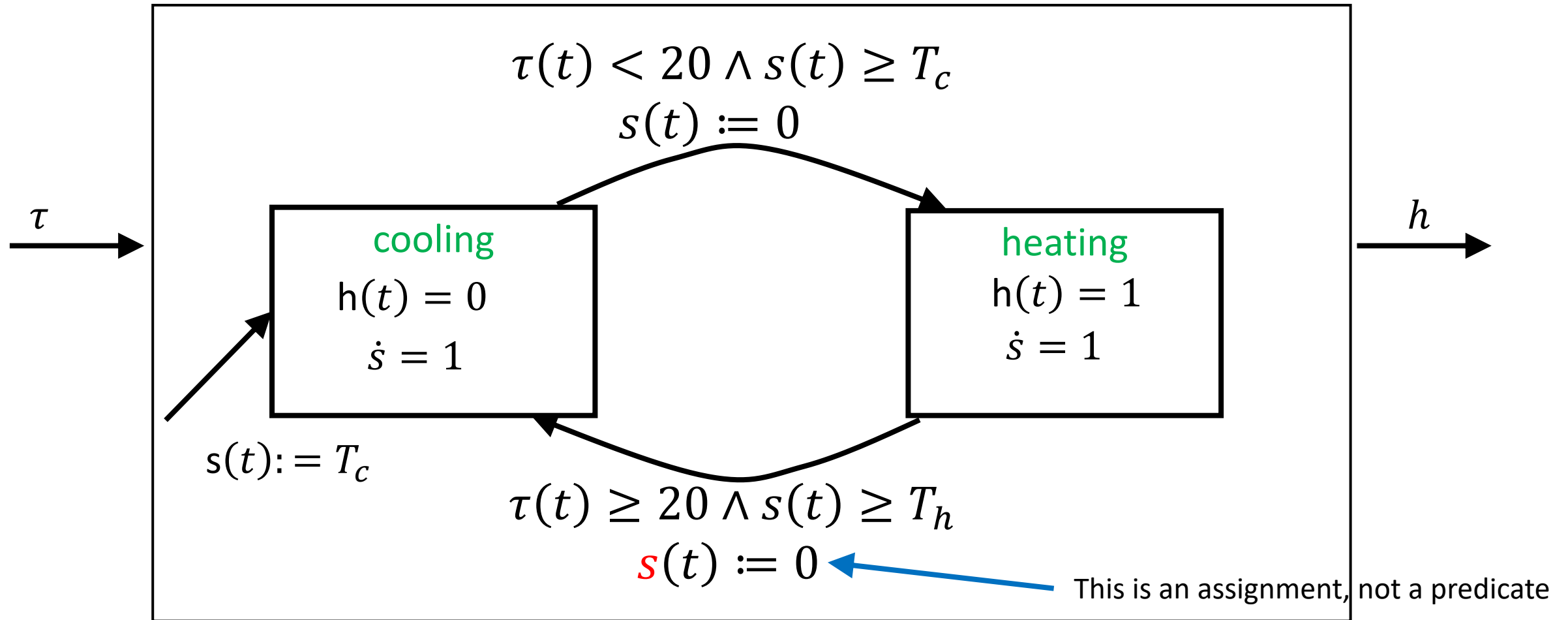
cooling and heating are discrete states, s is a continuous state

Timed Automata

$$\tau(t) < 20 \wedge s(t) \geq T_c$$
$$s(t) := 0$$

τ →

cooling

$\text{h}(t) = 0$

$\dot{s} = 1$

heating

$\text{h}(t) = 1$

$\dot{s} = 1$

→ h

$s(t) := T_c$

$$\tau(t) \geq 20 \wedge s(t) \geq T_h$$
$$s(t) := 0$$

This is an assignment, not a predicate
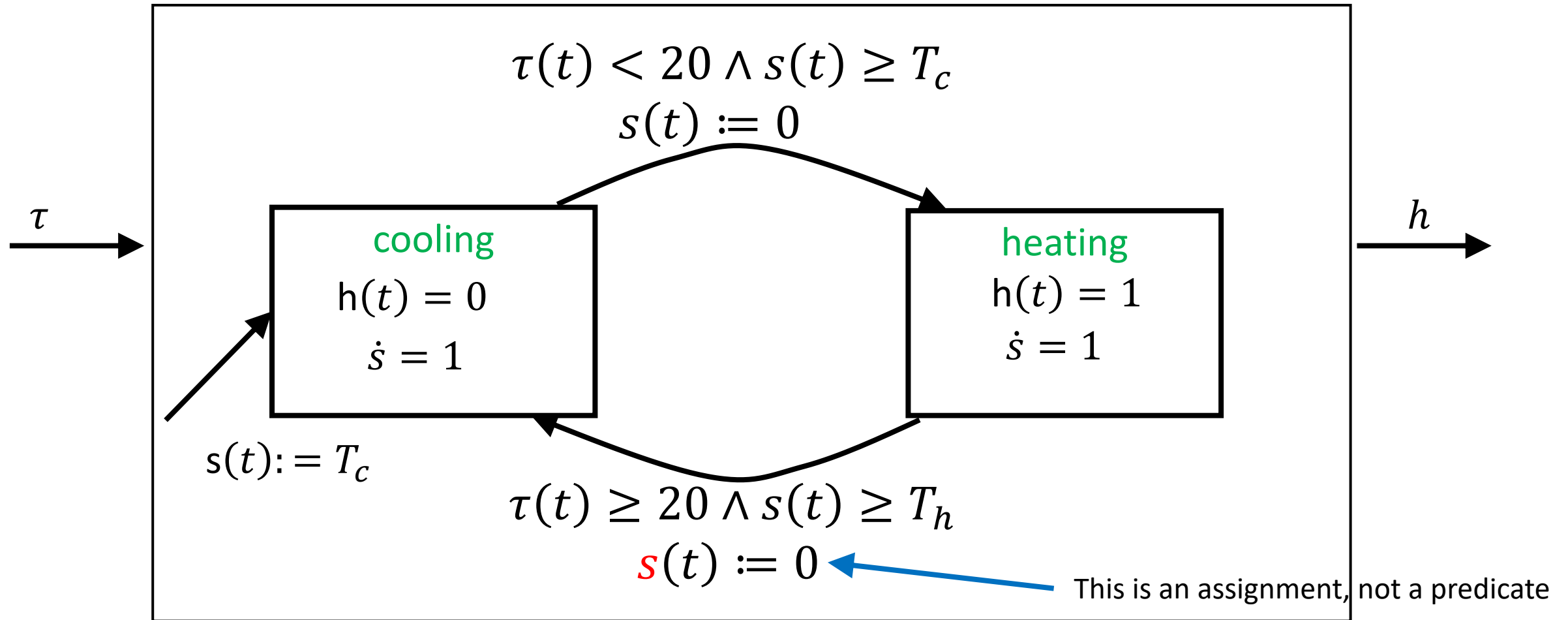
cooling and heating are discrete states, s is a continuous state

