Cyber Physical Systems - Hybrid Control

Lecture 2: Introduction: Hybrid Automata,

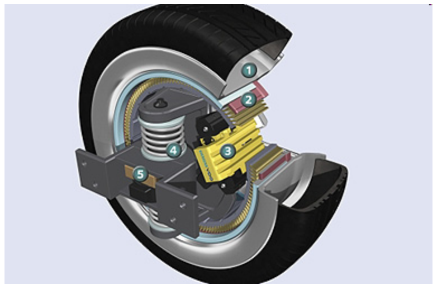
Finite-State Machines (FSM)

Andreas Podelski

Literature:
Edward A. Lee & Sanjit A. Seshia,
Introduction to Embedded Systems
leeseshia.org

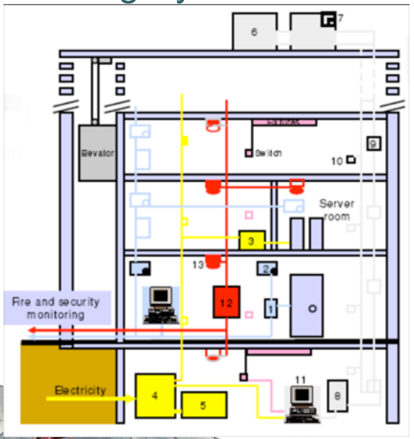
Cyber-Physical Systems

Automotive



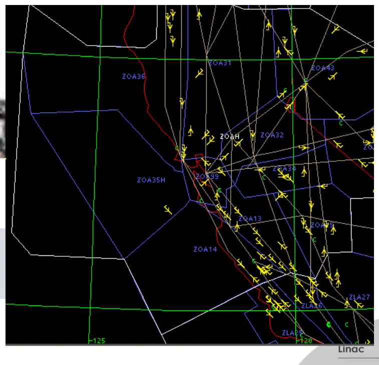
E-Corner, Siemens

Building Systems

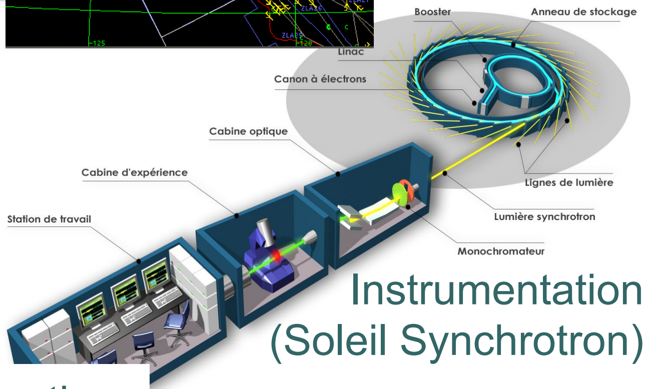


Avionics

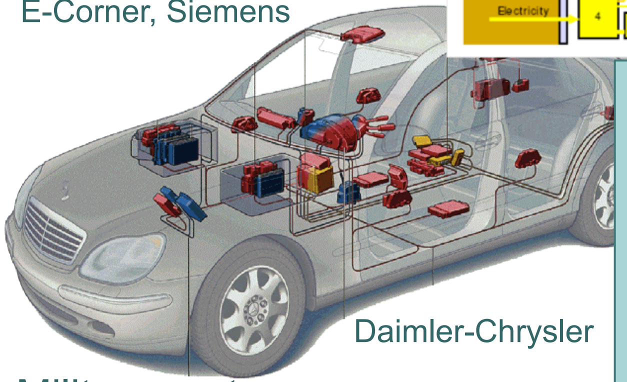
Telecommunications



Transportation (Air traffic control at SFO)



Instrumentation (Soleil Synchrotron)

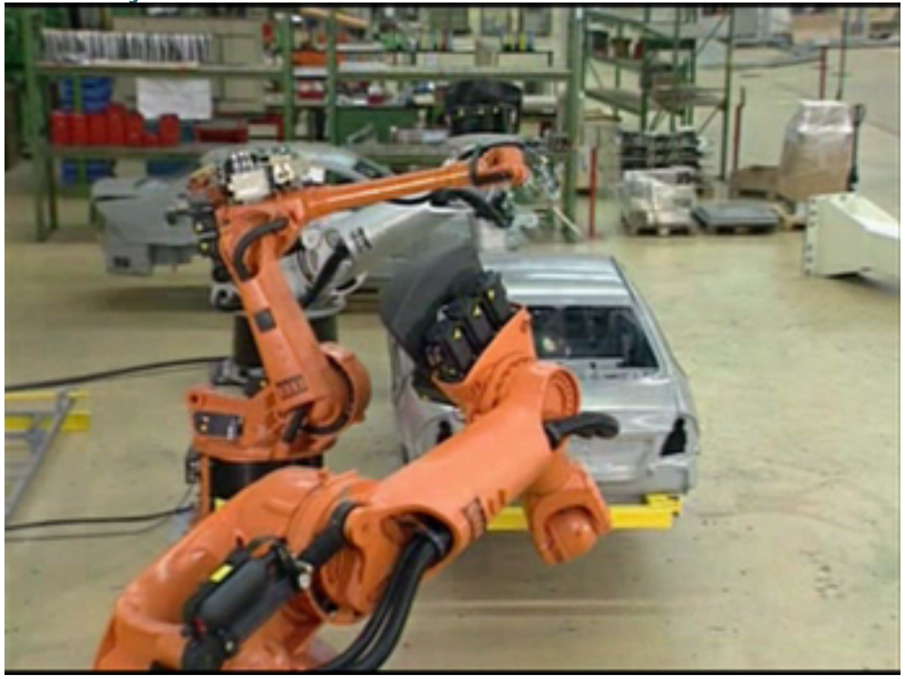


Daimler-Chrysler

Power generation and distribution

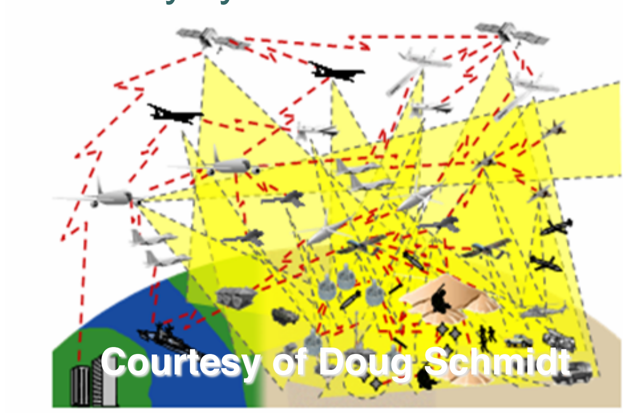
Courtesy of General Electric

Factory automation



Courtesy of Kuka Robotics Corp.

Military systems:



Courtesy of Doug Schmidt

Cyber-Physical Systems (CPS)

networked

computational resources

interacting with physical systems

CPS vs. Embedded Systems

embedded systems view:

software on small computers => limited resources

technical problem: extract performance

CPS view:

computation and networking integrated with physical processes

technical problem: manage dynamics, time, and concurrency

Fundamental problems

time matters

- “as fast as possible” is not good enough

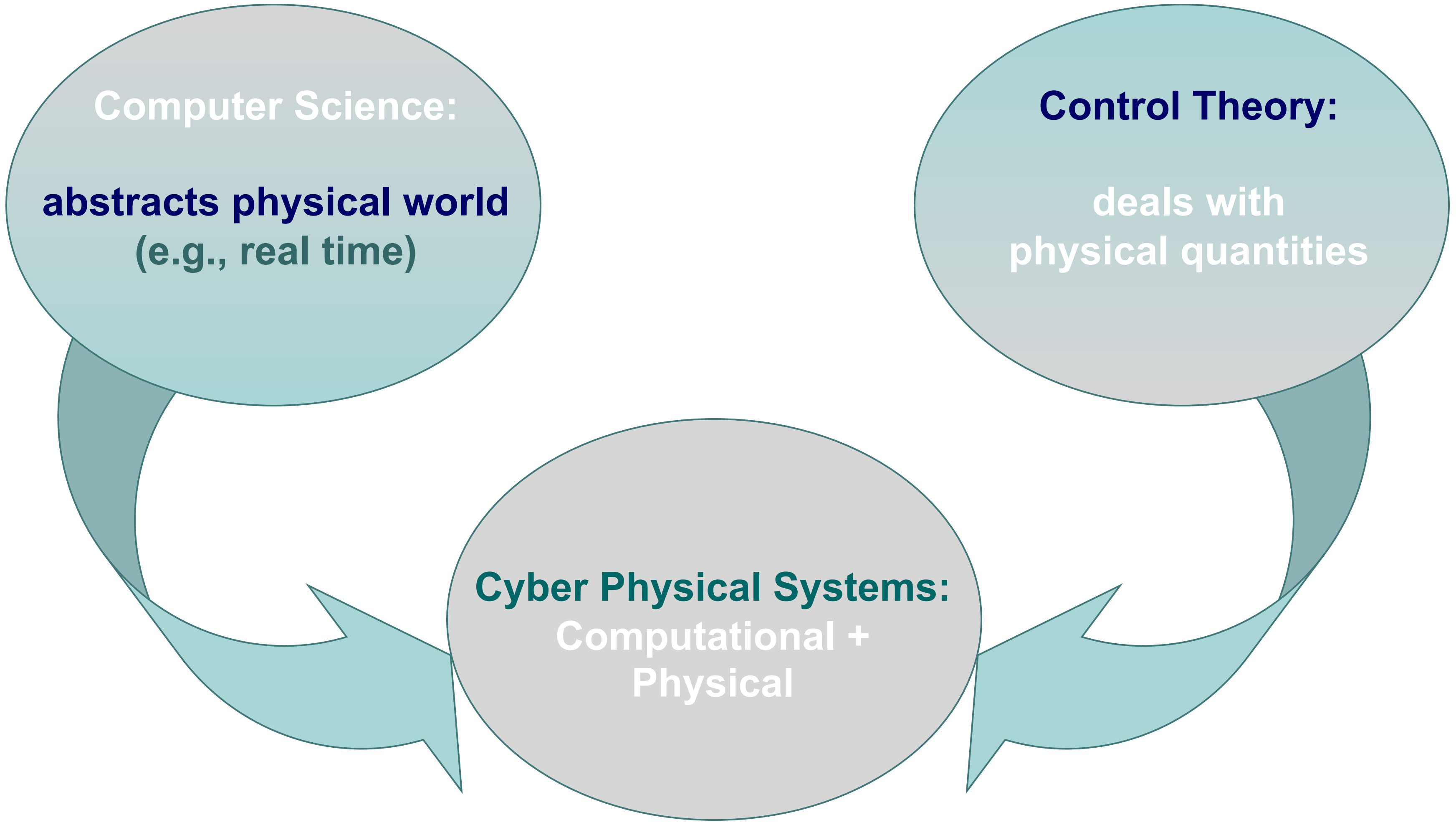
concurrency is intrinsic

- it’s not an illusion (as in time sharing), and
- it’s not (necessarily) about exploiting parallelism

environment is physical

- behavior obeys physical laws
- depends on continuous variables
  (force, acceleration, speed, position)

CPS is Multidisciplinary



First Challenge

Models for the physical world and for computation diverge.

