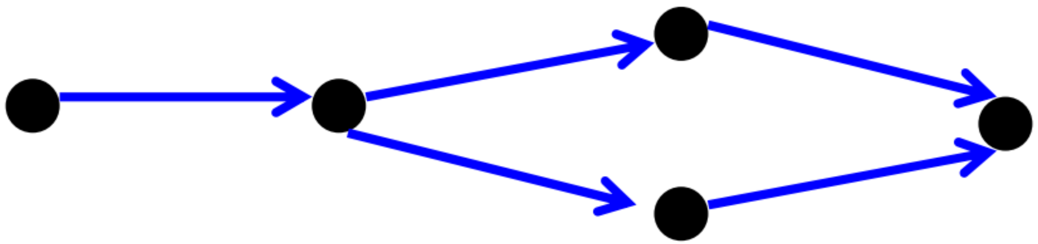
Cyber-Physical Systems

Laura Nenzi

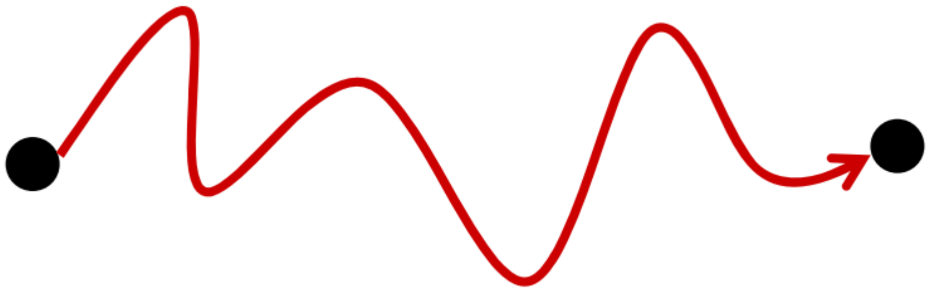
Università degli Studi di Trieste
I Semestre 2023

Lecture 5: Hybrid Models

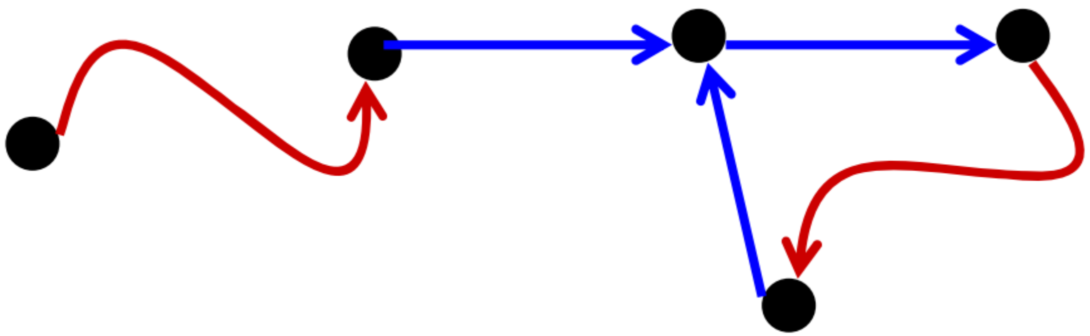
Discrete System (FSM)

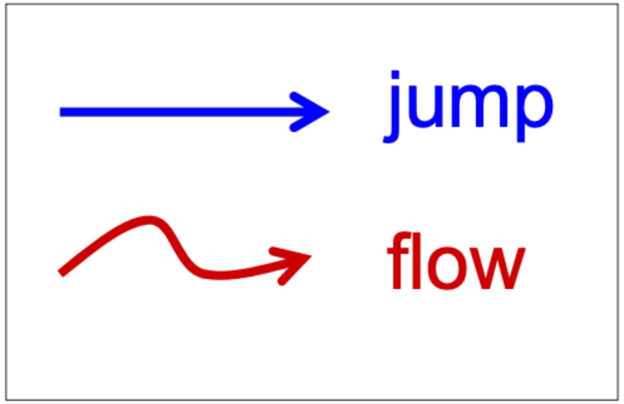


Continuous System



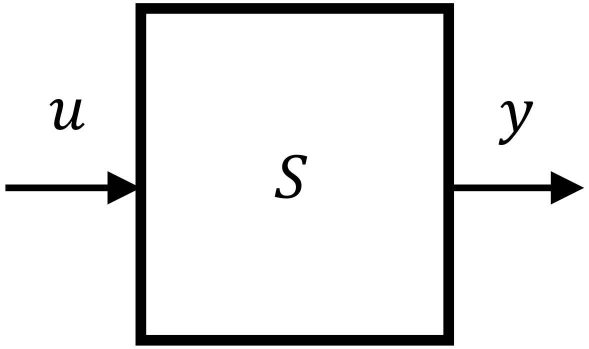
Hybrid System





Actor Models

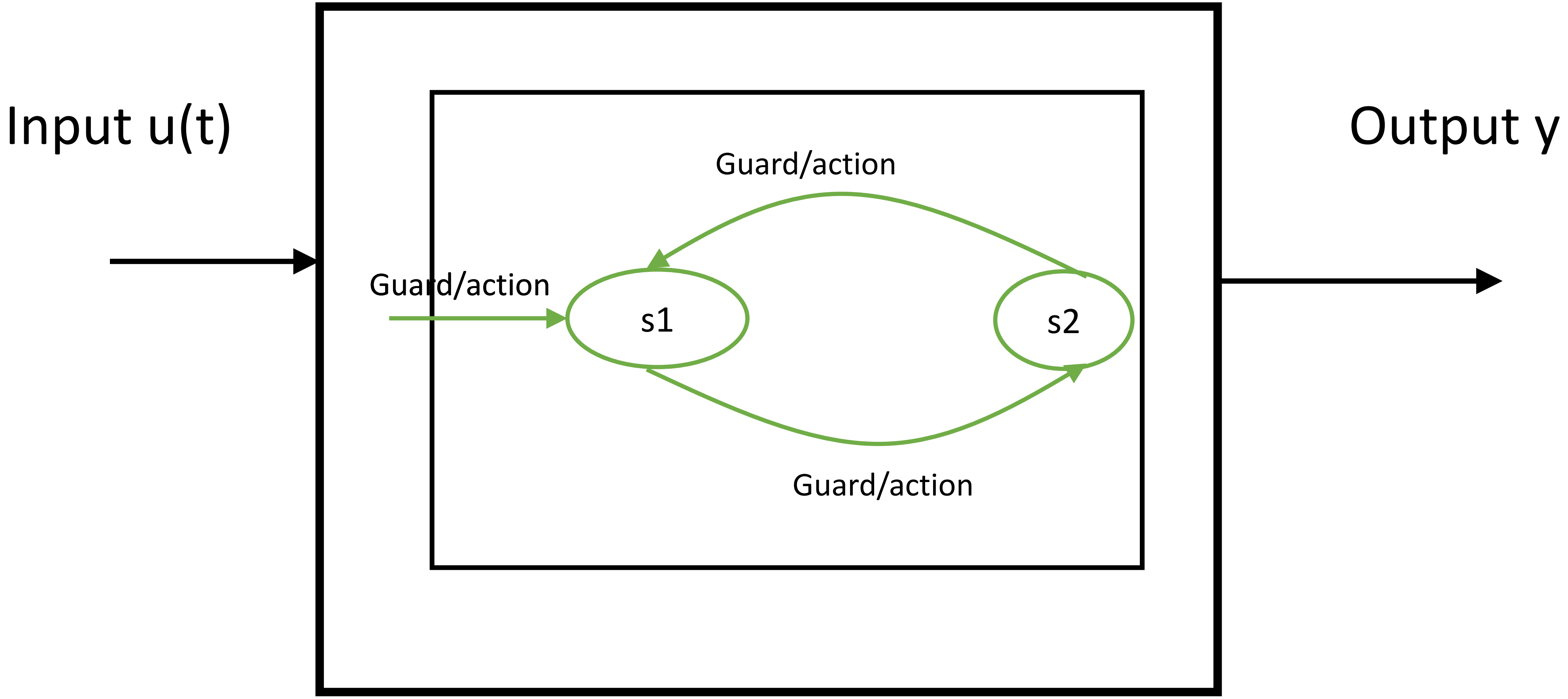
A box, where the inputs and the outputs are functions $S: u \to y$



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

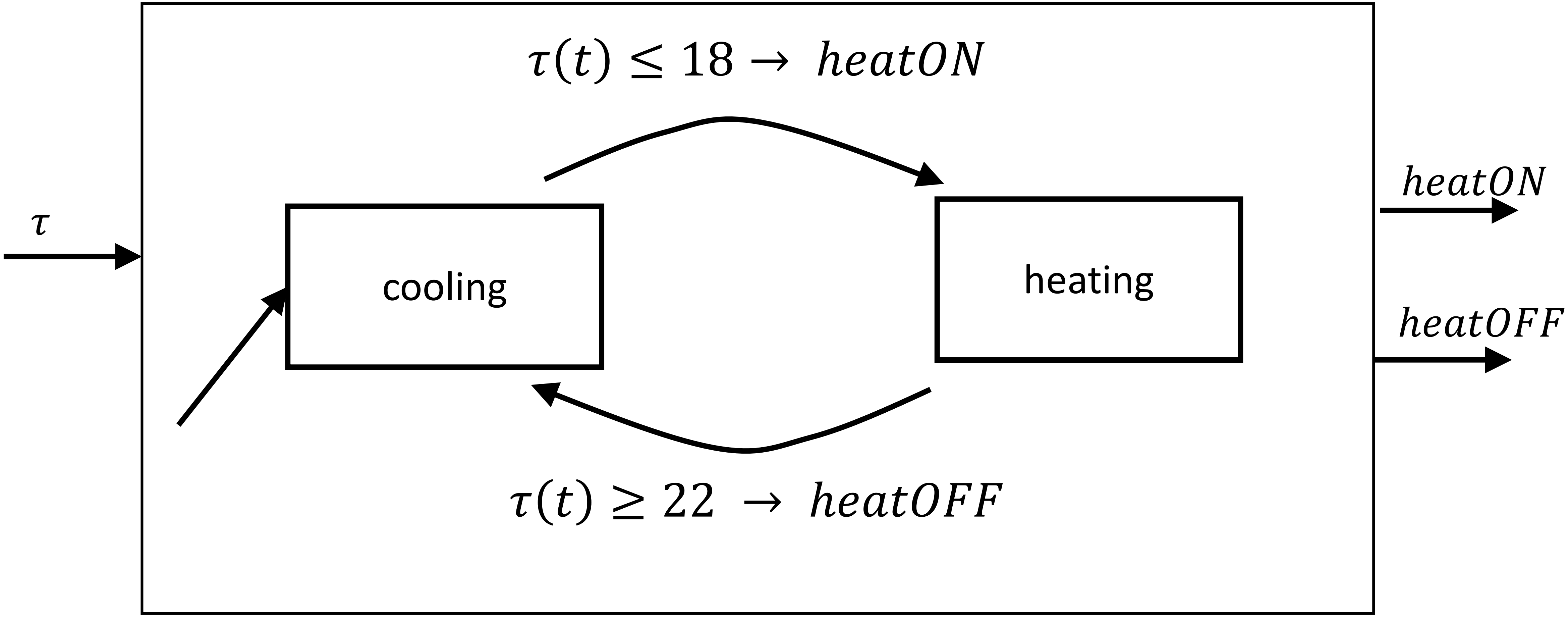
We have so far assumed that state machines operate in a sequence of discrete reactions.
We have assumed that inputs and outputs are absent between reactions.