Hybrid Automata

$$\tau \leq 18$$

cooling

$\dot{\tau} = -\alpha\,\tau$

$\tau \geq 18$

$h = 0$

heating

$\dot{\tau} = -\alpha\,\tau + k$

$\tau \leq 20$

$h = 1$

$\tau \geq 18$

$\tau \geq 22$

h

τ
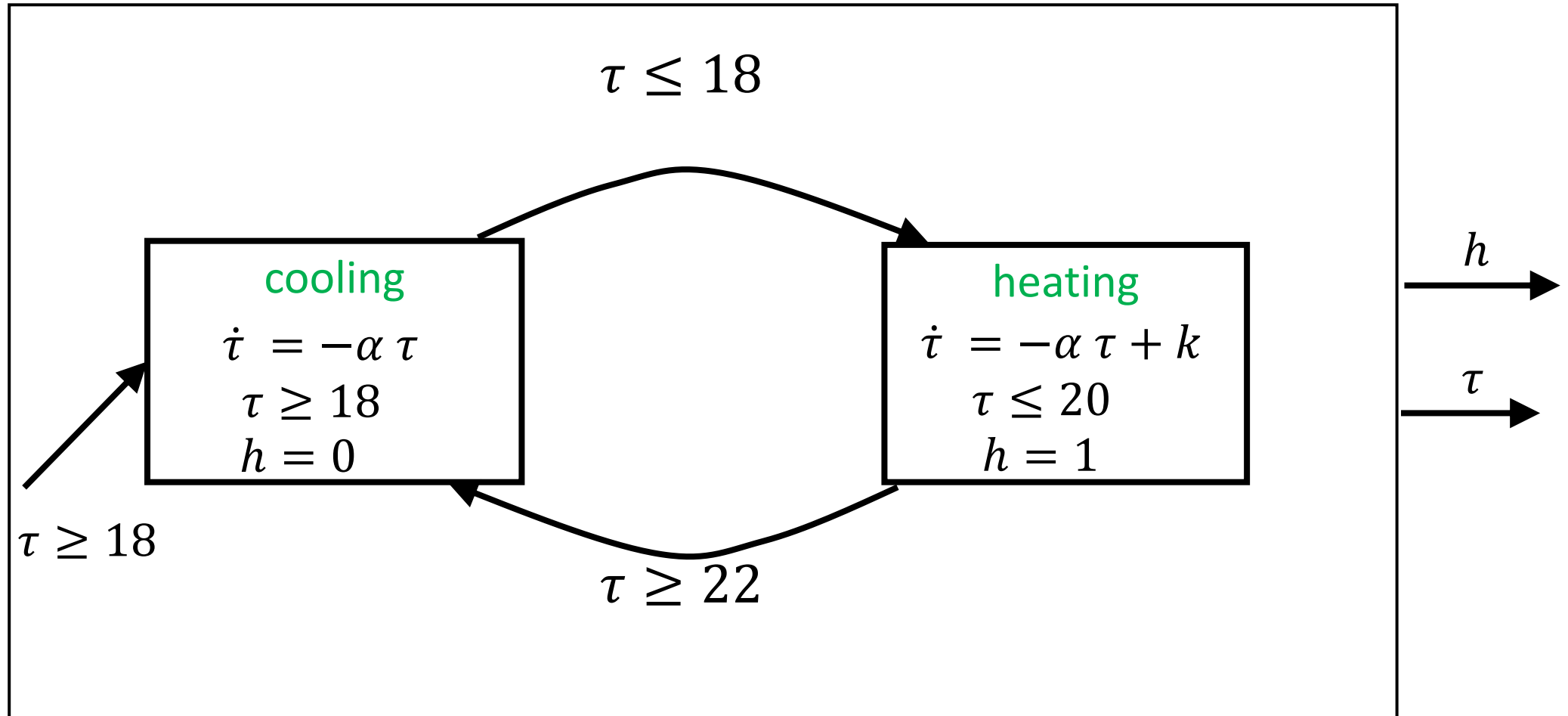
Modeling a bouncing ball

▶ Ball dropped from an initial height of h_0 with an initial velocity of v_0

▶ Velocity changes according to $\dot{v} = -g$

▶ When ball hits the ground, i.e. when $h(t) = 0$, velocity changes discretely from negative (downward) to positive (upward)

　▶ I.e. $v(t^+) := -av(t)$, where t^+ is just after t, and a is a damping constant

▶ Can model as a hybrid system!

Hybrid Process for Bouncing ball

$h = 10, v = 15;$ → $\begin{array}{c} \dot{y} = v \\ \dot{v} = -g \end{array}$

$h \leq 0$
$v := -av$
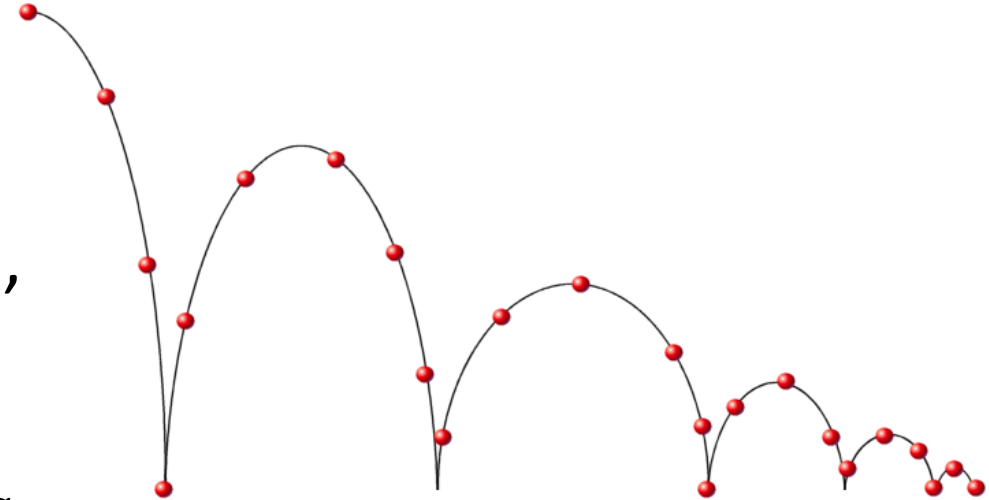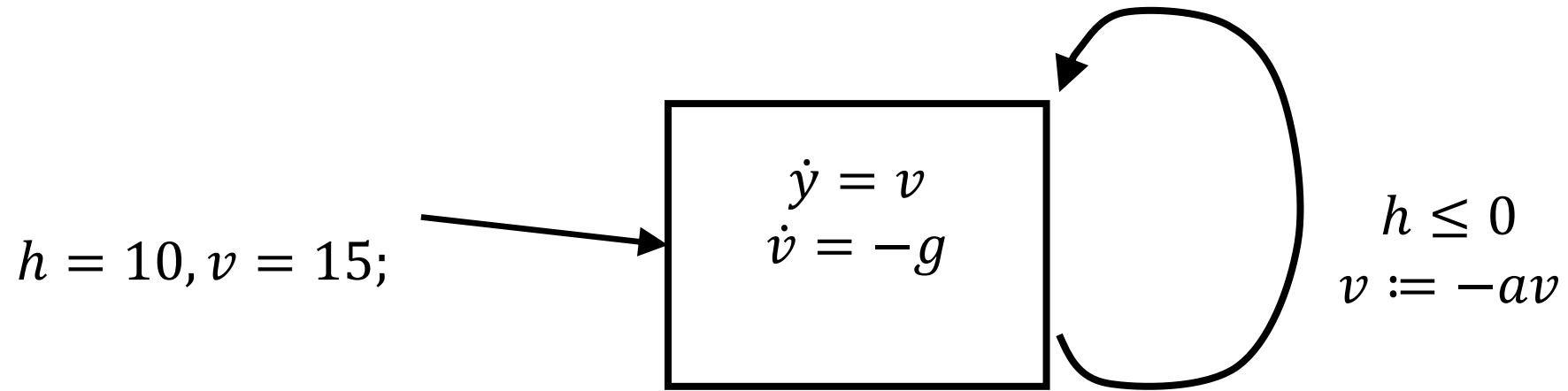
What happens as $h \to 0$?