- physical: time continuum, differential equations, dynamics
- computational: algorithm, procedure, state transitions, logic

bridge the gap:

- use physical world models to specify behavior of systems
- use computational view to study dynamics of physical system

Model

artifact that imitates the system

mathematical model:

- definitions in terms of mathematical formulas
- mathematical correctness statement
- formal, automatic correctness proof

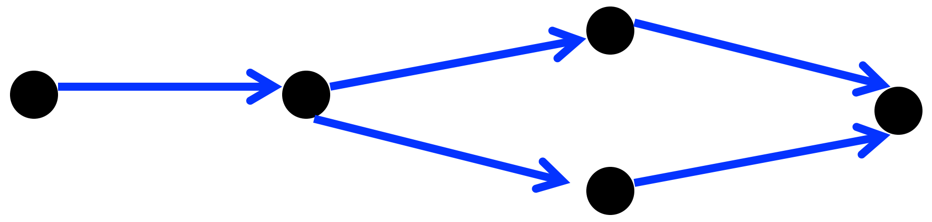
formal: mathematical, logical, machine checkable

automatic: push-button, scalable

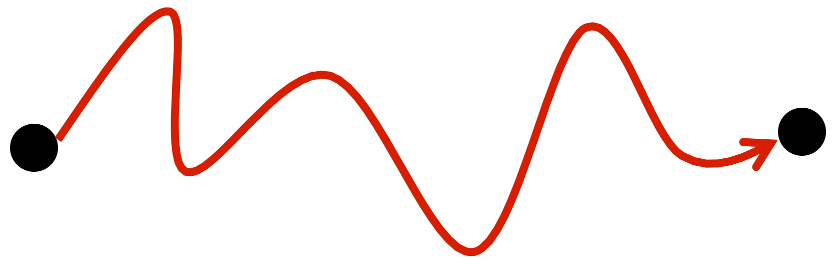
Model = abstraction of **system dynamics**

- physical phenomena:
  differential equations
- computation / discrete mode change:
  finite-state automata
- combination:
  hybrid automata

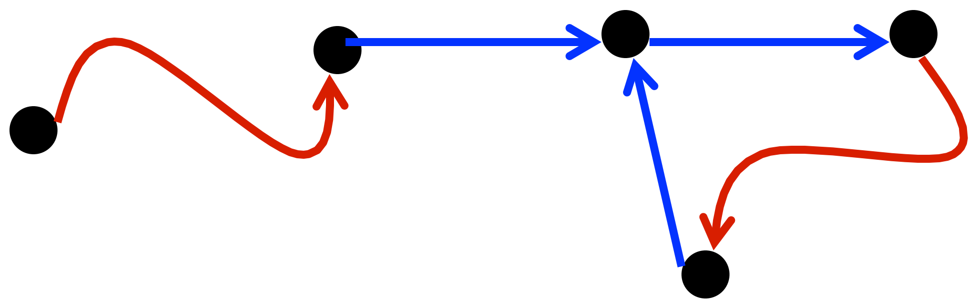
Discrete System (FSM)

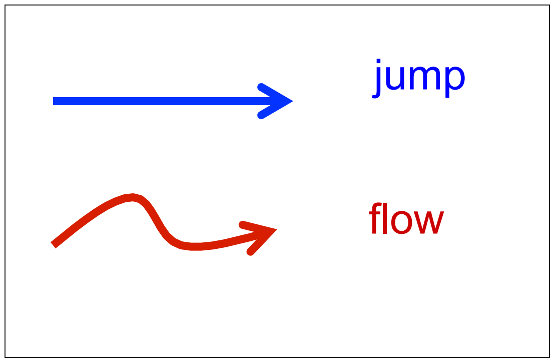


Continuous System



Hybrid System





Next: The *Hybrid Automaton* Model

Thermostat

State has both discrete and continuous components:

$x \in \mathbb{R}$	temperature
$h \in \{\text{on}, \text{off}\}$	heating mode

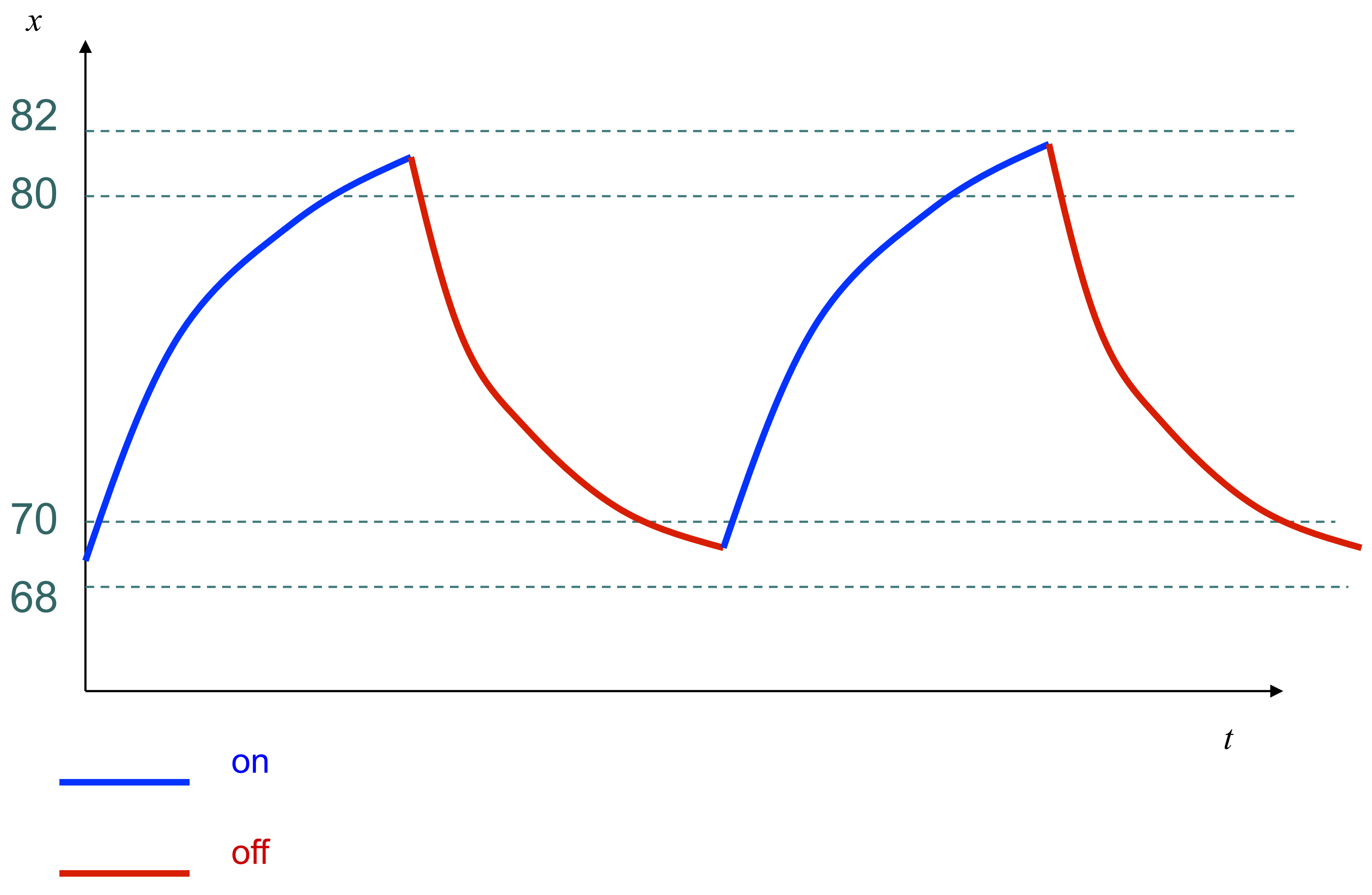
Flow in each mode is:

$$h = \text{on} \wedge x < 82 \qquad \dot{x} = K(100 - x)$$
$$h = \text{off} \wedge x > 68 \qquad \dot{x} = -Kx$$