Having continuous inputs



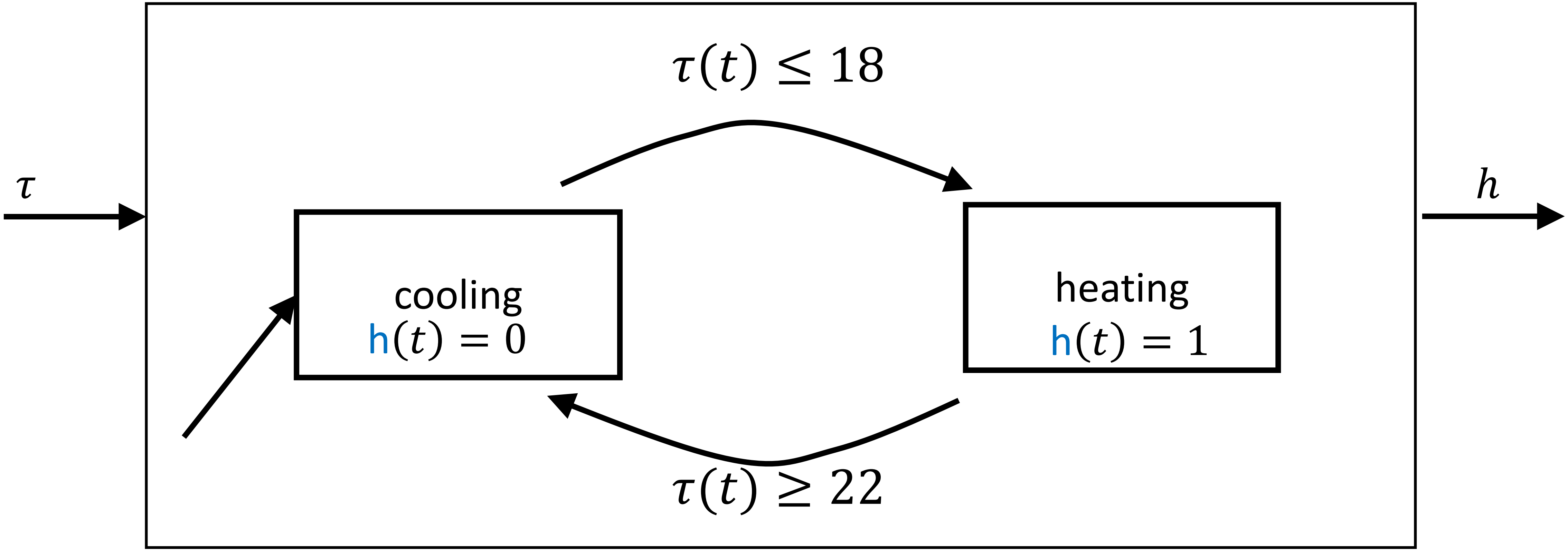
We will define a transition to occur when a guard on an outgoing transition from the current state becomes enabled

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.

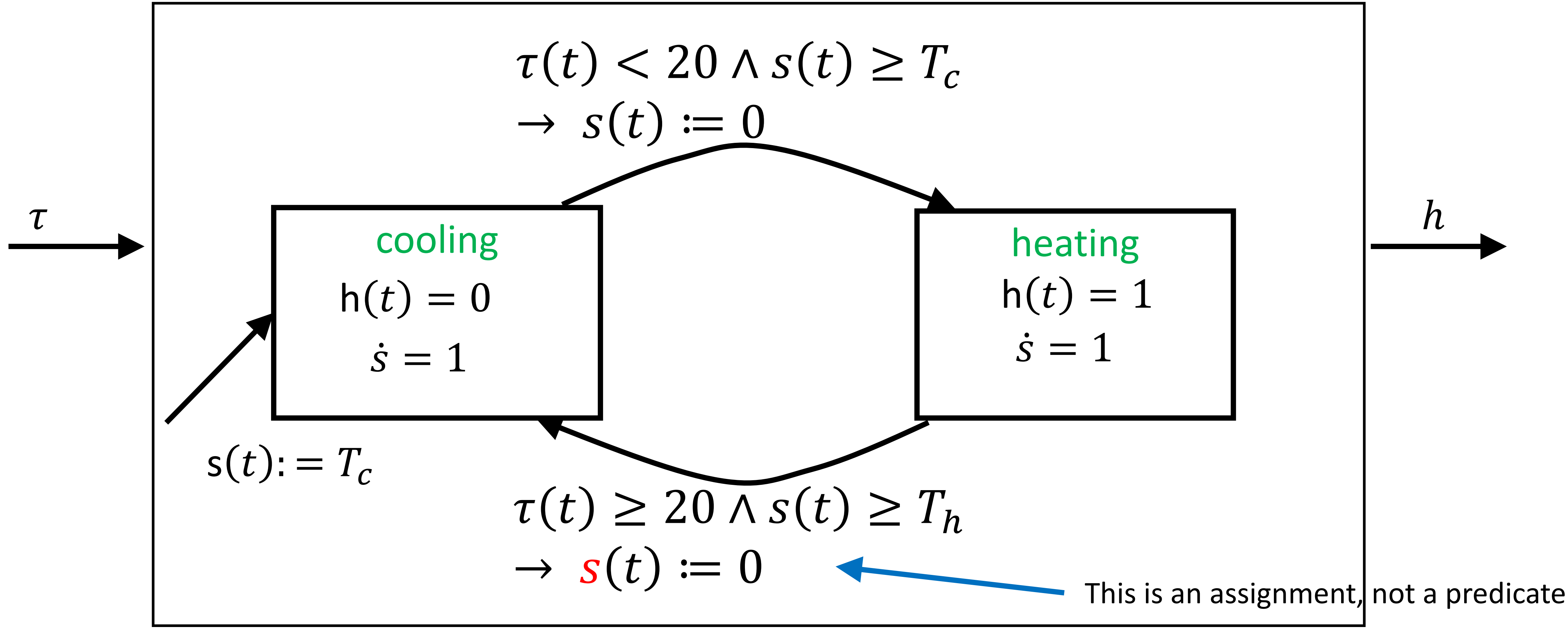
Modal Models

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.

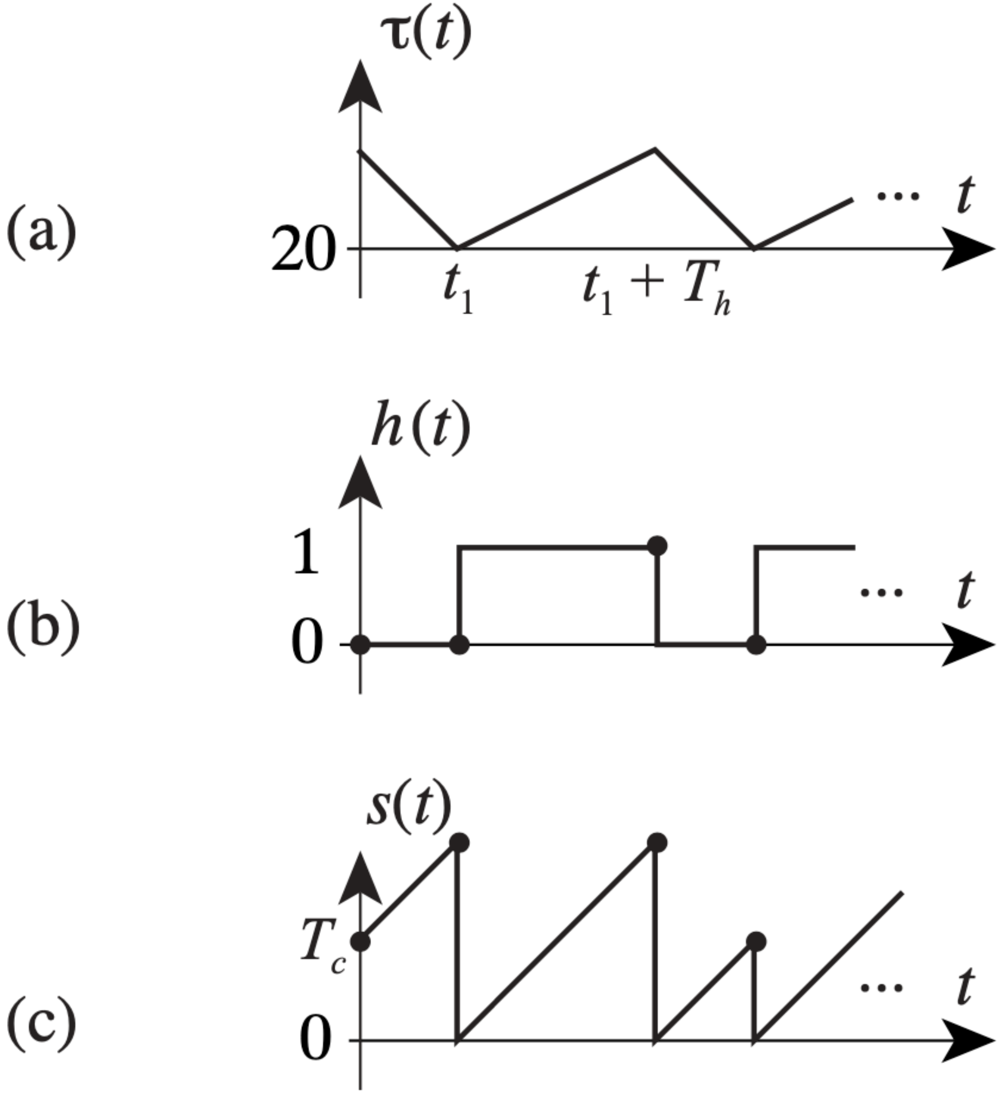
Timed Automata

- Introduced by Alur and Dill (A theory of timed Automata, TCS,1994)
- They are the simplest non-trivial hybrid systems
- All they do is measuring the passage of time
- A **clock** $s(t)$ is modeled by a first-ODE: $\dot{s} = a \quad \forall t \in T_m$
  where $s : \mathbb{R} \rightarrow \mathbb{R}$ is a continuous-time signal,
  $s(t)$ is the value of the clock at time t, and
  $T_m \subset \mathbb{R}$ is the subset of time during which the hybrid system is in mode m.
  The rate of the clock, a, is a constant while the system is in this mode.