Hybrid Time Set

A hybrid time set is a finite or infinite sequence of intervals

$\tau = \{ I_i, i = 0, \ldots, M \}$:

- $I_i = [\tau_i, \tau_i'] \; for \; i < M$
- $I_M = [\tau_M, \tau_M']$ or $I_M = [\tau_M, \tau_M')$ if M<∞
- $\tau_i' = \tau_{i+1}$
- $\tau_i \le \tau_i'$
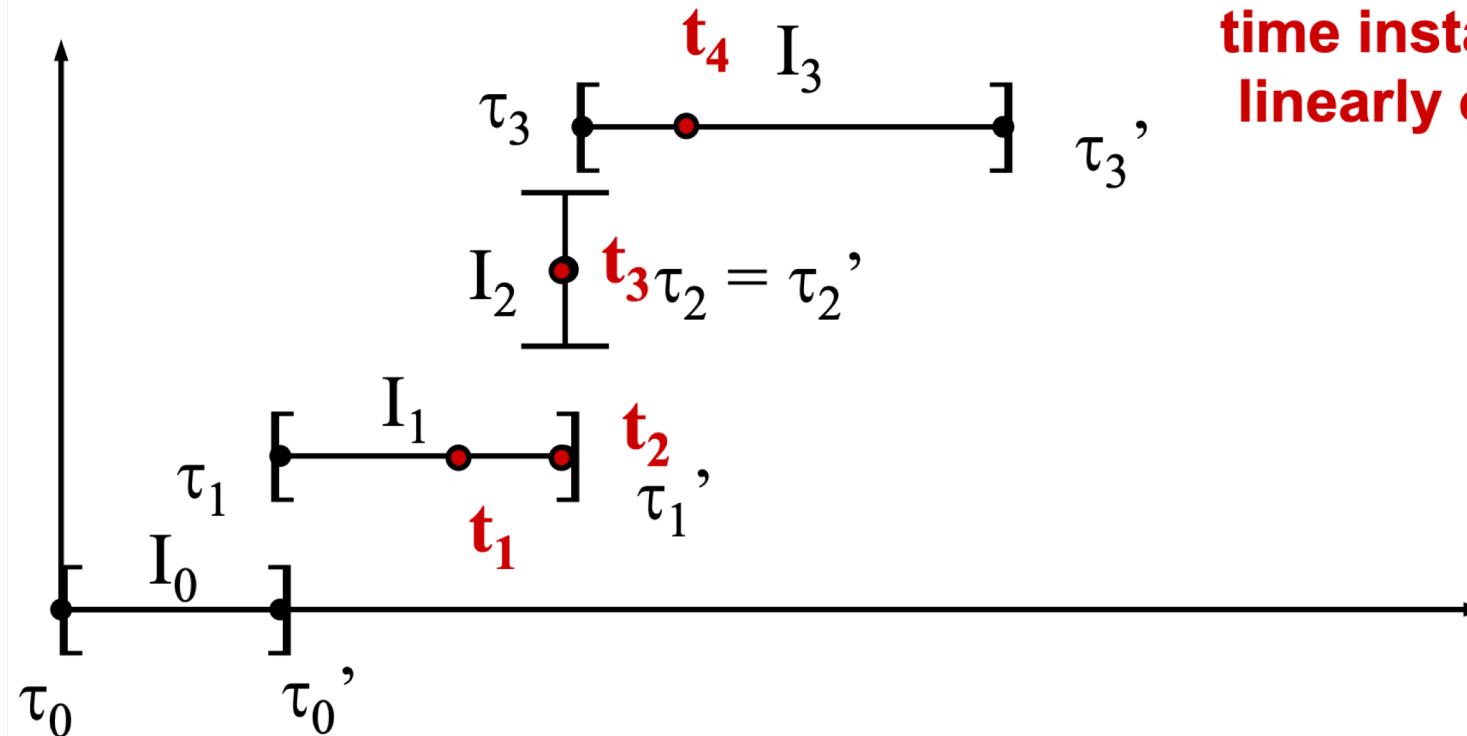
$t_1 \prec t_2 \prec t_3 \prec t_4$

time instants in τ are linearly ordered

Hybrid Time Set: Length

Two notions of length for a hybrid time set $\tau = \{ I_i, i = 0, \dots, M \}$:

- Discrete extent: $< \tau > = M + 1$ $\qquad$ number of discrete transition
- Continuous extent: $\left\| \tau \right\| = \sum_{i=0}^{M} |\tau_i' - \tau_i|$ $\qquad$ total duration of interval in τ
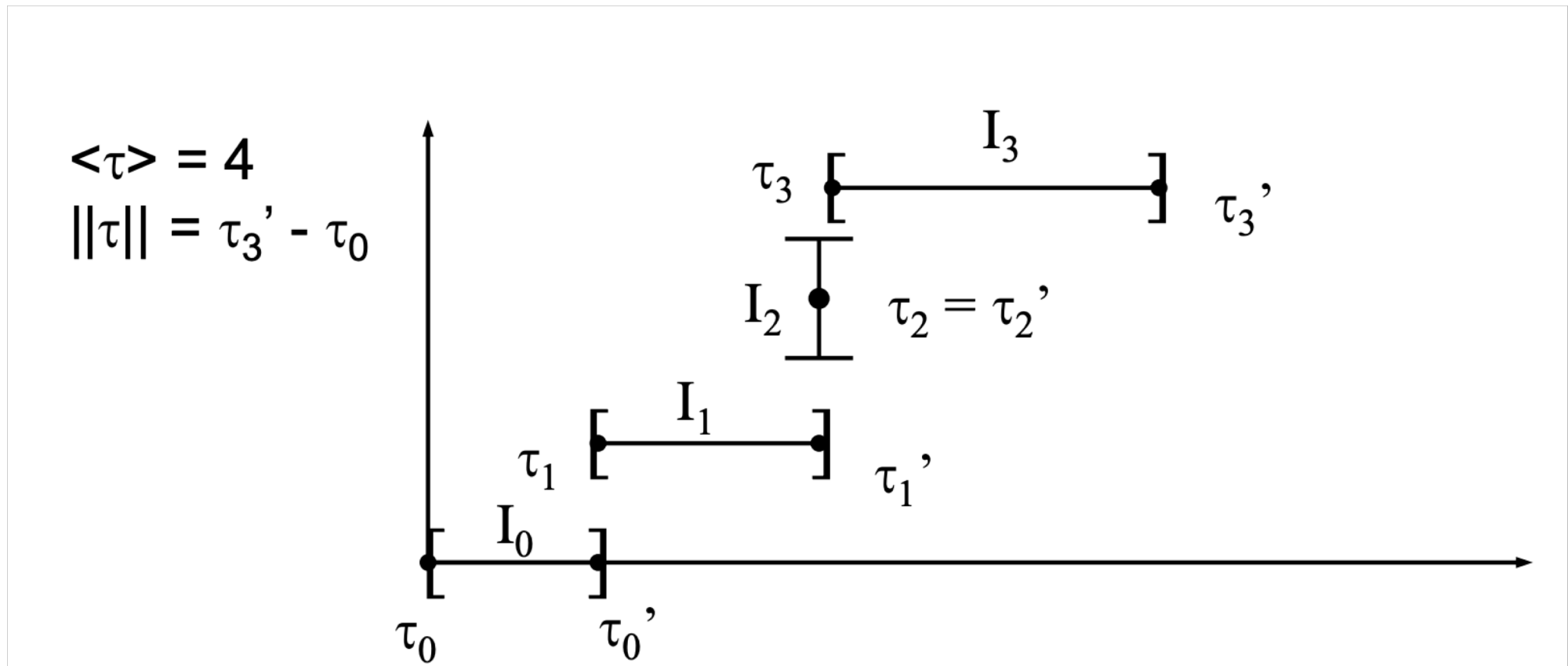
Hybrid Time Set: Classification

A hybrid set $\tau = \{ I_i, i = 0, \dots, M \}$ is :