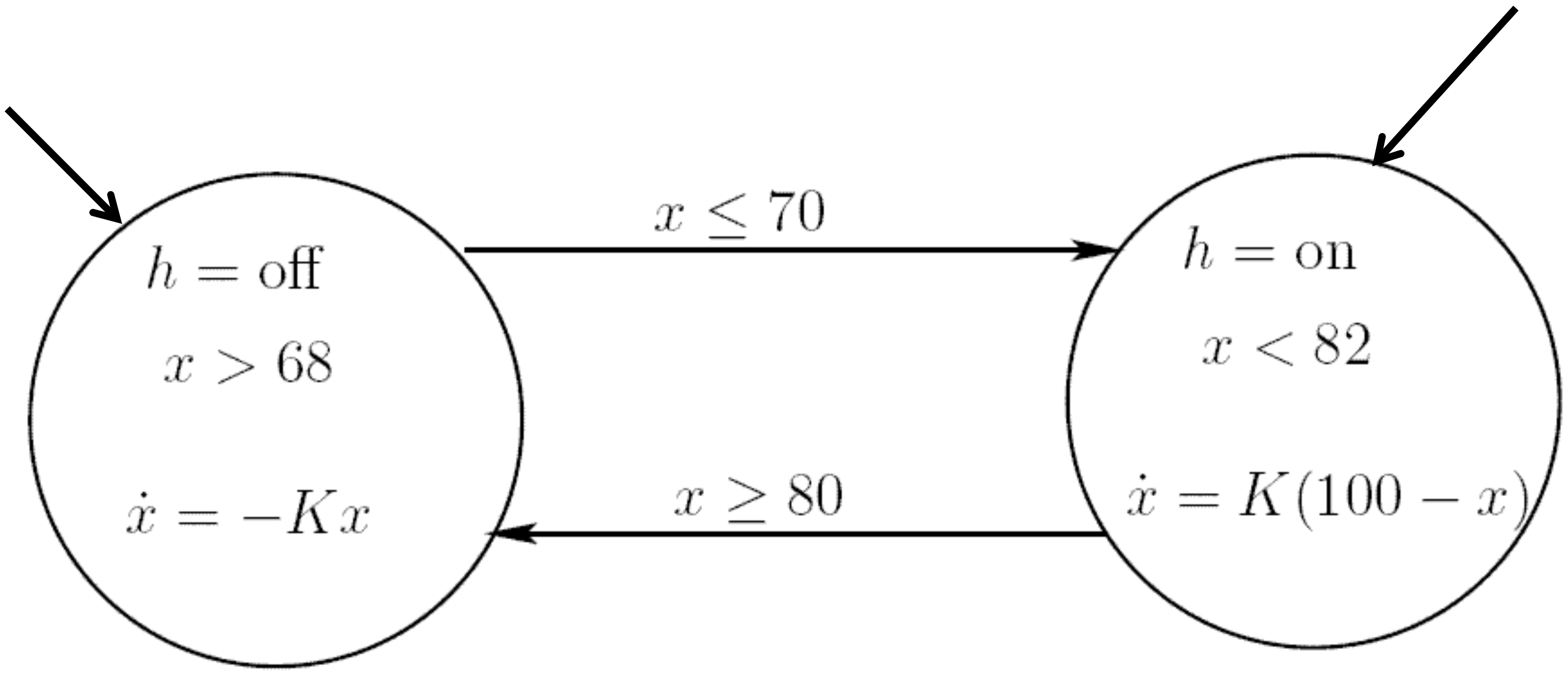
Jumps between modes: (happen instantaneously)

$$h = \text{on} \wedge x \geq 80 \quad \rightarrow \quad h := \text{off}$$
$$h = \text{off} \wedge x \leq 70 \quad \rightarrow \quad h := \text{on}$$

Dynamics of Thermostat



Hybrid Automaton for Thermostat



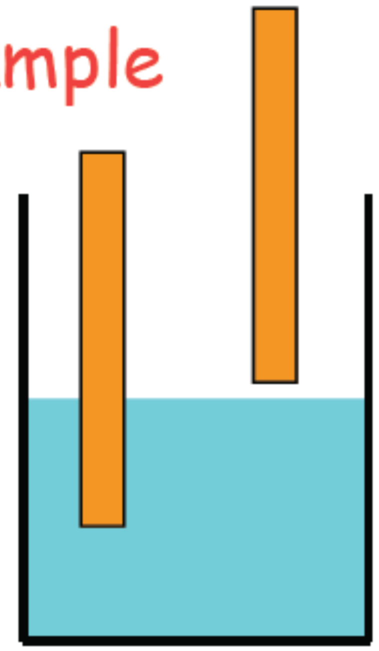
Automaton not deterministic: for some values of x,
non-deterministic choice between continuous evolution and jump

Hybrid Automata

- Digital controller of physical "plant"
  - thermostat
  - controller for power plant
  - intelligent cruise control in cars
  - aircraft auto pilot
- Phased operation of natural phenomena
  - bouncing ball
  - biological cell growth
- Multi-agent systems
  - ground and air transportation systems
  - interacting robots (e.g., RoboSoccer)

Another example

Nuclear reactor example



Without rods

$$\dot{T} = 0.1\,T - 50$$

With rod 1

$$\dot{T} = 0.1\,T - 56$$

With rod 2

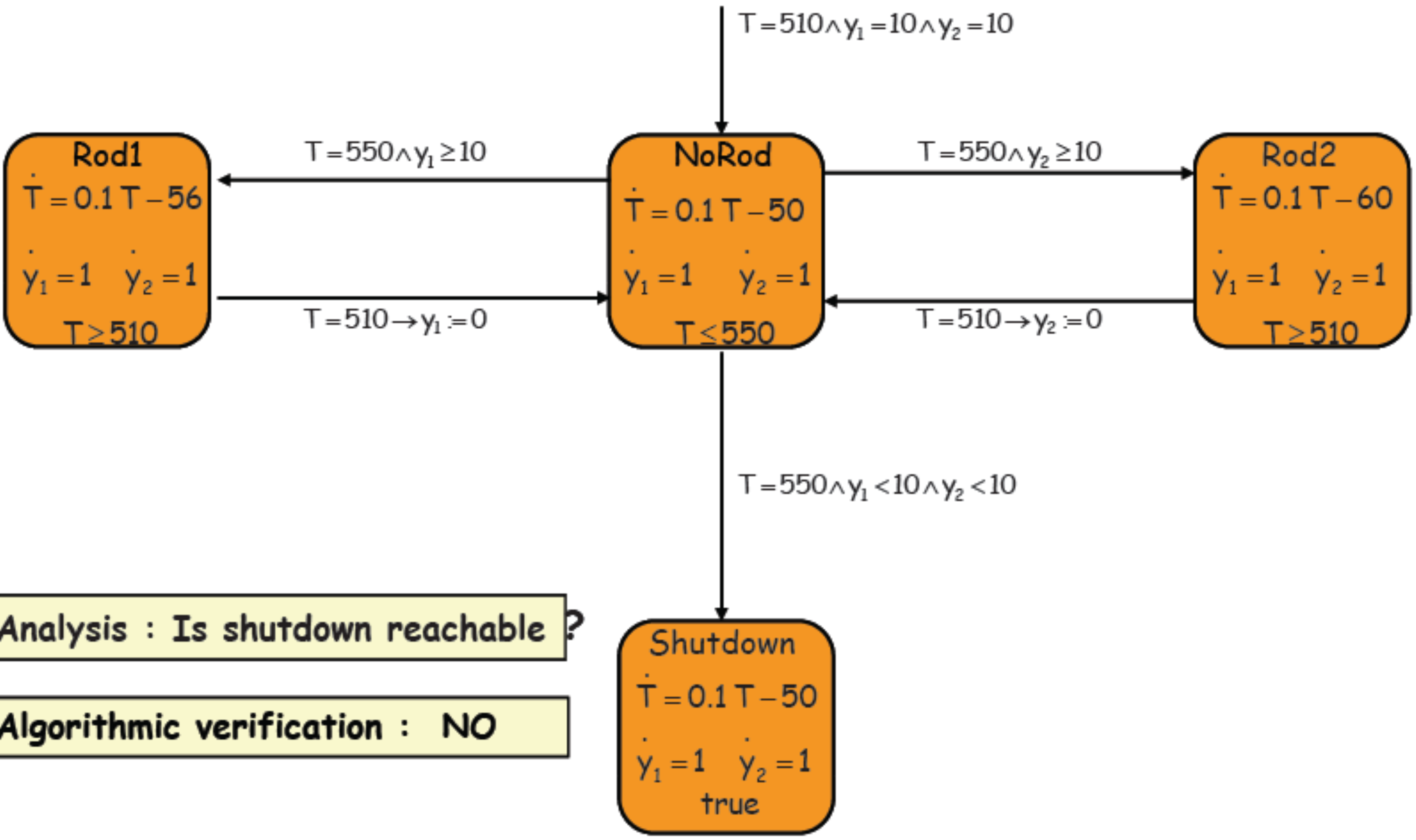
$$\dot{T} = 0.1\,T - 60$$

Rod 1 and 2 cannot be used simultaneously
Once a rod is removed, you cannot use it for 10 minutes

Specification : Keep temperature between 510 and 550 degrees.
If T=550 then either a rod is available or we shutdown the plant.

Example due to George Pappas, UPenn

Nuclear reactor example (contd.)

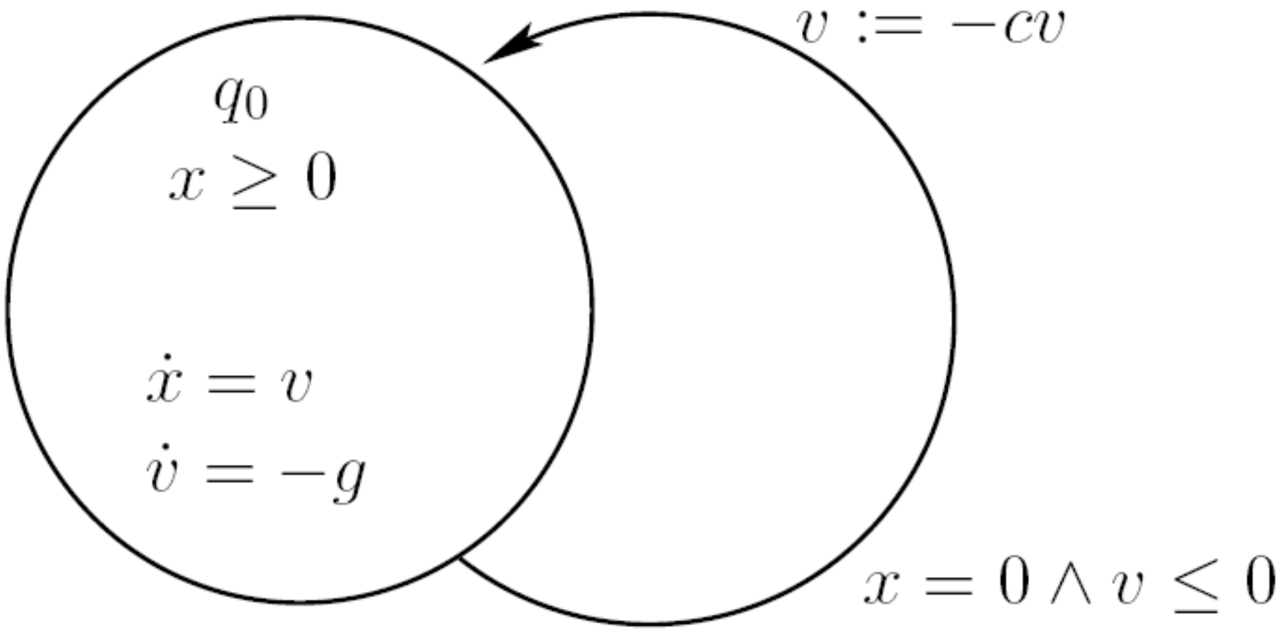

$T=510 \wedge y_1=10 \wedge y_2=10$
Rod1
$\dot{T}=0.1T-56$
$\dot{y}_1=1 \quad \dot{y}_2=1$
$T \geq 510$
$T=550 \wedge y_1 \geq 10$
$T=510 \rightarrow y_1:=0$
NoRod
$\dot{T}=0.1T-50$
$\dot{y}_1=1 \quad \dot{y}_2=1$
$T \leq 550$
$T=550 \wedge y_2 \geq 10$
$T=510 \rightarrow y_2:=0$
Rod2
$\dot{T}=0.1T-60$
$\dot{y}_1=1 \quad \dot{y}_2=1$
$T \geq 510$
$T=550 \wedge y_1<10 \wedge y_2<10$
Shutdown
$\dot{T}=0.1T-50$
$\dot{y}_1=1 \quad \dot{y}_2=1$
true