Timed Automata



cooling and heating are discrete states, s is a continuous state

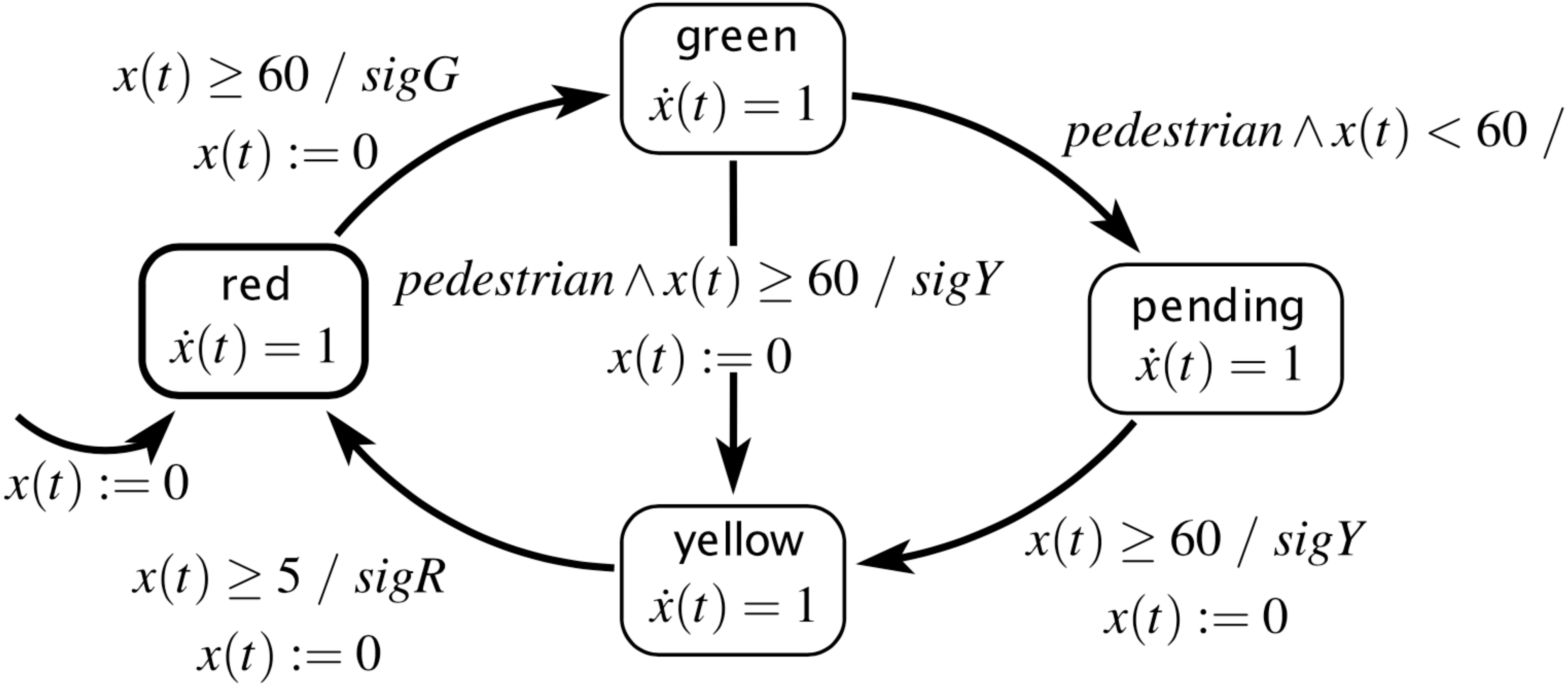


Temperature input $\tau(t)$

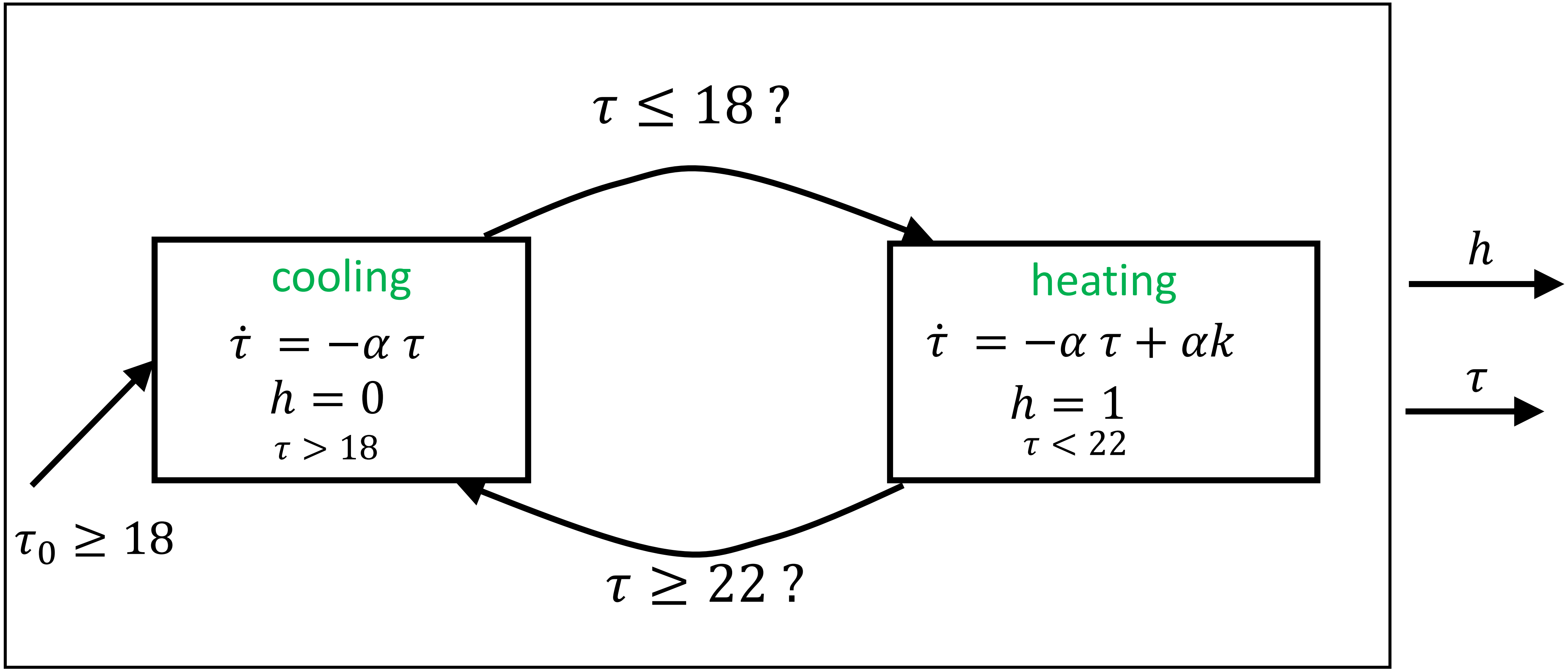
The output h

The refinement state s.

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



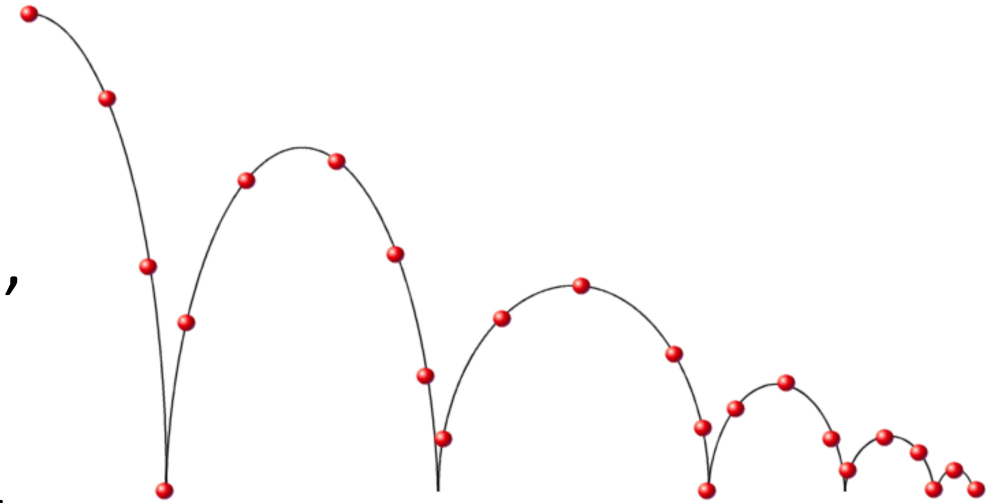
Hybrid Automata



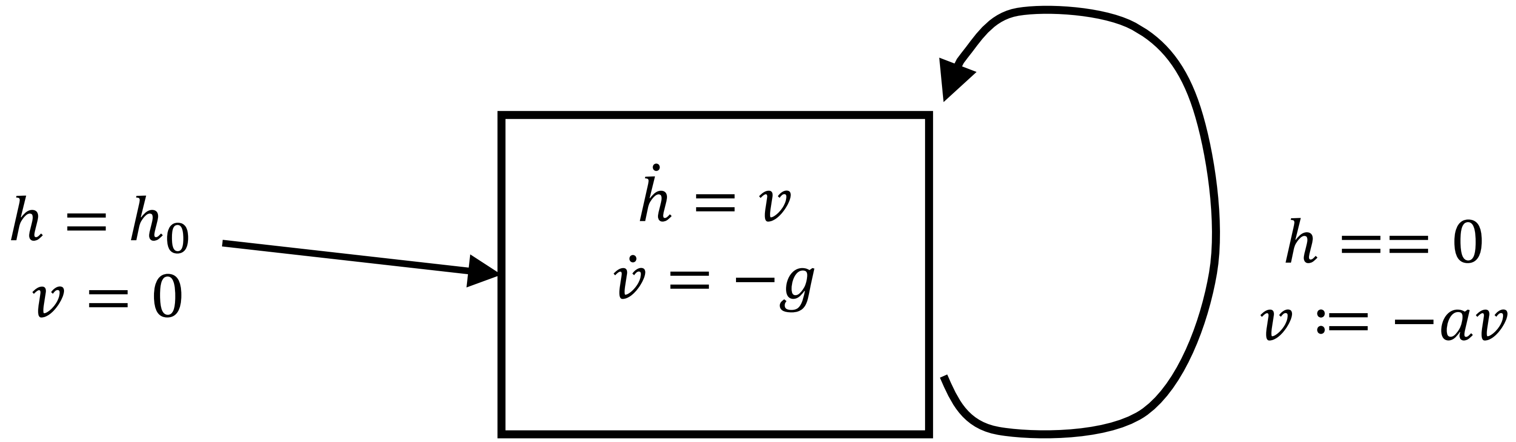
- Generalization of a timed process
- Instead of timed transitions, we can have arbitrary evolution of state/output variables, typically specified using differential equations