- Finite: if $< \tau >$ is finite and $I_M = [\tau_M, \tau'_M]$
- Infinite: if $||\tau||$ is infinite
- Zeno: if $< \tau >$ is infinite but $||\tau||$ is finite
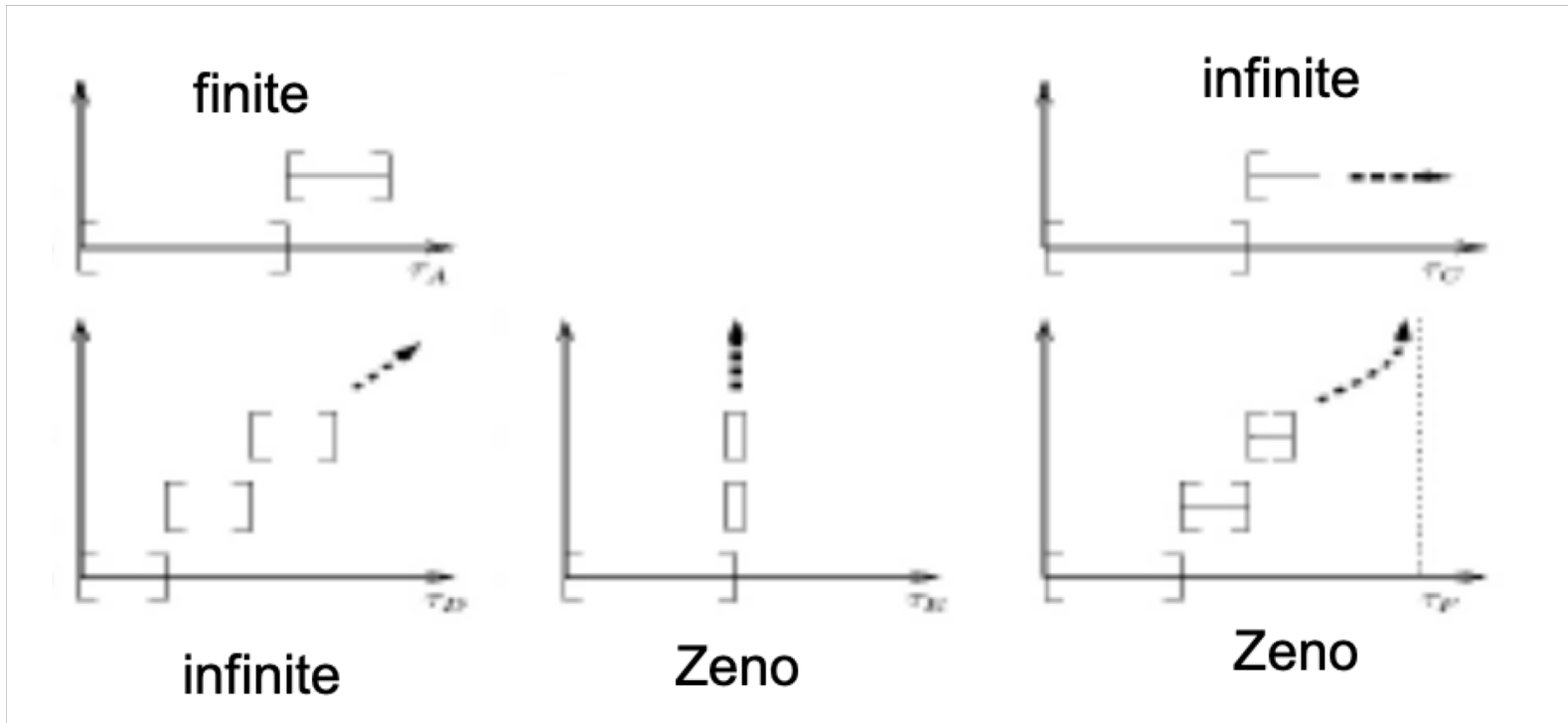
Zeno's Paradox

▶ Described by Greek philosopher Zeno in context of a race between Achilles and a tortoise

▶ Tortoise has a head start over Achilles, but is much slower

▶ In each discrete round, suppose Achilles is d meters behind at the beginning of the round

▶ During the round, Achilles runs d meters, but by then, tortoise has moved a little bit further

▶ At the beginning of the next round, Achilles is still behind, by a distance of $a \times d$ meters, where a is a fraction 0<a<1

▶ By induction, if we repeat this for infinitely many rounds, Achilles will never catch up!

Non-Zeno hybrid process for bouncing ball

faling

$$\dot{h} = v$$
$$\dot{v} = -g$$

$h = 10, v = 15$

$h = 0 \rightarrow$
$v := -av$

$h = 0 \land v < \epsilon \rightarrow$
$v := 0$

halt
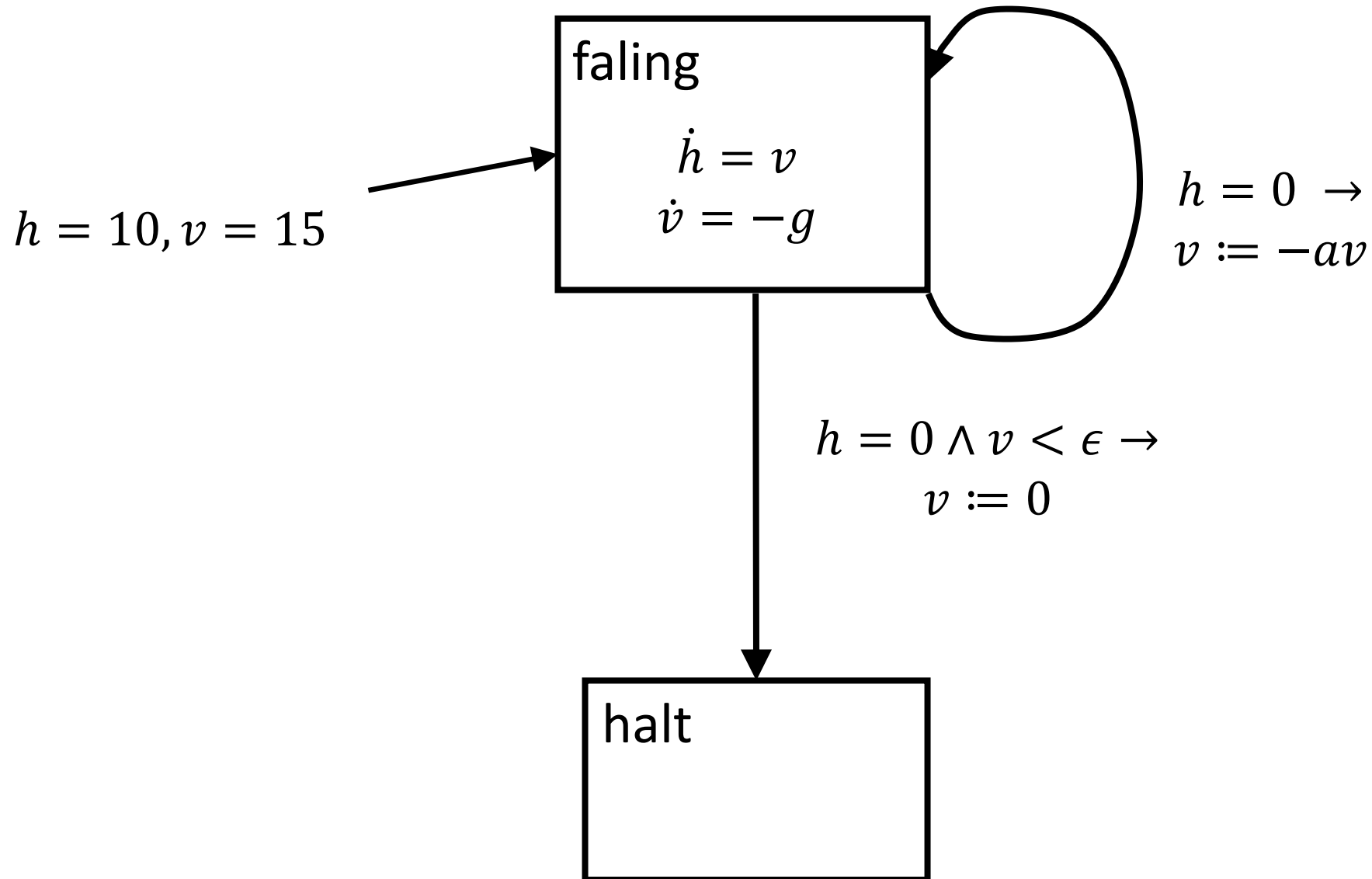
Hybrid Process

▶ Inputs, Outputs, States (both continuous and discrete), Internal actions, input and output actions exactly like the asynchronous model