Analysis : Is shutdown reachable?

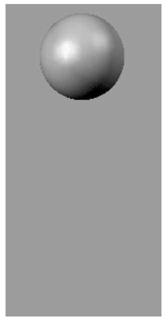
Algorithmic verification : NO

Example due to George Pappas, UPenn

Hybrid Automaton for Bouncing Ball



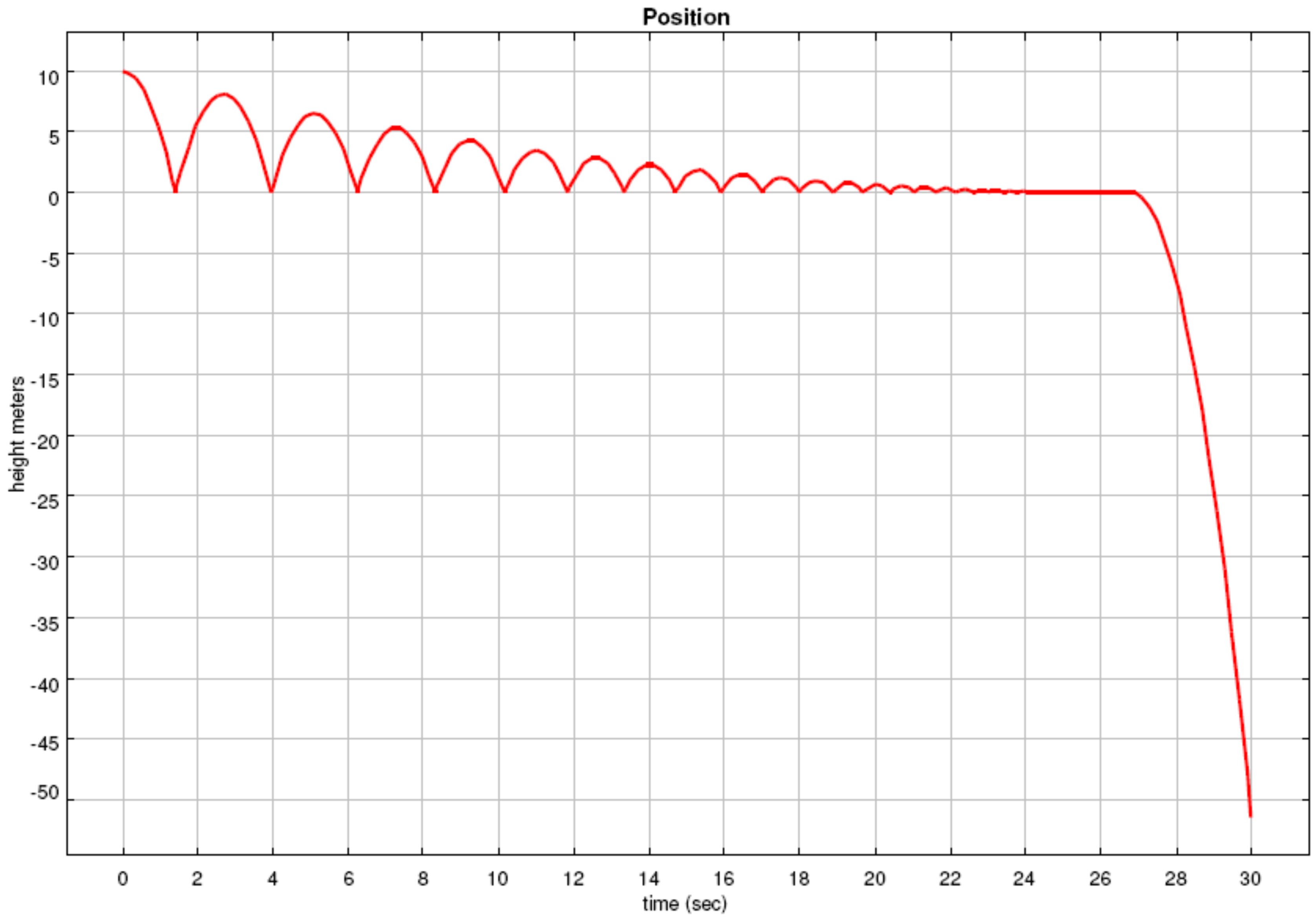
x – vertical distance from ground (position)
v – velocity
c – coefficient of restitution, $0 \cdot c \cdot 1$



Behavior of bouncing ball model in form of hybrid automaton = expected behavior?

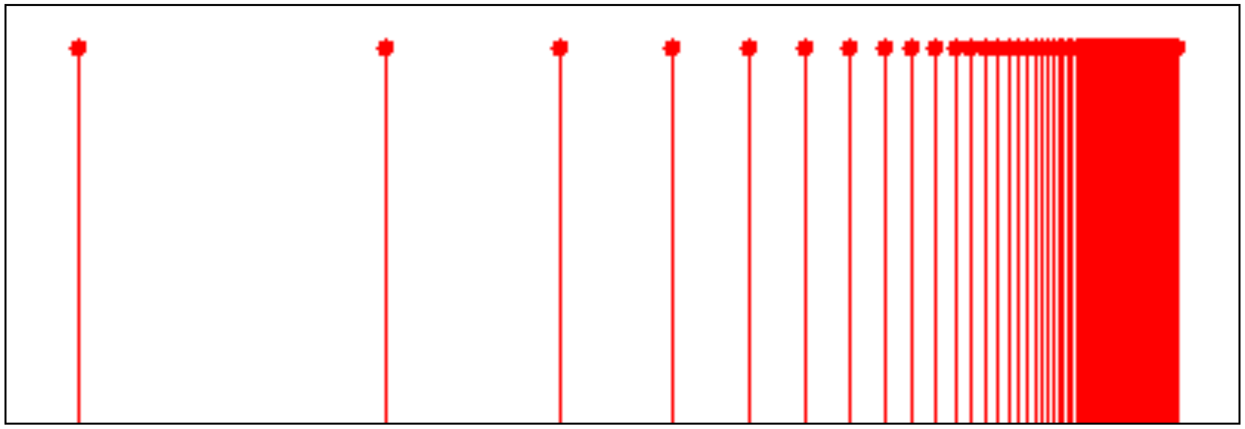
Next:
plot position x as a function of time t ,
where x starts at height $x_{\max}$

Simulation of Bouncing Ball Automaton in Ptolemy II / HyVisual



Zeno Behavior

system makes infinite number of jumps in finite time



A Run/Execution of a Hybrid Automaton

time

$$
\begin{array}{ll}
\tau_0 & (q_0, \mathbf{x}_0) \\
\tau_0' & \rightsquigarrow (q_0, \mathbf{x}_0') \\
= & \downarrow \\
\tau_1 & (q_1, \mathbf{x}_1) \\
\tau_1' & \rightsquigarrow (q_1, \mathbf{x}_1') \\
 & \downarrow \\
 & \vdots \\
 & \downarrow \\
\tau_N & (q_N, \mathbf{x}_N) \\
\tau_N' & \rightsquigarrow (q_N, \mathbf{x}_N') \\
 & \downarrow \\
 & \vdots
\end{array}
$$

$$\tau = \tau_0, \tau_1, \tau_2, \ldots, \tau_N [, \ldots]$$

Continuous extent of τ:

$$|\tau| = \sum_{i=0}^{\infty} \tau_{i+1} - \tau_i$$

Discrete extent of τ:

$$\langle \tau \rangle = \begin{cases} N & \text{if } \tau \text{ is a finite sequence of length } N \\ \infty & \text{if } \tau \text{ is an infinite sequence} \end{cases}$$

Zeno Behavior: Formal Definition

time

$$\begin{array}{ll}
\tau_0 & (q_0, \mathbf{x}_0) \\
\tau_0' & \rightsquigarrow (q_0, \mathbf{x}_0') \\
= & \downarrow \\
\tau_1 & (q_1, \mathbf{x}_1) \\
\tau_1' & \rightsquigarrow (q_1, \mathbf{x}_1') \\
 & \downarrow \\
 & \vdots \\
 & \downarrow \\
\tau_N & (q_N, \mathbf{x}_N) \\
\tau_N' & \rightsquigarrow (q_N, \mathbf{x}_N') \\
 & \downarrow \\
 & \vdots
\end{array}$$

An execution of a hybrid automaton with time set τ is **zeno** iff $\langle \tau \rangle = \infty$ but $|\tau| < \infty$.

Analysis of Zeno Behavior of Bouncing Ball

If $c < 1$ all infinite executions are Zeno. The first bounce occurs at time:

$$\tau_1 = \tau_0 + \frac{v(\tau_0) + \sqrt{v^2(\tau_0) + 2gx(\tau_0)}}{g}$$

The second bounce occurs at time:

$$\tau_2 = \tau_0 + \tau_1 + \frac{2v(\tau_1)}{g}$$

where $v(\tau_1) = -cv(\tau_0') = \sqrt{v^2(\tau_0) + 2gx(\tau_0)}$.