Modeling a bouncing ball

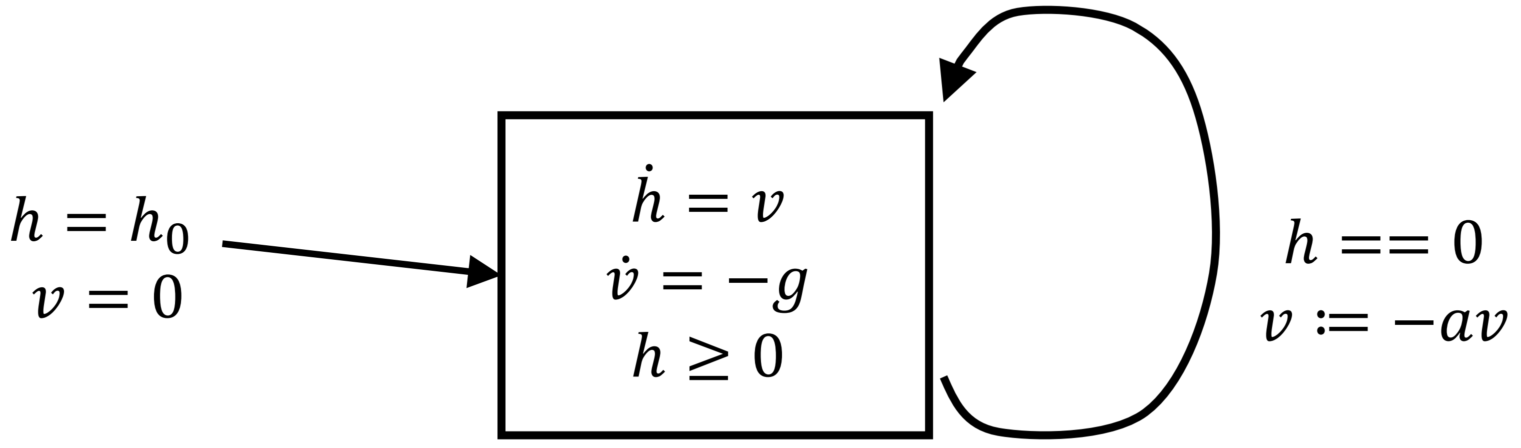
- Ball dropped from an initial height of h_0 with an initial velocity of v_0
- Velocity changes according to $\dot{v} = -g$
- When ball hits the ground, i.e. when $h(t) = 0$, velocity changes discretely from negative (downward) to positive (upward)
  - I.e. $v(t) := -av(t)$, where a is a damping constant
- we can model it as a hybrid system!