▶ Continuous action/transition:

$$(q, \mathbf{x}_\tau) \xrightarrow[\delta]{\mathbf{u}(t)/\mathbf{y}(t)} (q, \mathbf{x}(t + \delta))$$

   ▶ Discrete mode m does not change

   ▶ $\mathbf{x}(0) = \mathbf{x}_\tau$

   ▶ $\dfrac{d\mathbf{x}(t)}{dt}$ satisfies the given dynamical equation for mode m

   ▶ Output $\mathbf{y}$ satisfies the output equation for mode m: $\mathbf{y}(t) = h_q(\mathbf{x}(t), \mathbf{u}(t))$

   ▶ At all times $t \in [0, \delta]$, the state $\mathbf{x}(t)$ satisfies the invariant for mode m

Hybrid Process

▶ Discrete action/transition:

$$(q, \mathbf{x}_\tau) \xrightarrow{\;g(\mathbf{x})/\mathbf{x} := r(\mathbf{x})\;} \left(q', r(\mathbf{x}_\tau)\right)$$

- ▶ Happens instantaneously
- ▶ Changes discrete mode q to q'
- ▶ Can execute only if $g(\mathbf{x}_\tau)$ evaluates to true
- ▶ Changes state variable value from $\mathbf{x}_\tau$ to $r(\mathbf{x}_\tau)$
- ▶ $r(\mathbf{x}_\tau)$ should satisfy mode invariant of q$'$
- ▶ Some definitions make g a function of $\mathbf{x}$ and $\mathbf{u}$
- ▶ Output will change from $h_q(\mathbf{x}_\tau)$ to $h_{q'}\left(r(\mathbf{x}_\tau)\right)$

Stability of hybrid systems

▶ Hybrid systems can have surprising results with respect to stability

▶ No uniform method like Lyapunov analysis for analyzing all hybrid systems

▶ Example: Piecewise Linear (PWL) Dynamical System
  ▸ Special class of hybrid system, in which each mode has linear dynamics, guards, resets are all linear/affine

  ▸ Each mode in the PWL system can have stable dynamics (by doing eigen-value analysis), but resulting hybrid system may be unstable, or the opposite, each model in unstable but the resulting hybrid system is stable.

Example

$$x_2 = -0.2x_1 \ ?$$

$$\dot{\mathbf{x}} = \begin{pmatrix} -1 & 10 \\ -100 & -1 \end{pmatrix} \mathbf{x}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} -1 & 100 \\ -10 & -1 \end{pmatrix} \mathbf{x}$$

$$x_2 = 5x_1 \ ?$$
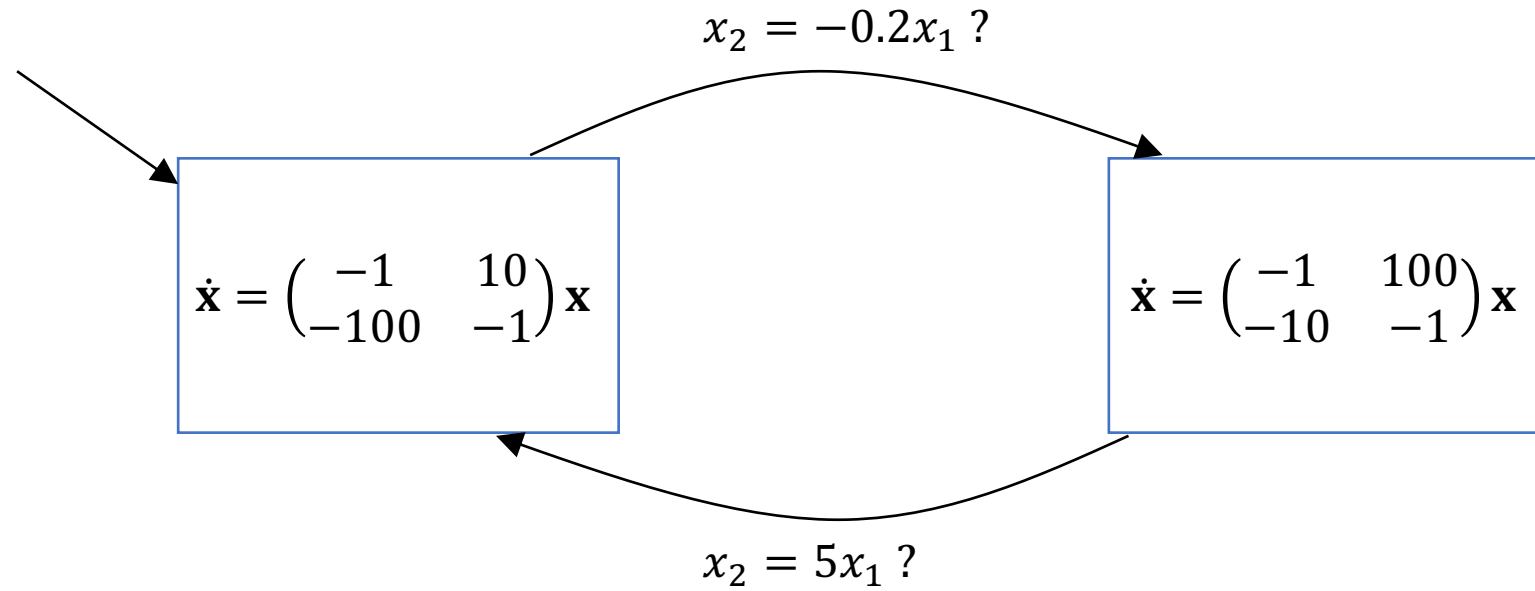
Dynamics in each mode are stable!

Simulation results

Design Application: Autonomous Guided Vehicle

▶ Objective: Steer vehicle to follow a given track

▶ Control inputs: linear vehicle speed (v), vehicle angular velocity (ω), start/stop

▶ Constraints on control inputs:
  ▸ $v \in \{v_{\max}, v_{\max}/2, 0\}$
  ▸ $\omega \in \{-\pi, 0, \pi\}$

▶ Designer choice: $v = v_{\max}$ only if $\omega = 0$, otherwise $v = \dfrac{v_{\max}}{2}$

On/Off control for Path following

Turn right

$$\dot{x} = (v_{\max}/2)\cos\theta$$
$$\dot{y} = (v_{\max}/2)\sin\theta$$
$$\dot{\theta} = -\pi$$
$$d \geq \epsilon$$

$d \leq \epsilon?$

$d \geq \epsilon?$

Go straight

$$\dot{x} = v_{\max}\cos\theta$$
$$\dot{y} = v_{\max}\sin\theta$$
$$\dot{\theta} = 0$$
$$-\epsilon \leq d \leq \epsilon$$

$ss?\ start \wedge$
$d \geq \epsilon?$

$ss?\ stop$

$ss?\ stop$

$ss?\ start \wedge$
$-\epsilon \leq d \leq \epsilon?$

$d \leq -\epsilon?$

$d \geq -\epsilon?$

$x := x_0$
$y := y_0$
$\theta := \theta_0$

$$\dot{x} = 0$$
$$\dot{y} = 0$$
$$\dot{\theta} = 0$$

Stationary

$ss?\ stop$

$ss?\ start \wedge$
$d \leq -\epsilon?$

$$\dot{x} = (v_{\max}/2)\cos\theta$$
$$\dot{y} = (v_{\max}/2)\sin\theta$$
$$\dot{\theta} = \pi$$
$$d \leq -\epsilon$$