More generally, the Nth bounce occurs at time:

$$\tau_N = \tau_0 + \tau_1 + \frac{2v(\tau_1)}{g} \sum_{i=1}^{N} c^{i-1}$$

For $c \in [0, 1)$, we have $\sum_{i=1}^{\infty} c^{i-1} = \frac{1}{1-c}$.

Thus $\lim_{N\to\infty} \tau_N < \infty$.

Why does Zeno Behavior Arise?

Our model is a mathematical artifact

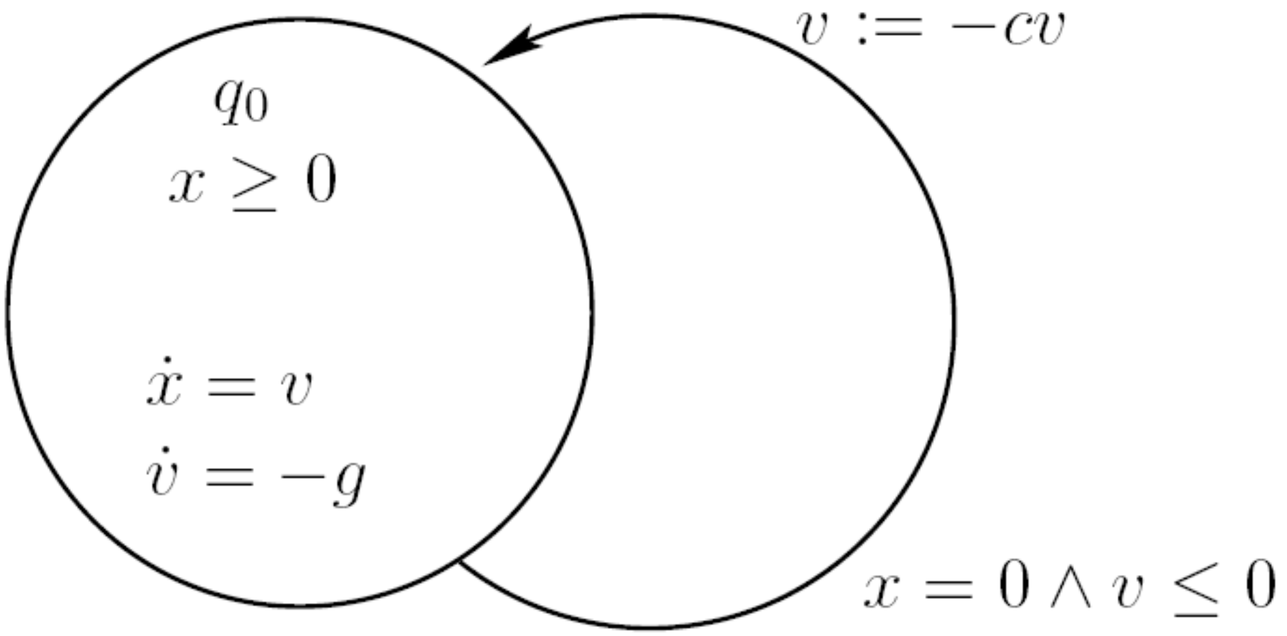
Zeno behavior is possible mathematically

but impossible in real (in physical world).

Some assumption in the model is unrealistic ...

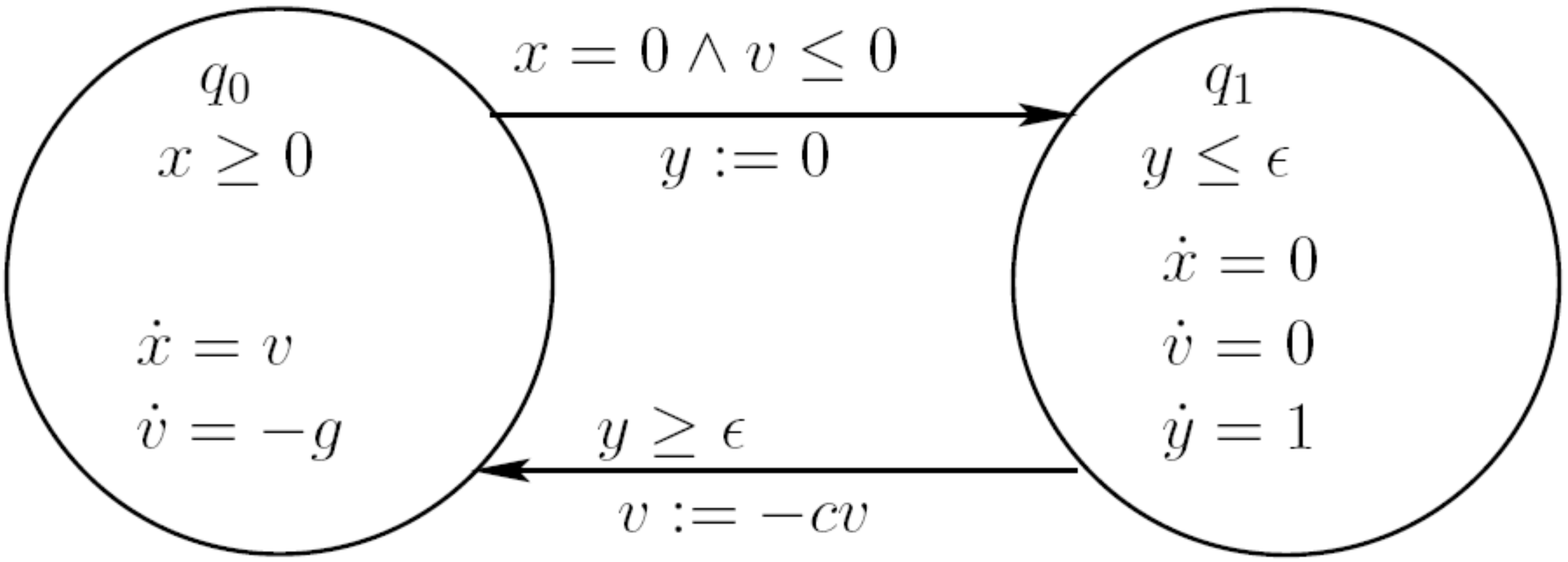
Hybrid Automaton for Bouncing Ball

What's Unrealistic about this model?



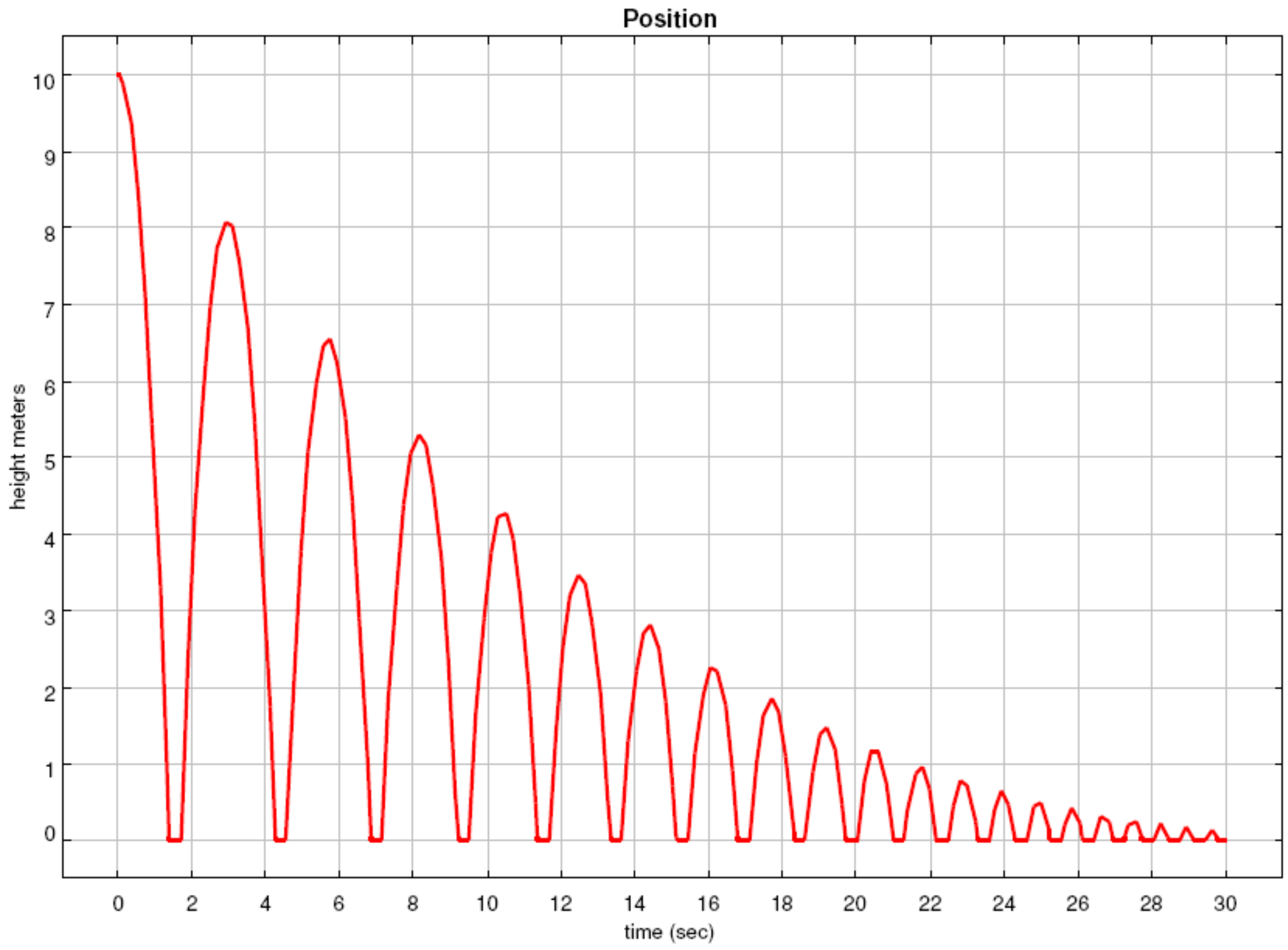
x – vertical distance
v – velocity
c – coefficient of restitution, $0 < c < 1$