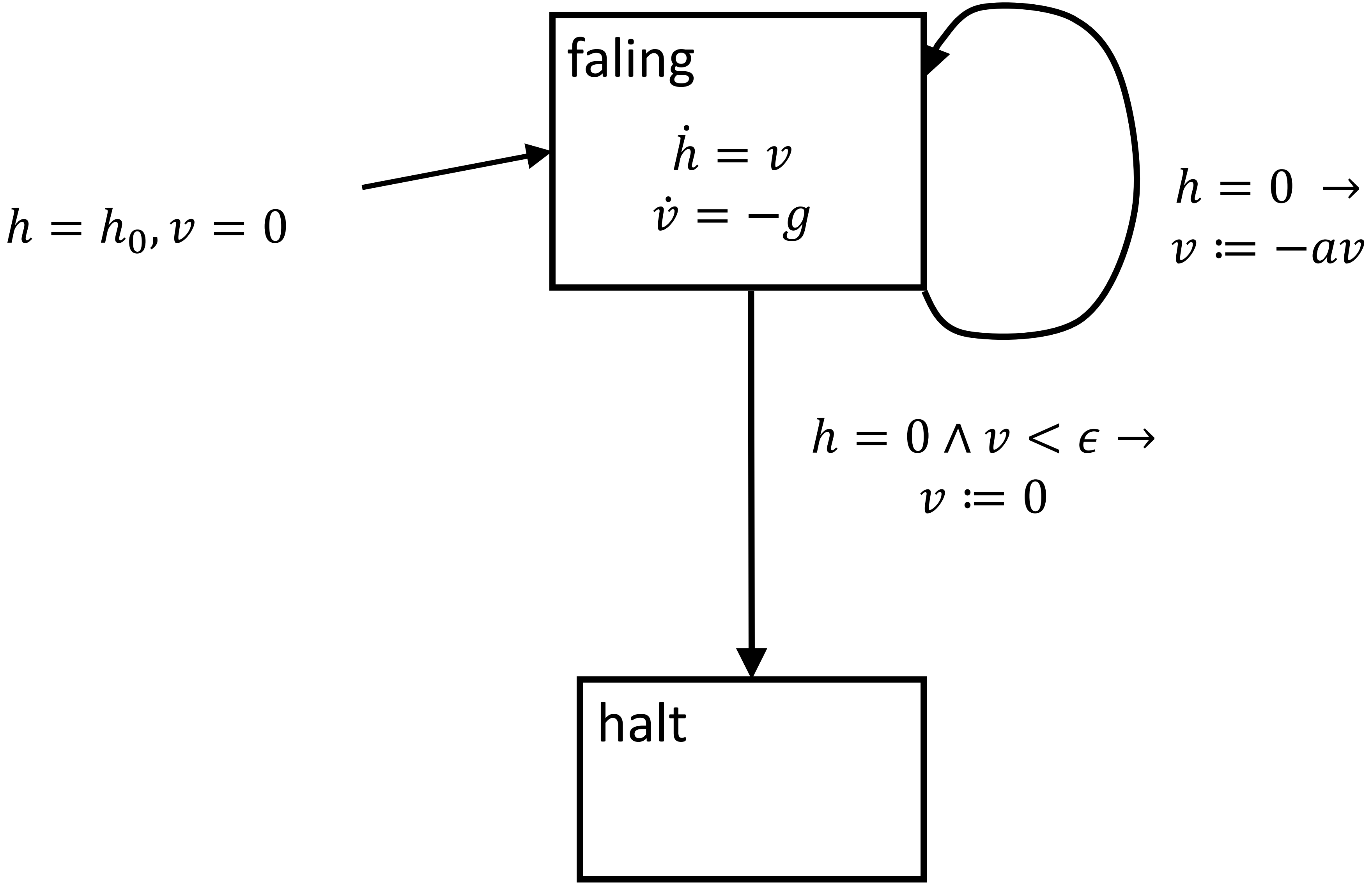
Hybrid Process for Bouncing ball



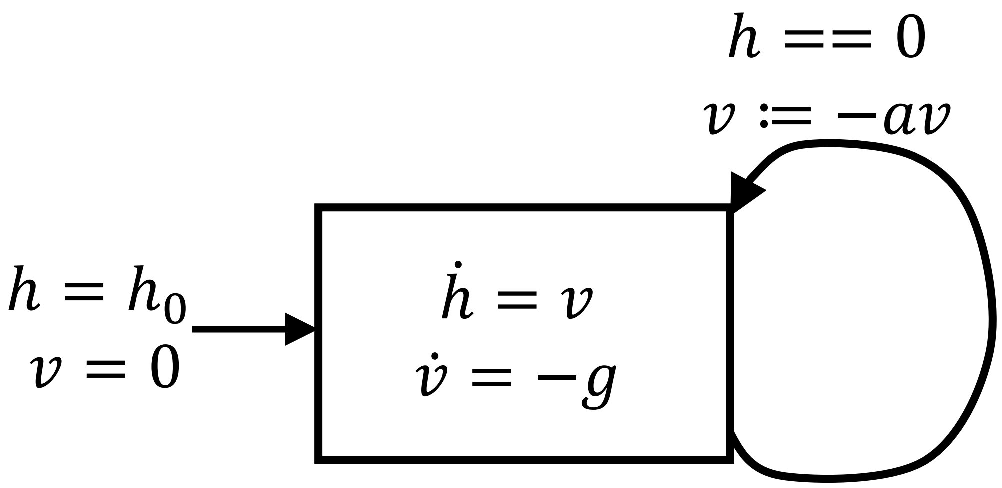
Hybrid Process for Bouncing ball

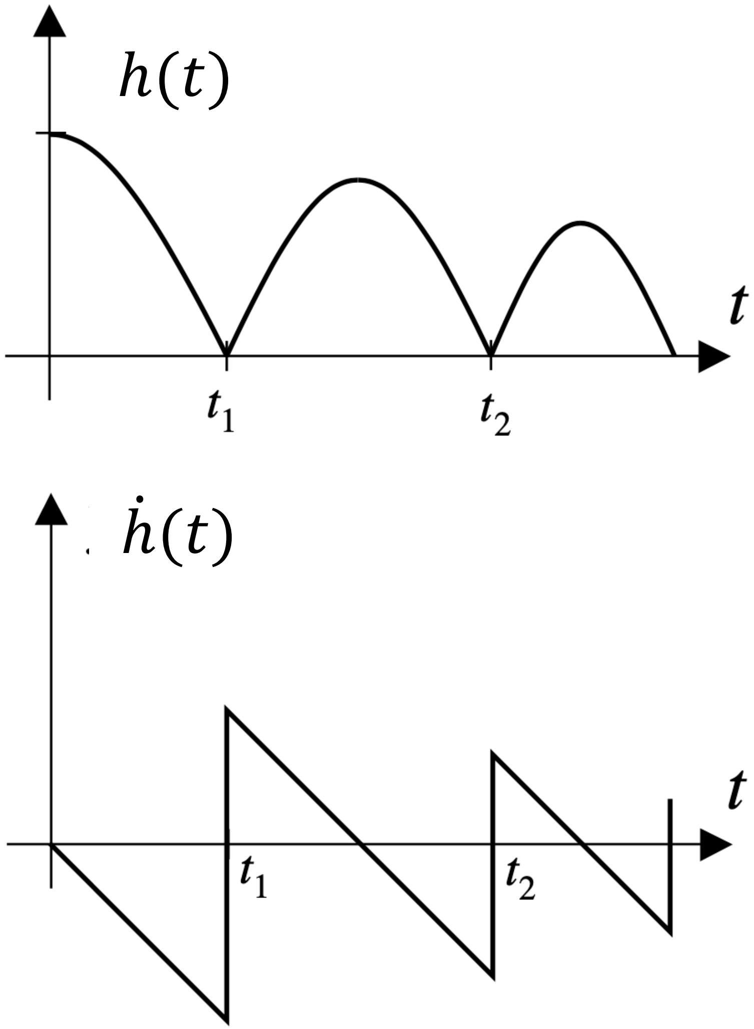


Non-Zeno hybrid process for bouncing ball



Hybrid Process for Bouncing ball





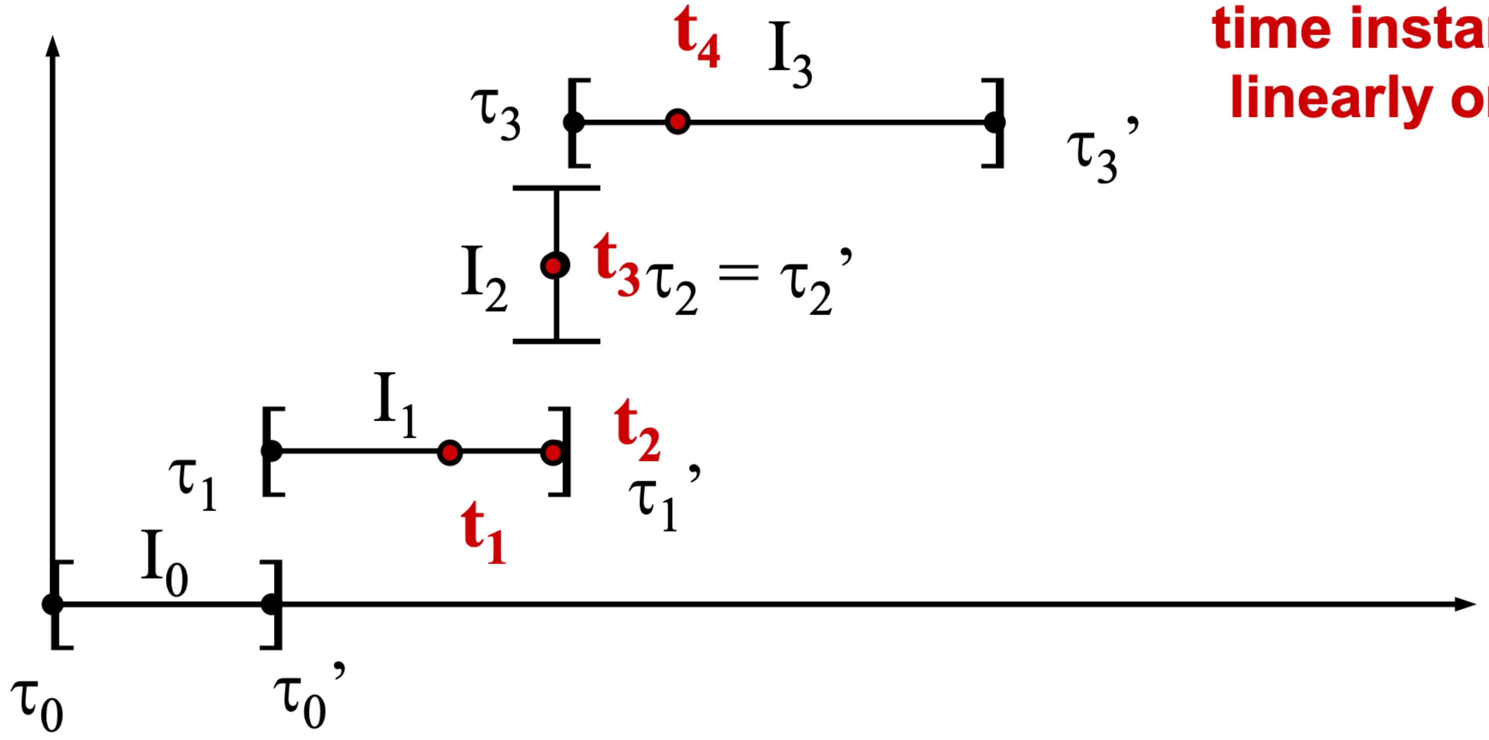
What happens as $h \to 0$?

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<∞
- $\tau_i' = \tau_{i+1}$
- $\tau_i \leq \tau_i'$



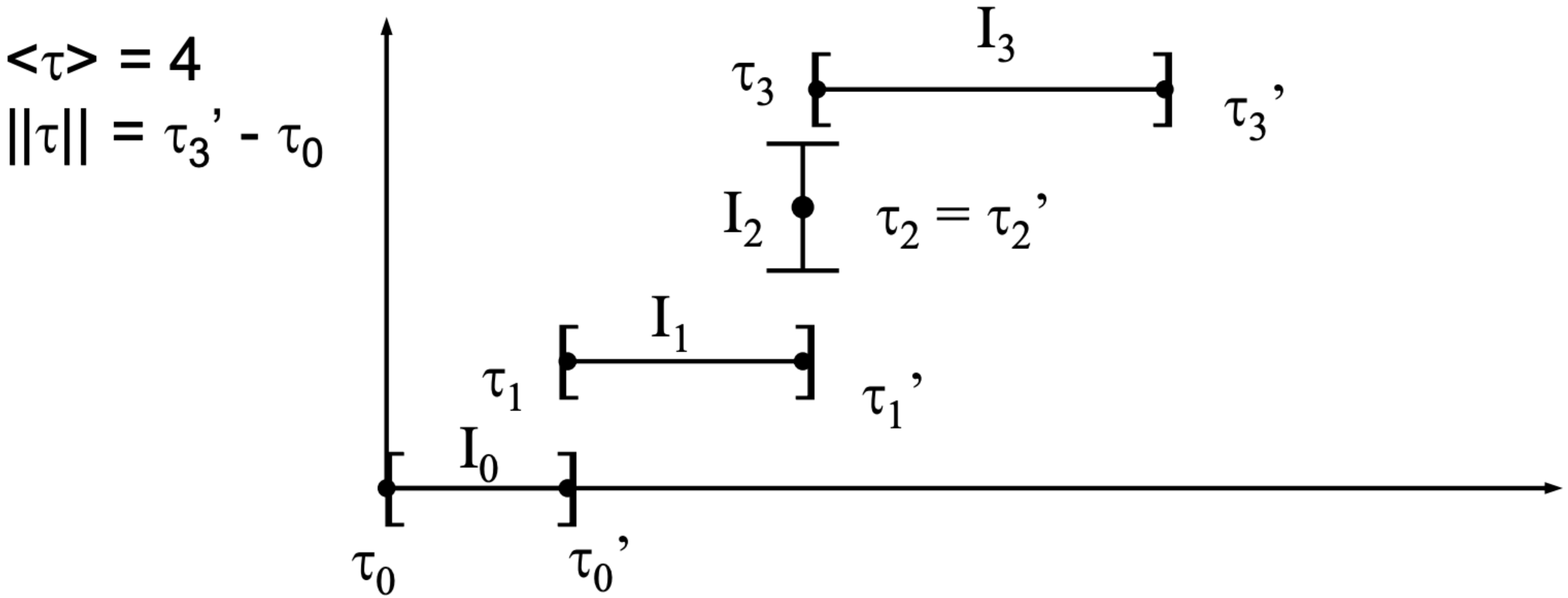
$t_1 \prec t_2 \prec t_3 \prec t_4$

time instants in τ are linearly ordered

Hybrid Time Set: Length

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

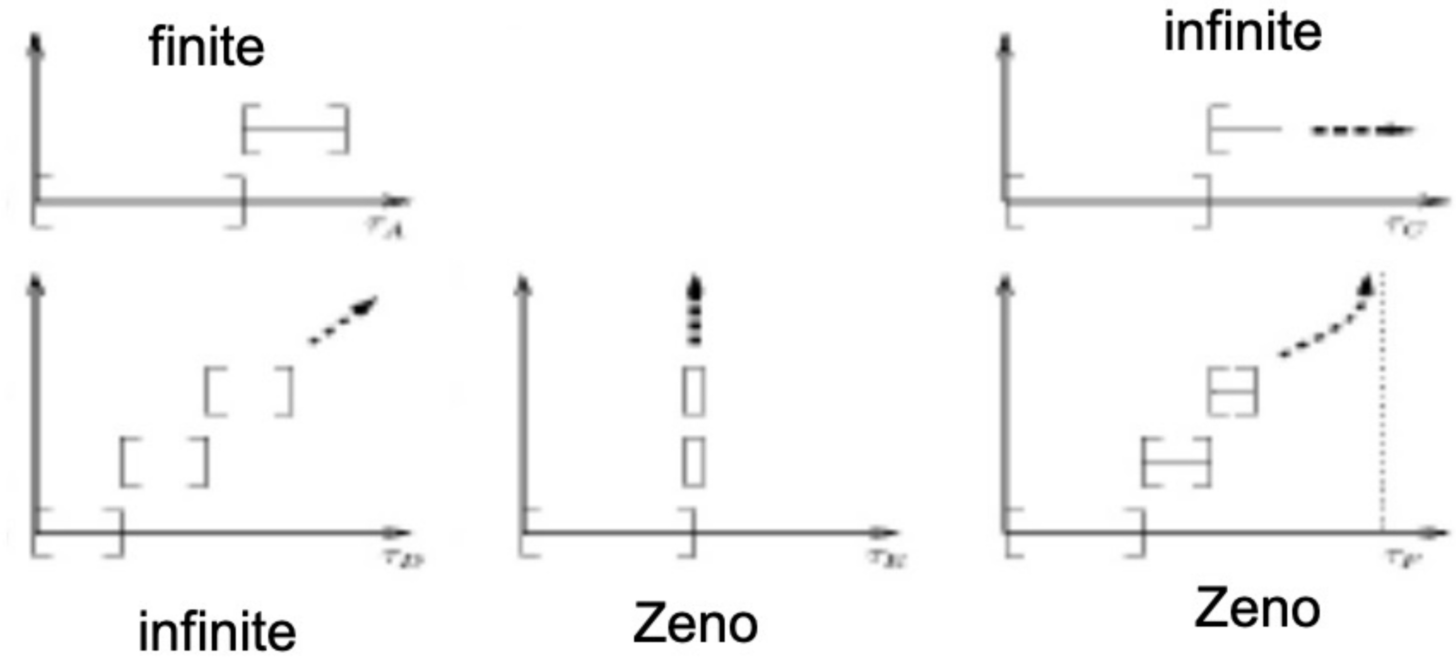
- Discrete extent: $< \tau > = M + 1$ number of discrete transition
- Continuous extent: $||\tau|| = \sum_{i=0}^{M} |\tau_i' - \tau_i|$ total duration of interval in τ