Turn left

y

θ

x

d

Track
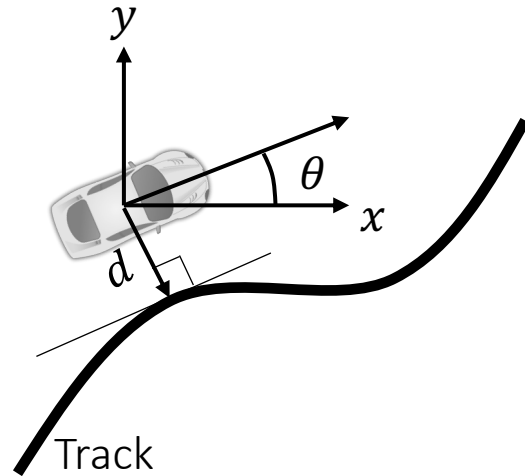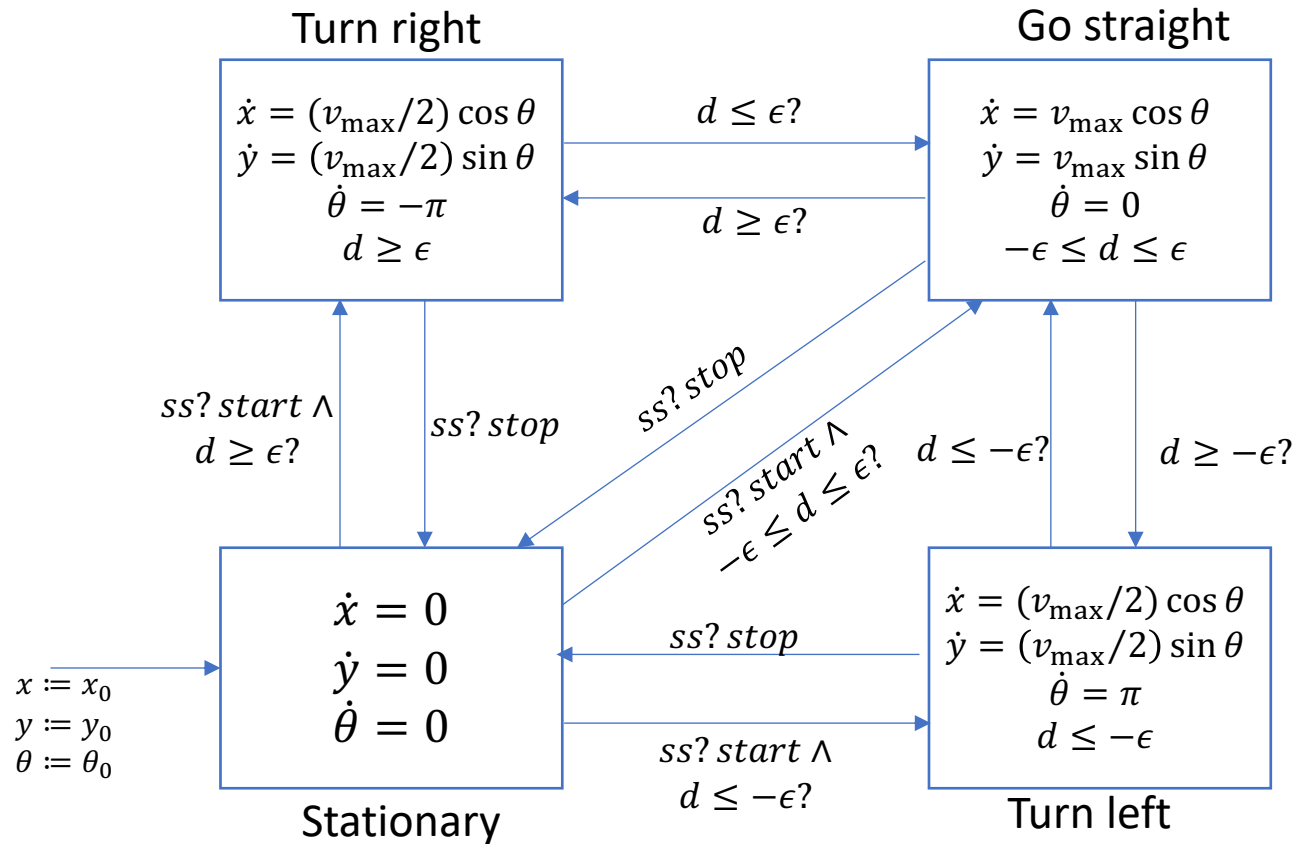
▶ When $d \in [-\epsilon, +\epsilon]$, controller decides that vehicle goes straight, otherwise executes a turn command to bring error back in the interval

Design Application: Robot Coordination

▶ Autonomous mobile robots in a room, goal for each robot:
  ▶ Reach a target at a known location
  ▶ Avoid obstacles (positions not known in advance)
  ▶ Minimize distance travelled