Eliminating Zeno Behavior: Regularization

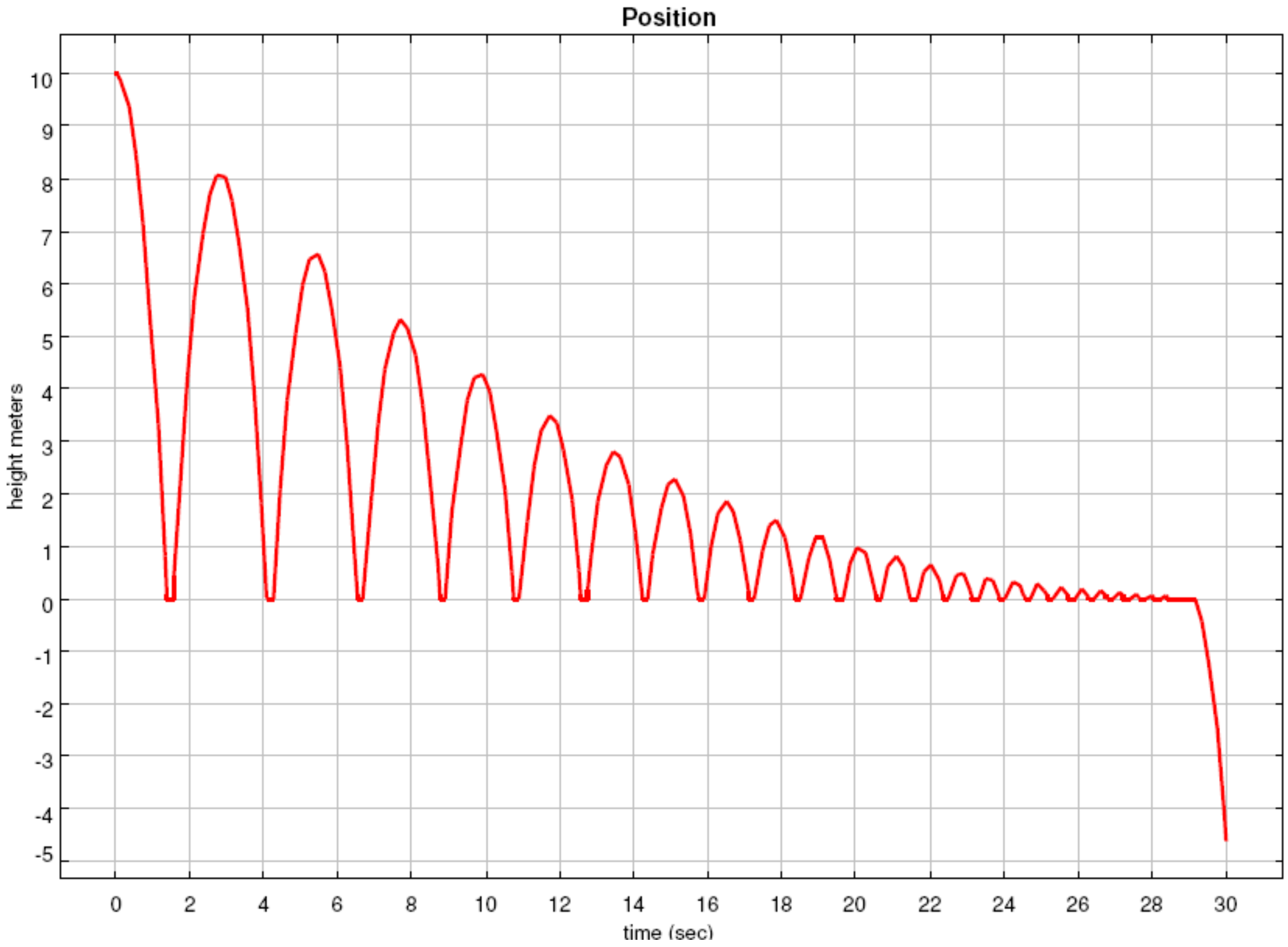


What happens as ε goes to 0?

Simulation for $\varepsilon = 0.3$



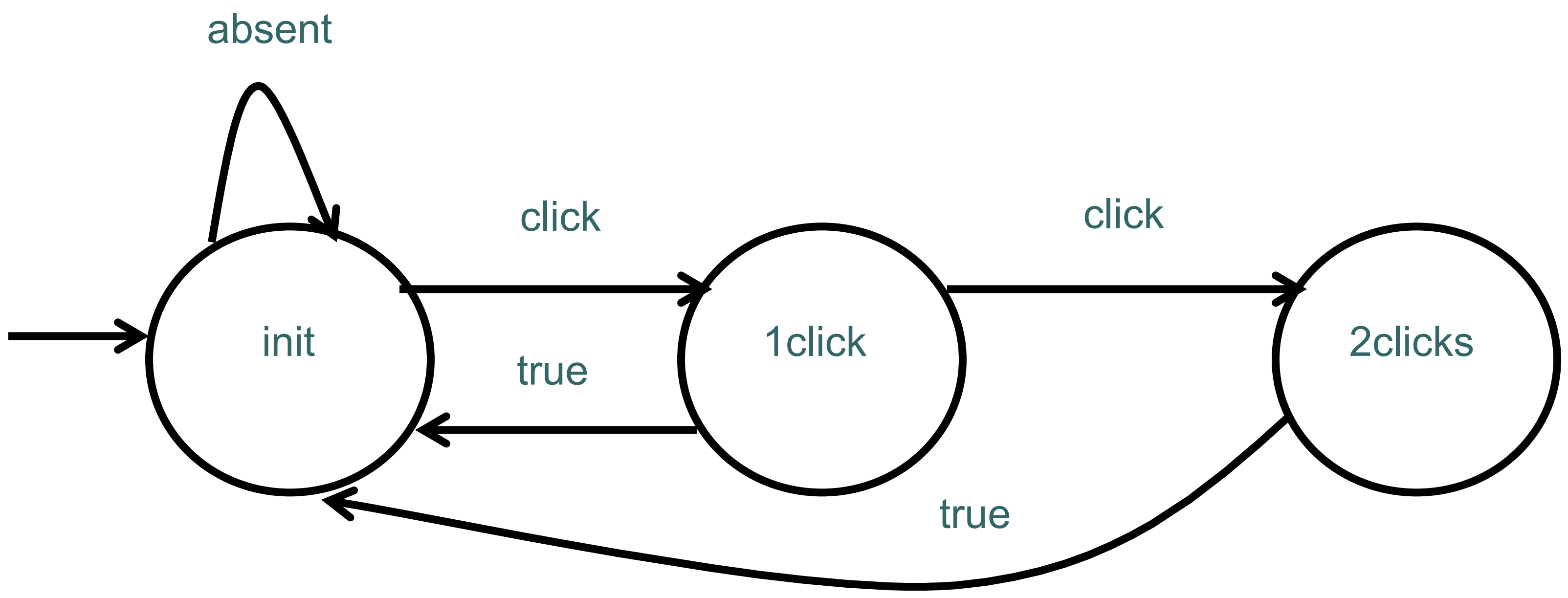
Simulation for ε = 0.15



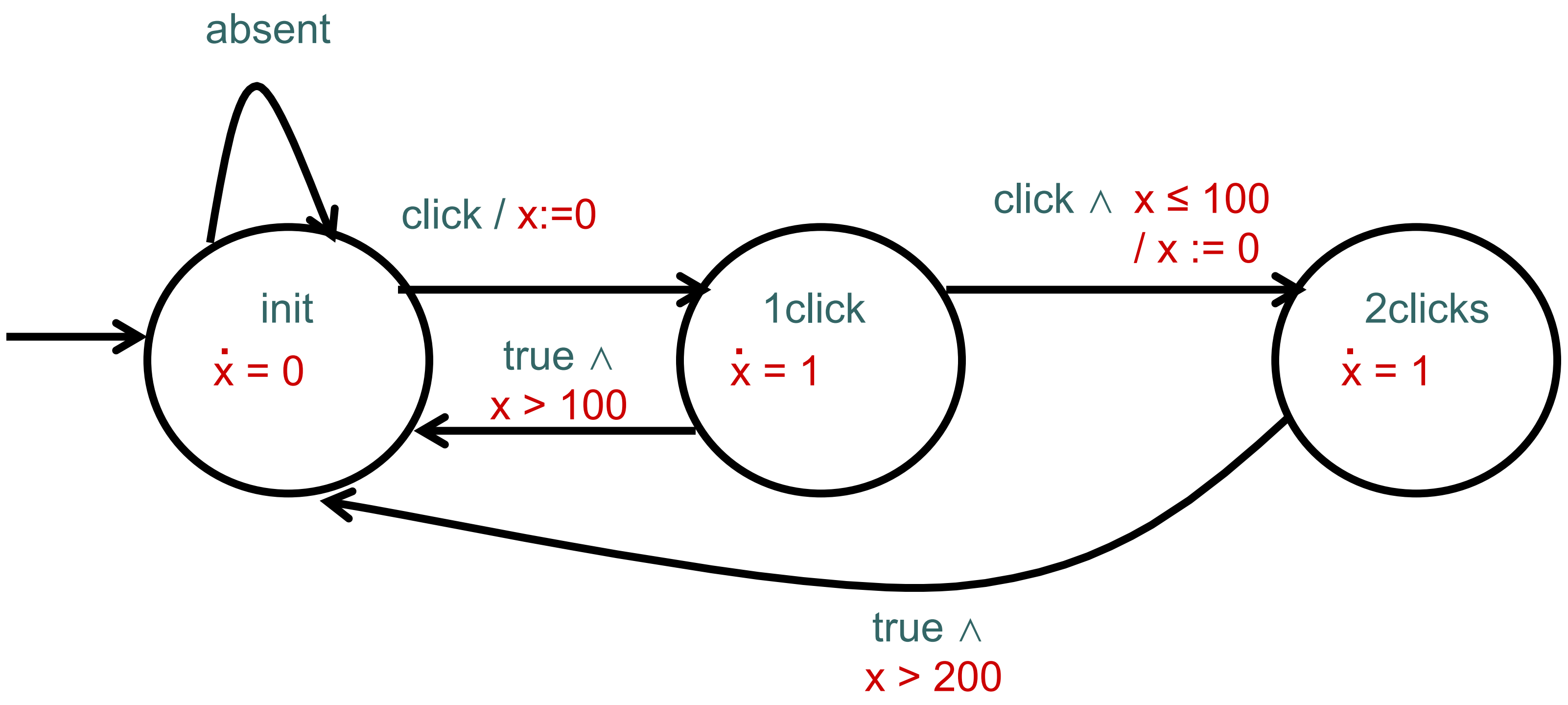
Next: Timed Automata

- sub-class of hybrid automata
- models of real-time systems

Capturing a “Double-Click” of a Mouse with a Finite-State Machine (FSM)