Hybrid Time Set: Classification

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

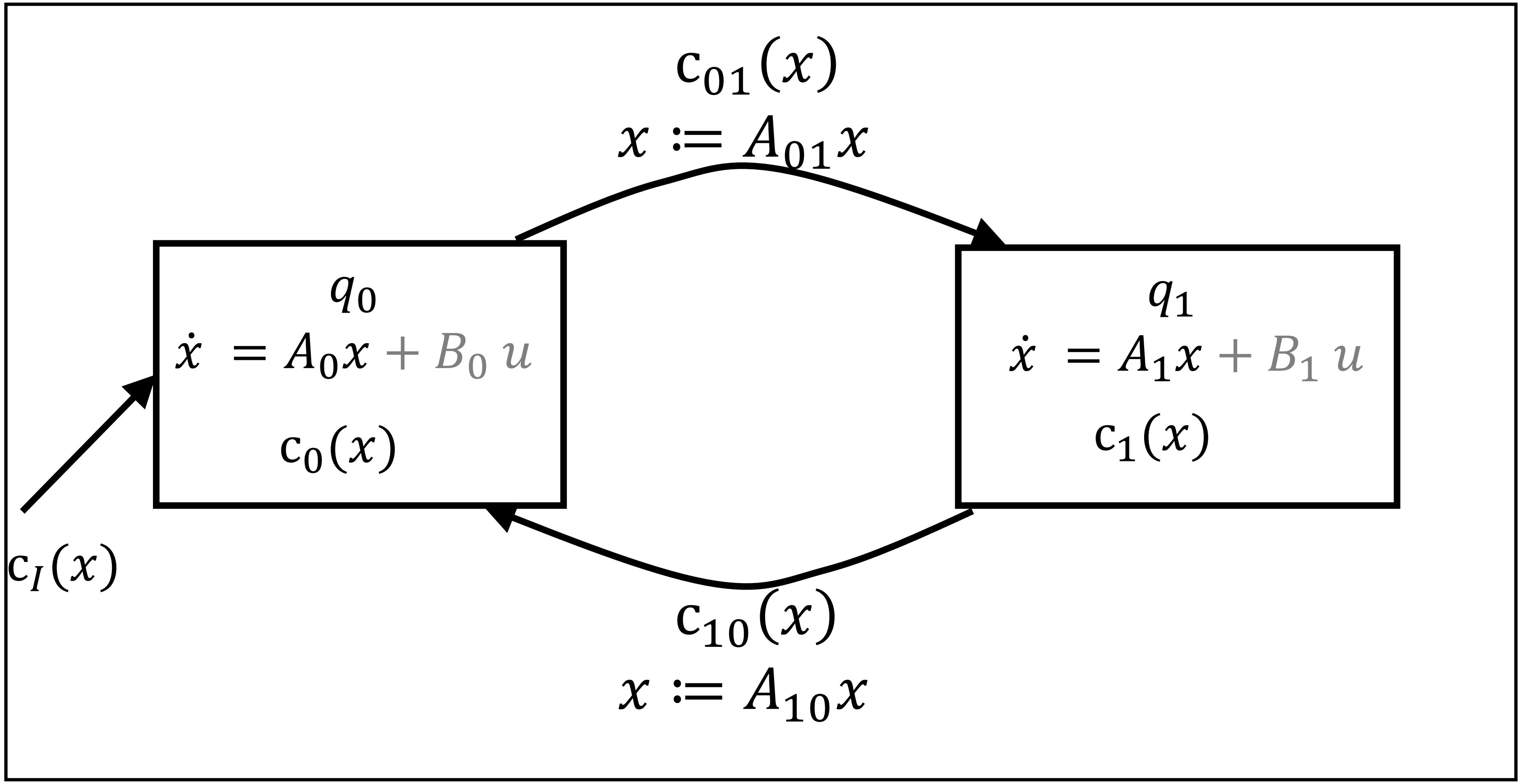
- Finite: if $< \tau >$ is finite and $\mathrm{I}_M = [\tau_M, \tau_M']$
- Infinite:if $||\tau||$ is infinite
- Zeno: if $< \tau >$ is infinite but $||\tau||$ is finite



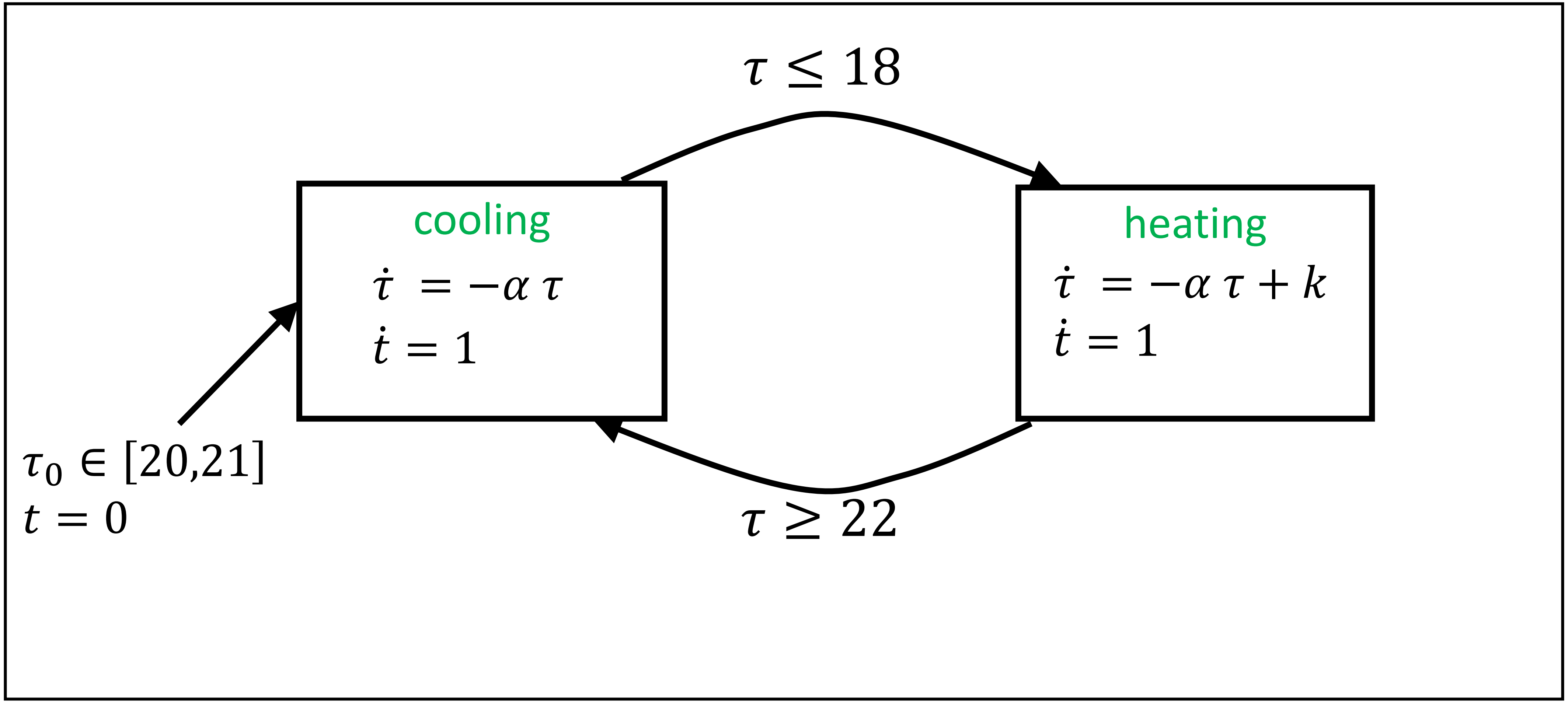
Zeno's Paradox

- 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!

(Linear) Hybrid Automata



(Linear) Hybrid Automata



Hybrid actions/transitions

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

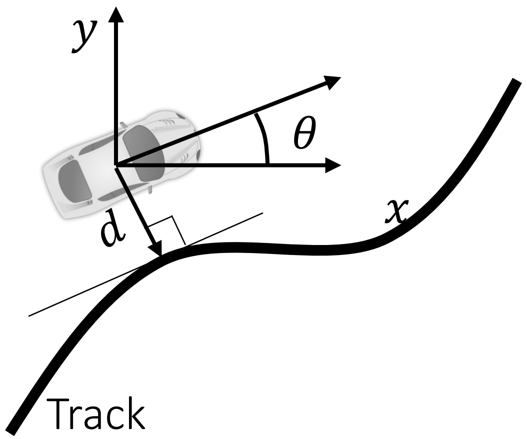
- Continuous action/transition:
  - Discrete mode m does not change
  - $\mathbf{x}_{\boldsymbol{\tau}} = \mathbf{x}(0)$
  - $\frac{d\mathbf{x}(t)}{dt}$ satisfies the given dynamical equation for mode m
  - Output $\mathbf{y}$ satisfies the output equation for mode m: $\mathbf{y}(t) = h_q(\mathbf{x}(t), \mathbf{u}(t))$
  - At all times $t \in [0, \delta]$, the state $\mathbf{x}(t)$ satisfies the invariant for mode m

Hybrid actions/transitions

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

- Discrete action/transition:
  - 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′Output will change from $h_q(\mathbf{x}_\tau)$ to $h_{q'}\left(r(\mathbf{x}_\tau)\right)$