▶ Design Problems:
  ▶ Cameras/vision systems can provide estimates of obstacle positions
    ▶ When should a robot update its estimate of the obstacle position?
  ▶ Robots can communicate with each other
    ▶ How often and what information can they communicate?
  ▶ High-level motion planning
    ▶ What path in the speed/direction-space should the robots traverse?
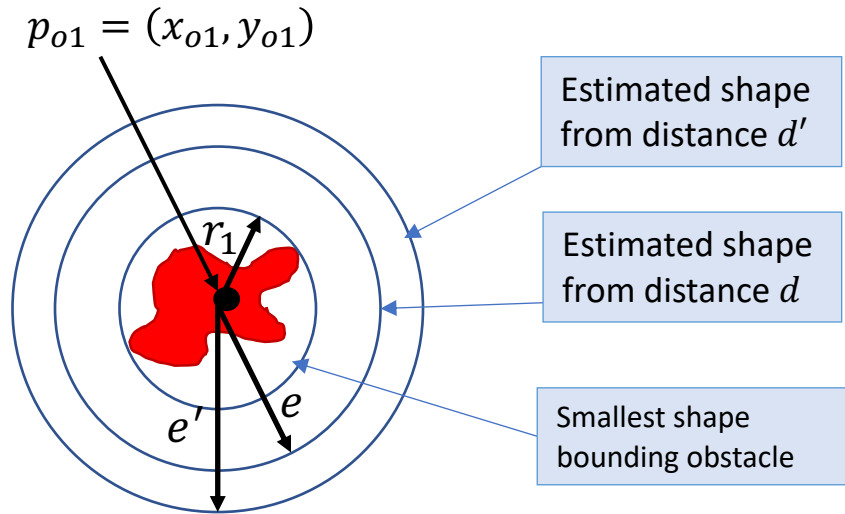
Path planning with obstacle avoidance
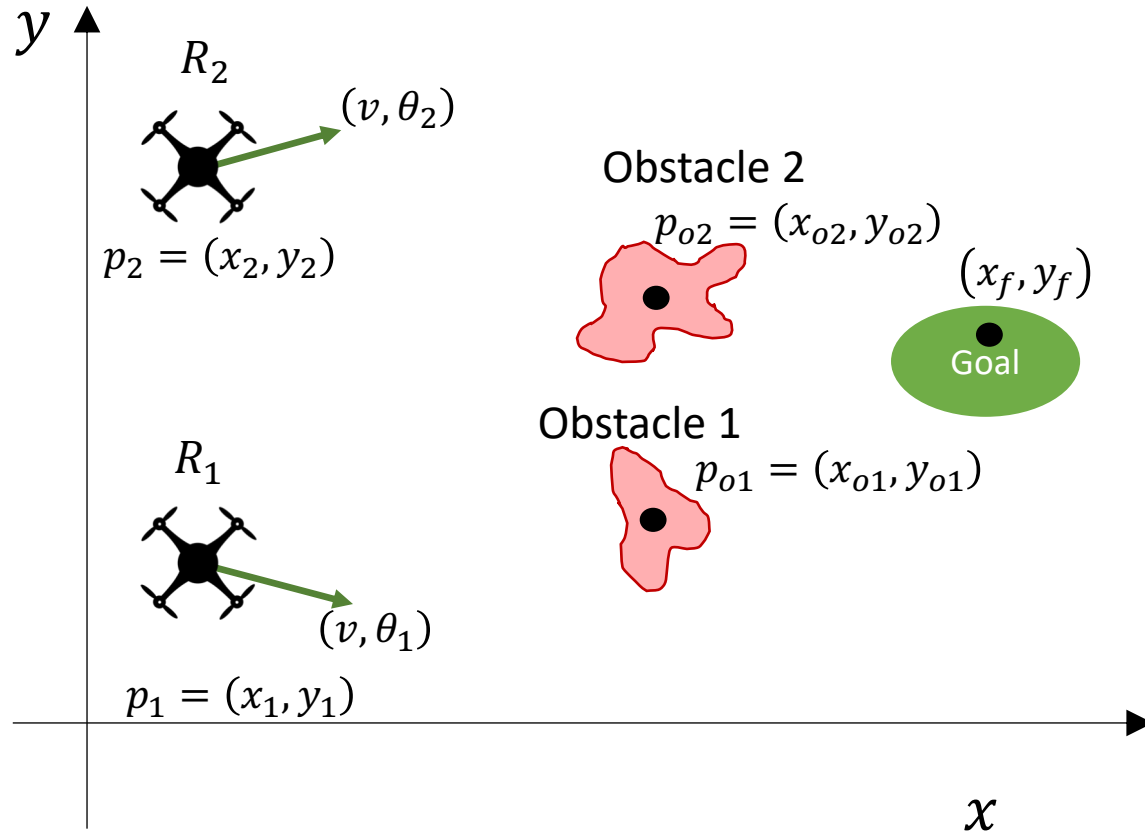
- ▶ Assumptions:
  - ▶ Two-dimensional world
  - ▶ Robots are just points
  - ▶ Each robot travels with a fixed speed
- ▶ Dynamics for Robot R_i:
  - ▶ $\dot{x}_i = v \cos \theta_i; \dot{y}_i = v \sin \theta_i$
- ▶ Design objectives:
  - ▶ Eventually reach (x_f, y_f)
  - ▶ Always avoid Obstacle1 and Obstacle 2
  - ▶ Minimize distance travelled

Divide path/motion planning into two parts

1. Computer vision tasks
2. Actual path planning task

▶ Assume computer vision algorithm identifies obstacles, and labels them with some easy-to-represent geometric shape (such as a bounding boxes)
   ▸ In this example, we will assume a sonar-based sensor, so we will use circles

▶ Assuming the vision algorithm is correct, do path planning based on the estimated shapes of obstacles

▶ Design challenge:
   ▸ Estimate of obstacle shape is not the smallest shape containing the obstacle
   ▸ Shape estimate varies based on distance from obstacle

Estimation error

$p_{o1} = (x_{o1}, y_{o1})$

Estimated shape from distance d'

Estimated shape from distance d

Smallest shape bounding obstacle

r_1

e

e'

Estimated radius (from distance d) $e = r + a(d - r)$,
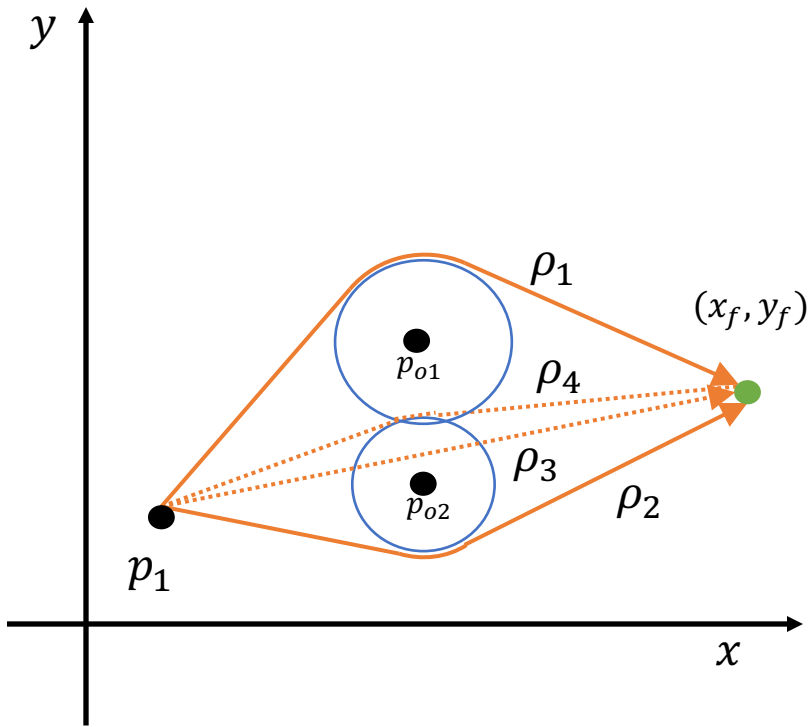where $a \in [0,1]$ is a constant

▶ Robot R_1 maintains radii e_1 and e_2 that are estimates of obstacle sizes

▶ Every τ seconds, R_1 executes following update to get estimates of shapes of each obstacle:

$$e_1 := \min\big(e_1, r_1 + a(\|p_1 - p_{o1}\| - r_1)\big)$$
$$e_2 := \min\big(e_2, r_2 + a(\|p_1 - p_{o2}\| - r_2)\big)$$

▶ Computation of R_2 is symmetric

Path planning

- ▶ Choose shortest path ρ_3 to target (to minimize time)

- ▶ If estimate of obstacle 1 intersects ρ_3, calculate two paths that are tangent to obstacle 1 estimate

- ▶ If estimate of obstacle 2 intersects ρ_3, or obstacle 1, calculate tangent paths