Capturing a "Double-Click" of a Mouse with a Timed Automaton



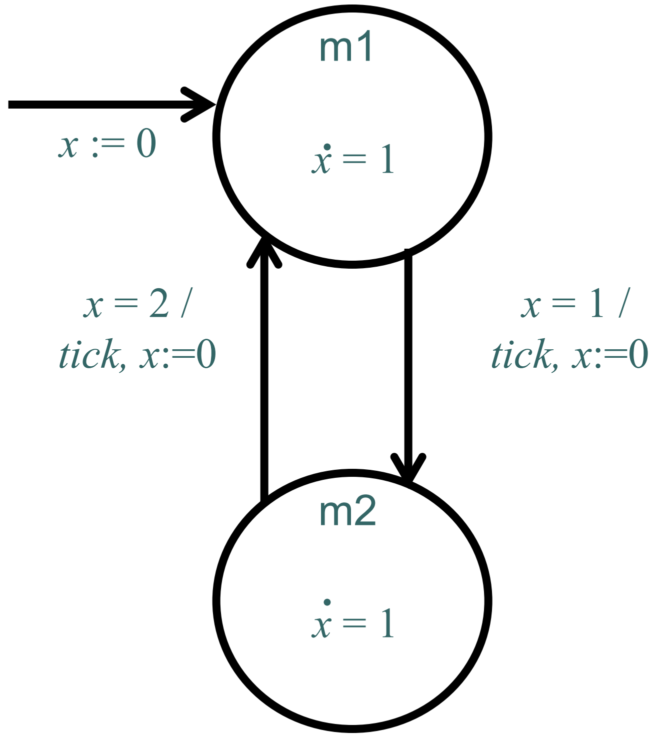
Timed Automata

- RHS of all differential equations is 1 (“ $\dot{x} = 1$ ”)
- Single-speed clocks that precisely tracks real time
- Reset of a clock is possible in jump (“ $x := 0$ ”)

Systems modeled as Timed Automata:

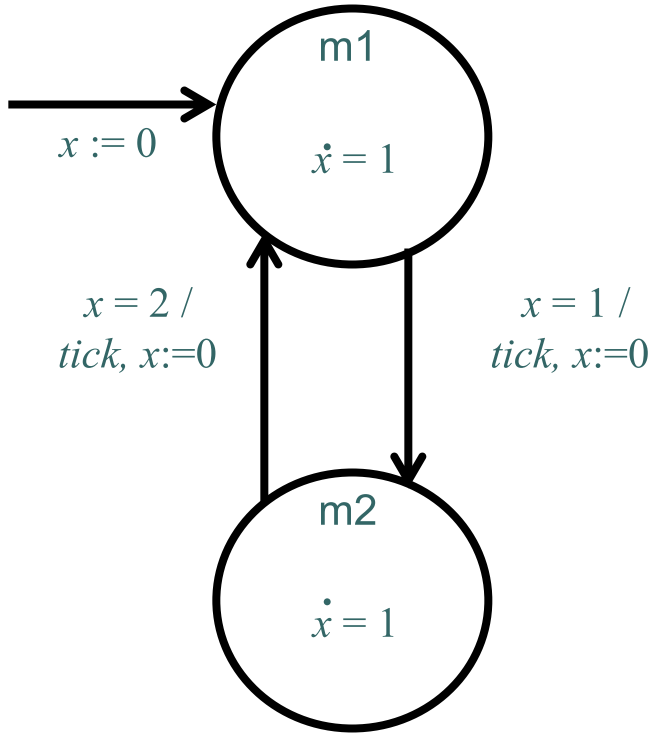
- Real-time controllers
- Self-timed circuits (clock-less circuits)
- Network protocols with timing-dependent behavior
- Scheduling of jobs

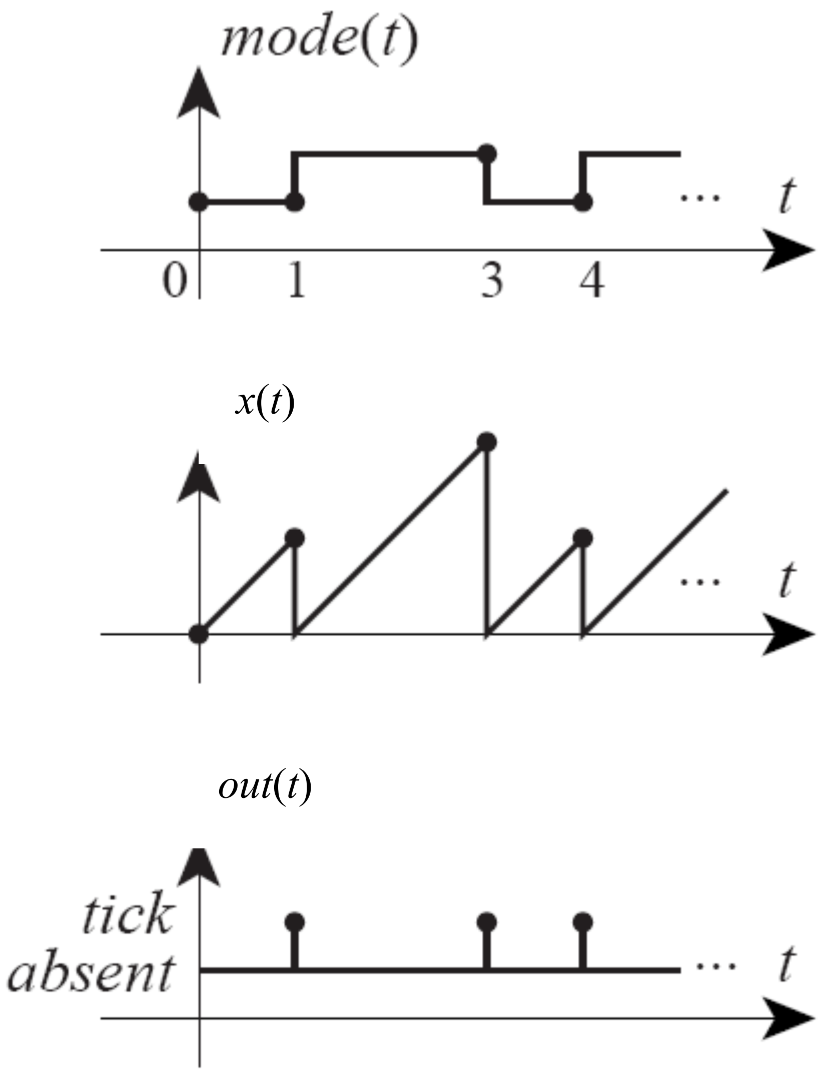
A ‘Tick’ Generator



What does $x(t)$ look like?

A ‘Tick’ Generator





Timed Traces and Time-Abstract (Untimed) Traces

time

$$\begin{array}{ll}
\tau_0 & (q_0, \mathbf{x}_0) \\
\tau_0' & \rightsquigarrow (q_0, \mathbf{x}_0') \\
= & \downarrow \\
\tau_1 & (q_1, \mathbf{x}_1) \\
\tau_1' & \rightsquigarrow (q_1, \mathbf{x}_1') \\
 & \downarrow \\
 & \vdots \\
 & \downarrow \\
\tau_N & (q_N, \mathbf{x}_N) \\
\tau_N' & \rightsquigarrow (q_N, \mathbf{x}_N') \\
 & \downarrow \\
 & \vdots
\end{array}$$

$$\mathbf{x}_i' = \mathbf{x}_i + (\tau_i' - \tau_i)$$

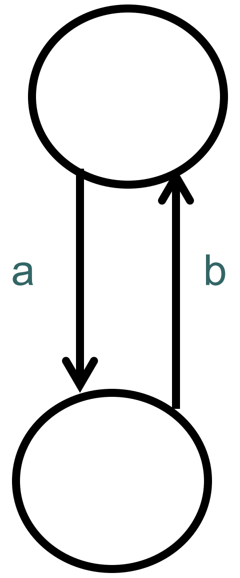
A time-abstract (untimed) trace of M is a sequence

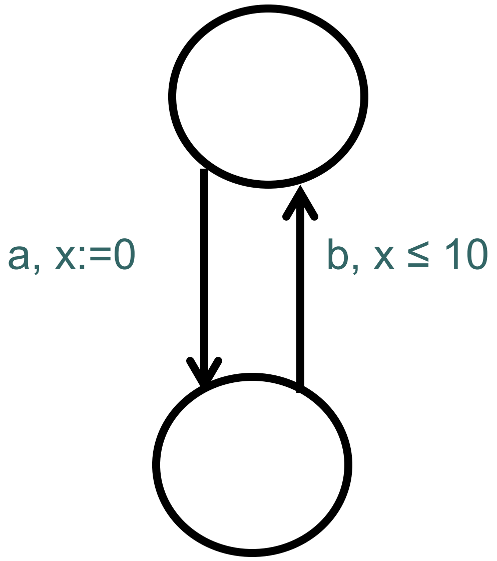
$q_0, q_1, q_2, \ldots$

that can be extended to a timed trace of M

(think of q_i's as also including input and output symbols)

Untimed vs. Timed Automata





Do these automata have the same untimed traces?

Two Problems

Verification

- Does the system do what it’s supposed to do?
  - Does the system satisfy its specifications?