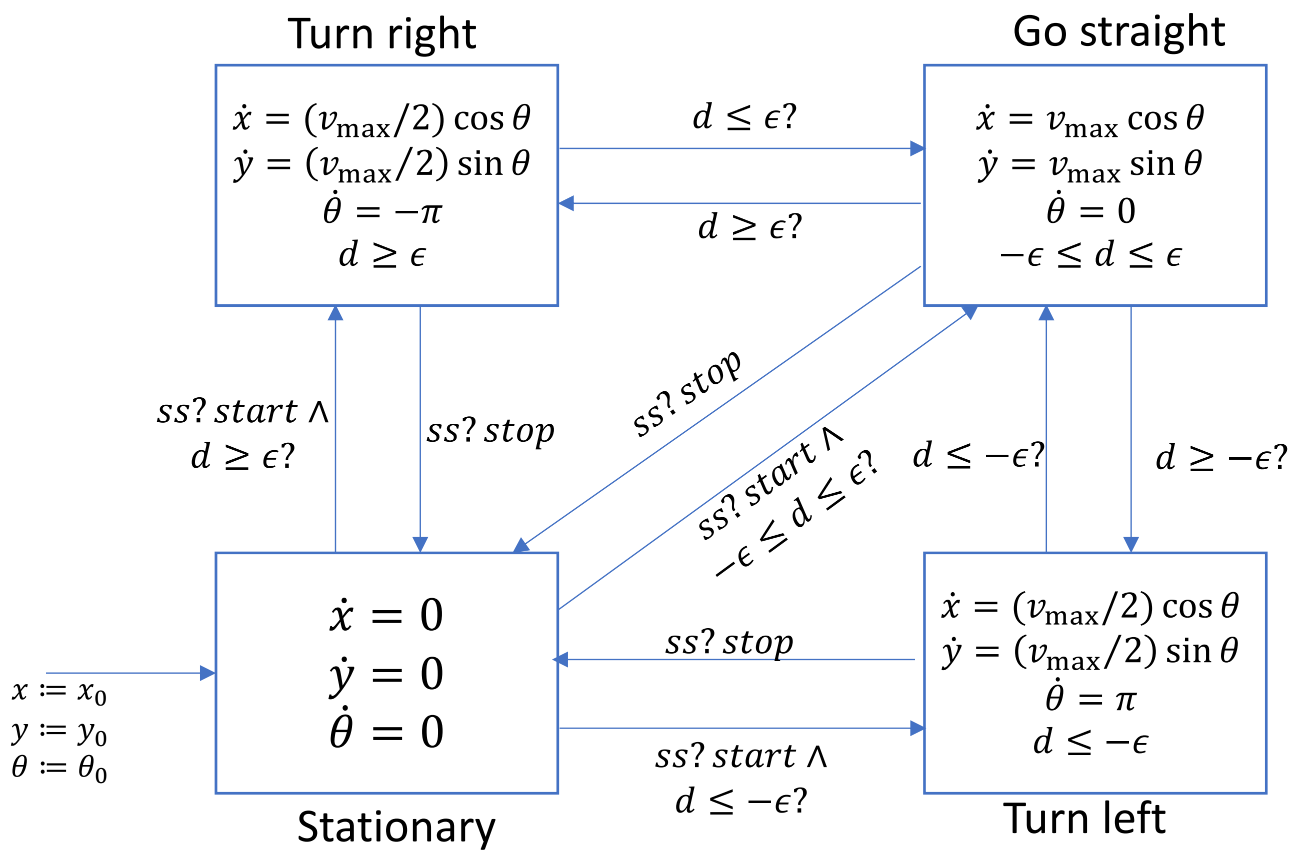
Design Application: Autonomous Guided Vehicle

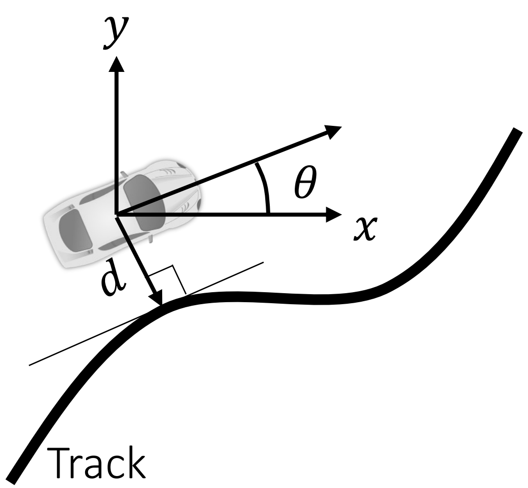


When $d \in [-\epsilon, +\epsilon]$, controller decides that vehicle goes straight, otherwise executes a turn command to bring error back in the interval

- Objective: Steer vehicle to follow a given track
- Control inputs: linear speed (v), angular speed (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 $v = \frac{v_{\max}}{2}$

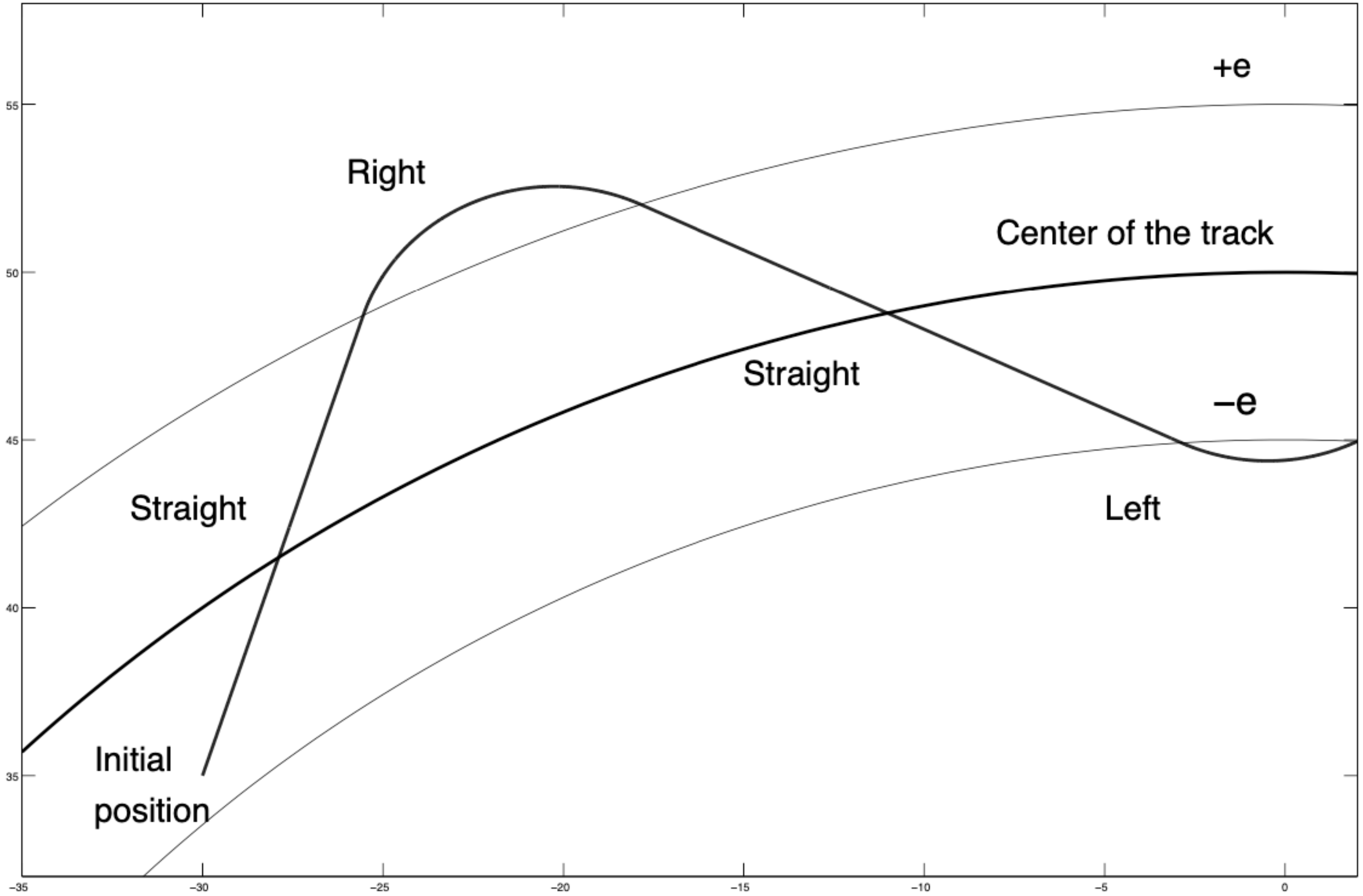
On/Off control for Path following





Inputs: ss $\in \{stop, start\}, d \in \mathbb{R}$

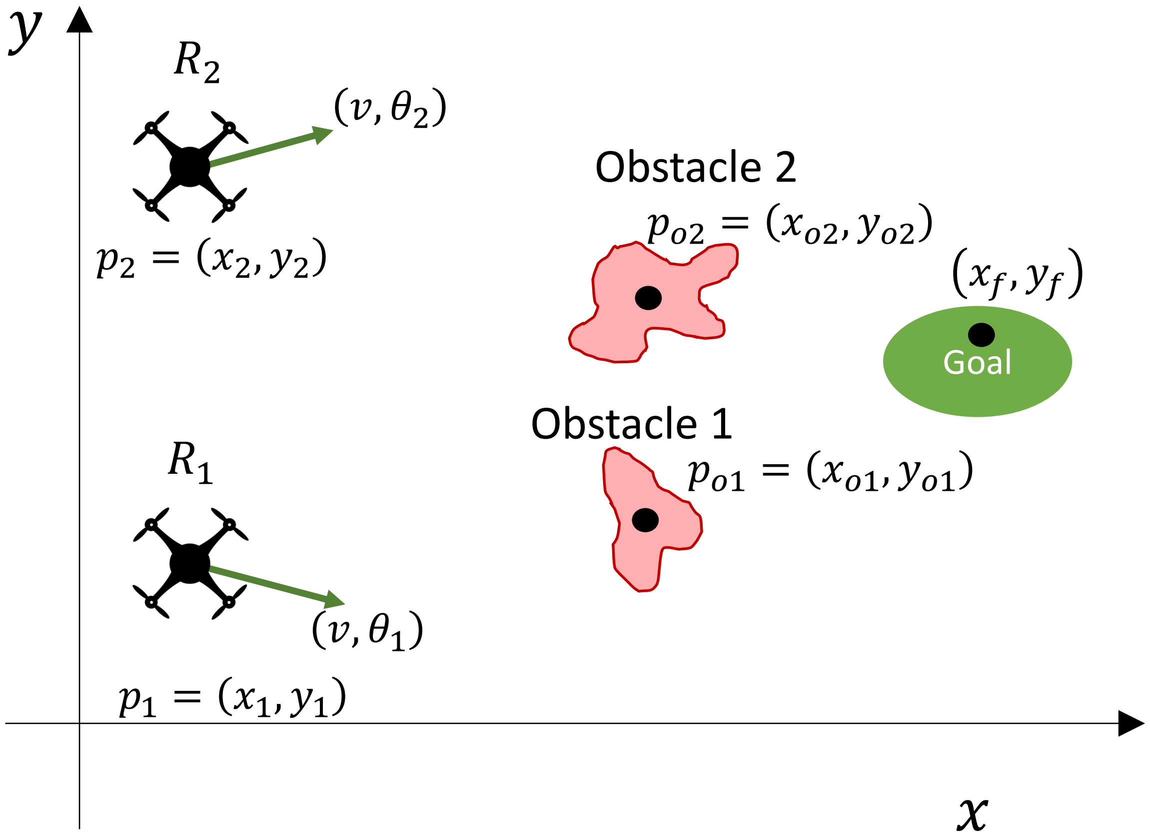
On/Off control for Path following



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



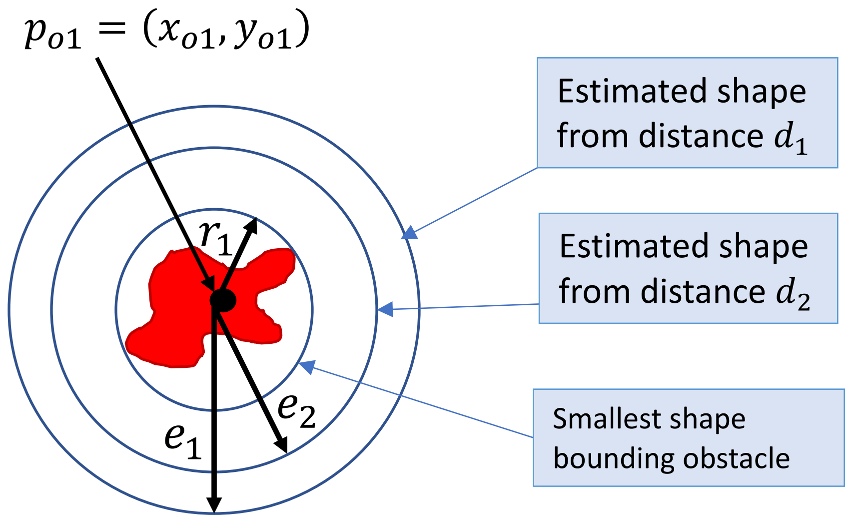
- Assumptions:
  - Two-dimensional world
  - Robots are just points
  - Each robot travels with a fixed speed
- Dynamics for Robot R_i:
  - $\dot{x}_i = v \cos \theta_i ; \dot{y}_i = v \sin \theta_i$
- Design objectives:
  - Eventually reach (x_f, y_f)
  - Always avoid Obstacle1 and Obstacle 2
  - Minimize 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