- ▶ Plausible paths: ρ_1 and ρ_2

- ▶ Calculate shorter one as the planned path

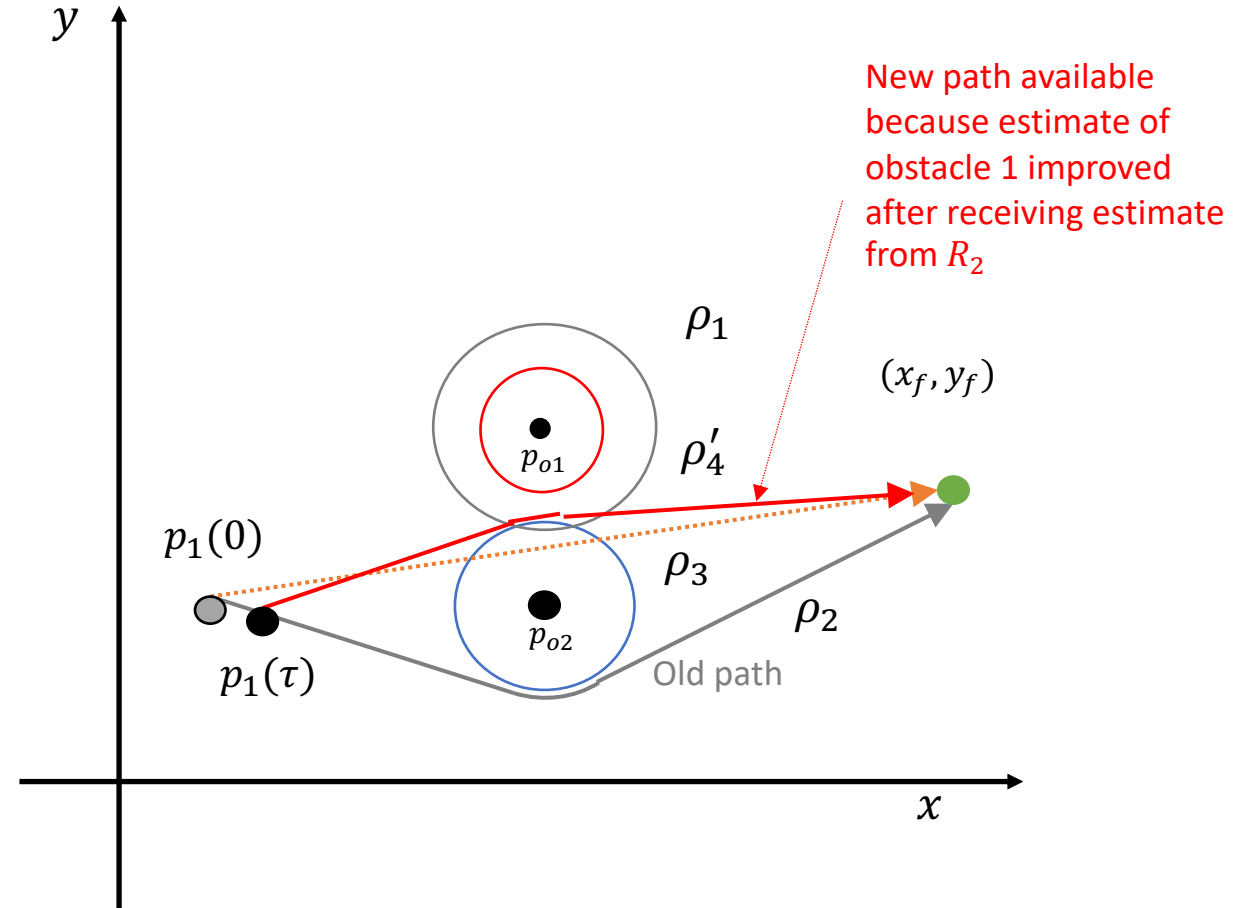# Dynamic path planning

- Path planning inputs:
  - Current position of robot
  - Target position
  - Position of obstacles and estimates

- Output:
  - Direction for motion assuming obstacle estimates are correct

- May be useful to execute planning algorithm again as robot moves!
  - Because estimates will improve closer to the obstacles
  - Invoke planning algorithm every τ seconds

Communication improves planning

▶ Every robot has its own estimate of the obstacle

▶ R_2's estimate of obstacle might be better than R_1's

▶ Strategy: every τ seconds, send estimates to other robot, and receive estimates

▶ For estimate e_i, use final estimate = $\min(e_i, e_i^{recv})$

▶ Re-run path planner

Improved path planning through communication
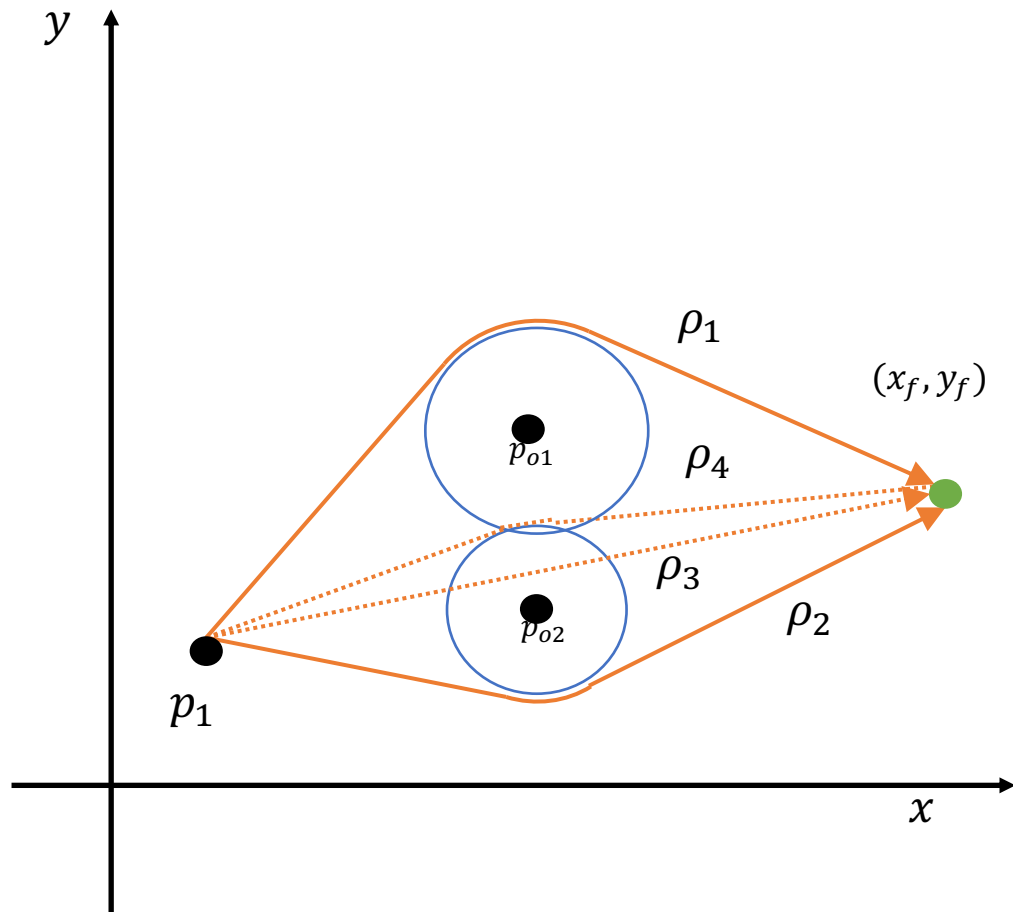
Hybrid State Machine for Communicating Robot

$(z_c = t_c) \rightarrow \{out!\,(e_1, e_2);\ z_c := 0\}$

$(z_p = t_p) \rightarrow$
$\{\theta := \mathtt{plan}(x, y, x_f, y_f, e_1, e_2);\ z_p := 0\}$

$\mathtt{clock}\ z_p,\ z_e,\ z_c := 0$
$x := x_0;\ y := y_0$
$e_1 := r_1 + a(\mathtt{dist}((x, y), (x_o^1, y_o^1)) - r_1)$
$e_2 := r_2 + a(\mathtt{dist}((x, y), (x_o^2, y_o^2)) - r_2)$
$\theta := \mathtt{plan}(x, y, x_f, y_f, e_1, e_2)$

Move
$$\dot{x} = v \cos\theta$$
$$\dot{y} = v \sin\theta$$
$$z_p \leq t_p \ \wedge\ z_c \leq t_c \ \wedge\ z_e \leq t_e$$
$$\wedge\ (x \neq x_f \ \vee\ y \neq y_f)$$

$(x = x_f \ \wedge\ y = y_f)?$

Stop
$$\dot{x} = \dot{y} = 0$$

$(z_e = t_e) \rightarrow \{\ z_e := 0;$
$e_1 := \mathtt{min}(e_1, r_1 + a(\mathtt{dist}((x, y), (x_o^1, y_o^1)) - r_1));$
$e_2 := \mathtt{min}(e_2, r_2 + a(\mathtt{dist}((x, y), (x_o^2, y_o^2)) - r_2))\}$

$in?\,(e_1', e_2') \rightarrow$
$\{e_1 := \mathtt{min}(e_1', e_1);\ e_2 := \mathtt{min}(e_2', e_2)\}$