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

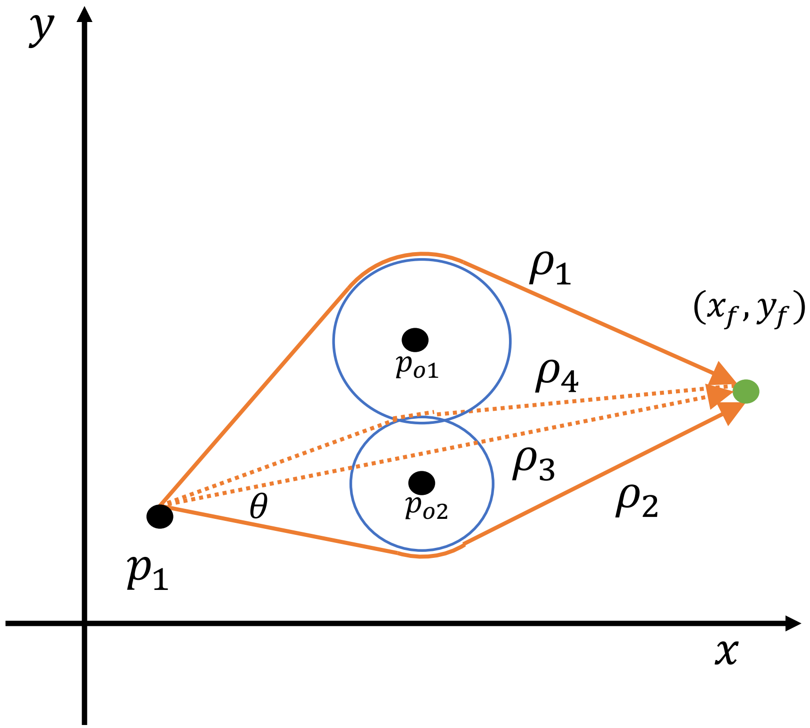
- Robot R_1 maintains radii e and e' that are estimates of obstacle sizes
- Every τ seconds, R_1 executes following update to get estimates of shapes of each obstacle:

$$e_1 \coloneqq \min\left(e_1, r_1 + a(\|p_1 - p_{o1}\| - r_1)\right)$$

- Computation of R_2 is symmetric

$$e_2 \coloneqq \min\left(e_2, r_2 + a(\|p_1 - p_{o2}\| - r_2)\right)$$

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

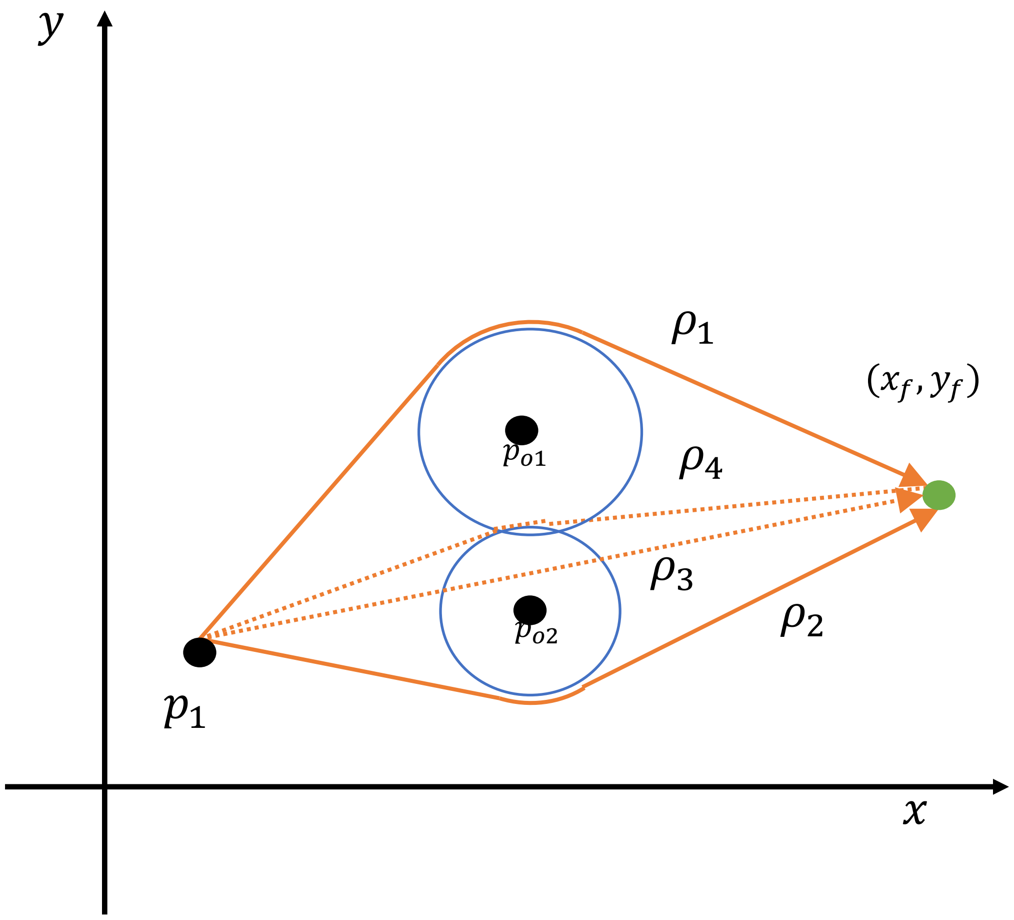
Dynamic path planning

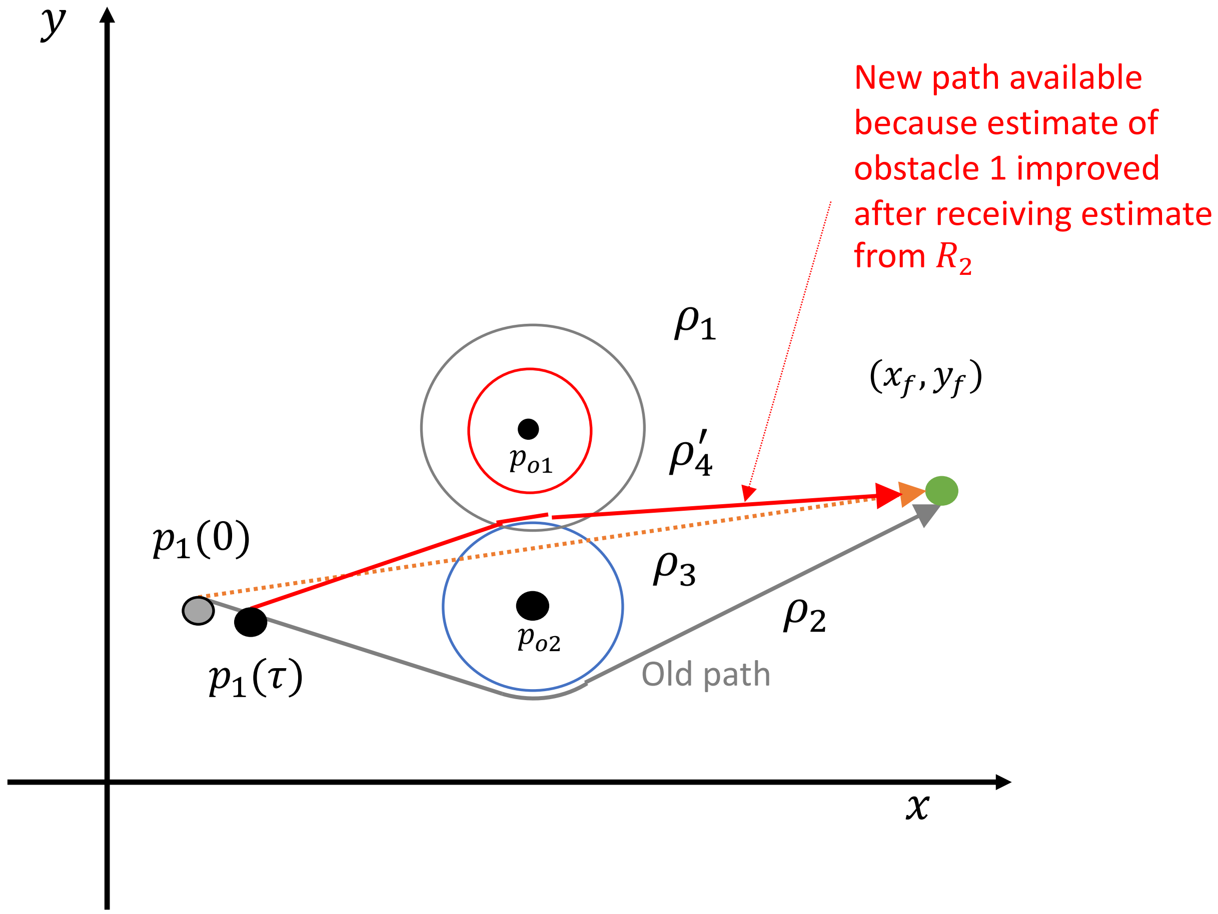
- Path planning inputs:
  - Current position of robot
  - Target position
  - Position of obstacles and estimates
- Output:
  - Direction for motion assuming obstacle estimates are correct
- May be useful to execute planning algorithm again as robot moves!
  - Because estimates will improve closer to 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