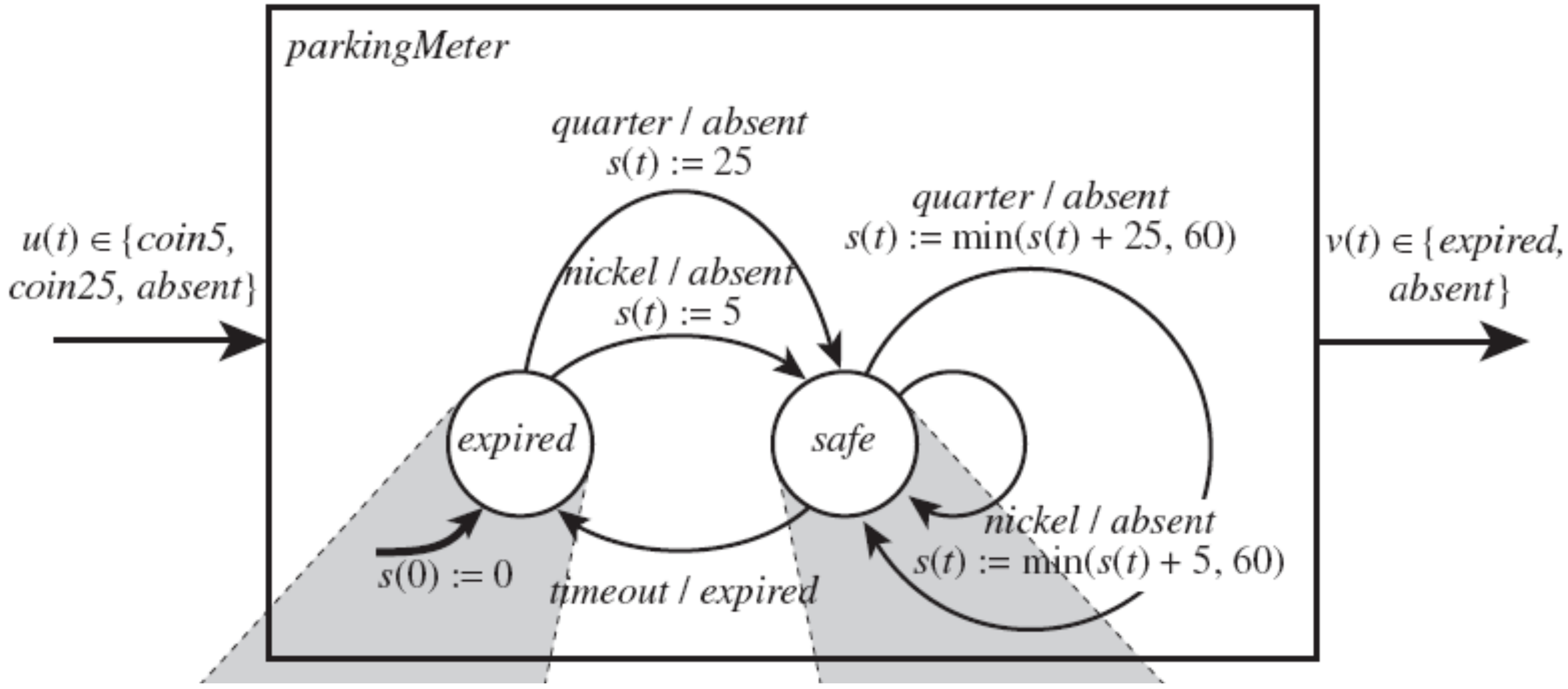
Synthesis/Control

- Construct a system that satisfies its specifications
  - e.g. by synthesizing a controller

In both cases: we need to specify the objective

Untimed Specifications

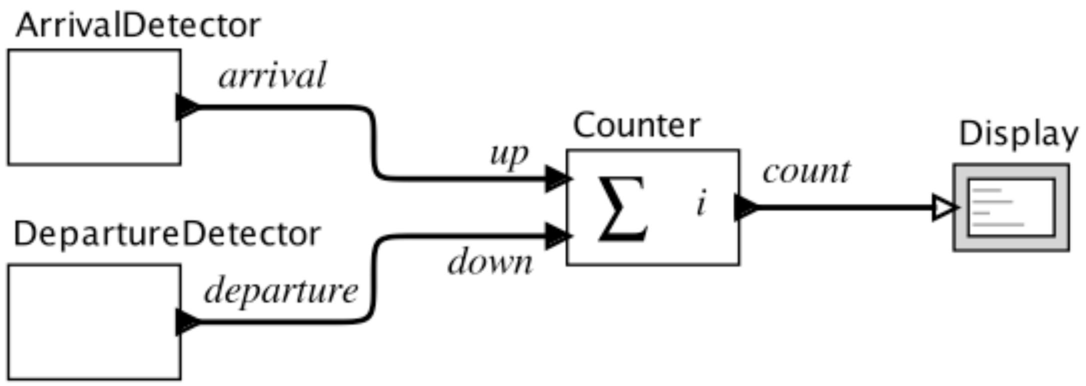
specifications that do not mention time
“parking meter reaches ‘safe’ state when coins are added”



Next: Finite-State Machines (FSM)

Discrete System: Counter

count number of cars that enter or leave parking garage



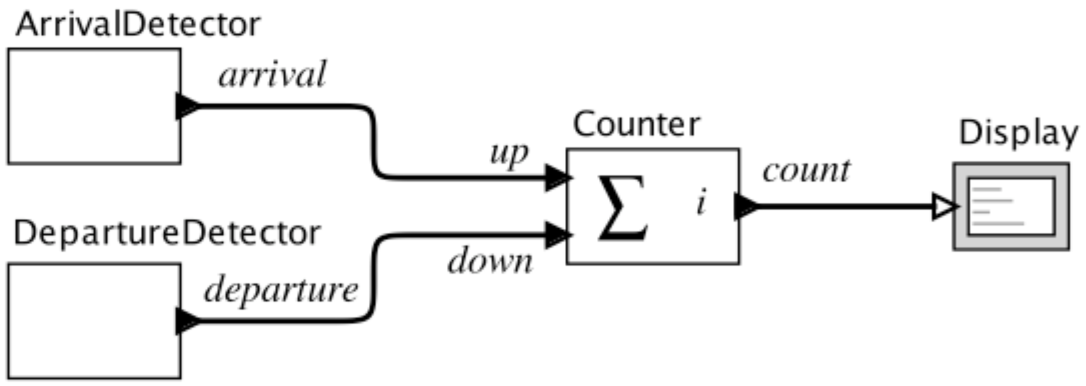
Pure signal: $up\colon \mathbb{R} \to \{absent, present\}$

Discrete actor:

$$Counter\colon (\mathbb{R} \to \{absent, present\})^P \to (\mathbb{R} \to \{absent\} \cup \mathbb{N})$$
$$P = \{up, down\}$$

Reaction

For any $t \in \mathbb{R}$ where $up(t) \neq absent$ or *down*$(t) \neq absent$ the Counter **reacts**. It produces an output value in $\mathbb{N}$ and changes its internal **state**.

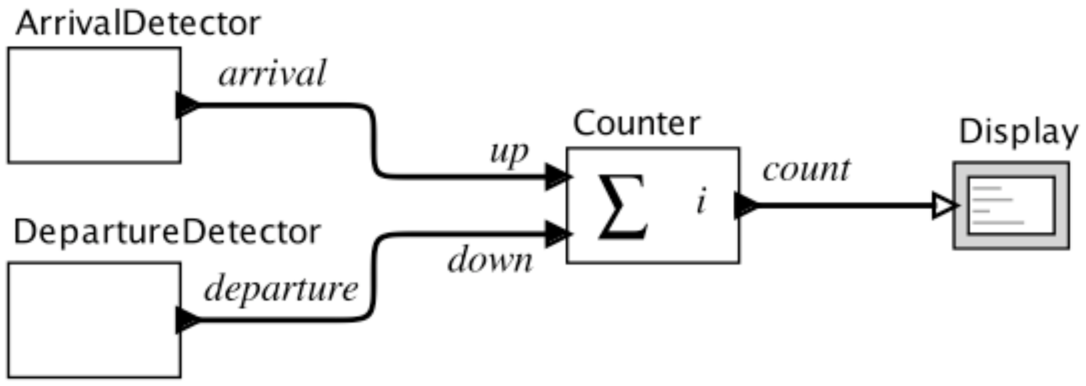


$$Counter\colon (\mathbb{R} \to \{absent, present\})^P \to (\mathbb{R} \to \{absent\} \cup \mathbb{N})$$
$$P = \{up, down\}$$

Input and Output Valuations at a Reaction

For $t \in \mathbb{R}$ a port p has a **valuation**, which is an assignment of a value in V_p (the **type** of port p). A valuation of the input ports $P = \{\mathit{up}, \mathit{down}\}$ assigns to each port a value in $\{absent, present\}$.

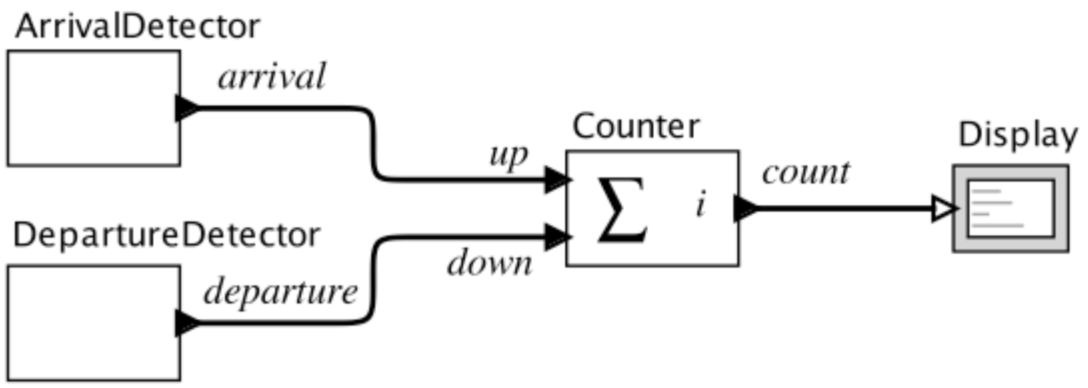
A **reaction** gives a valuation to the output port *count* in the set $\{absent\} \cup \mathbb{N}$.



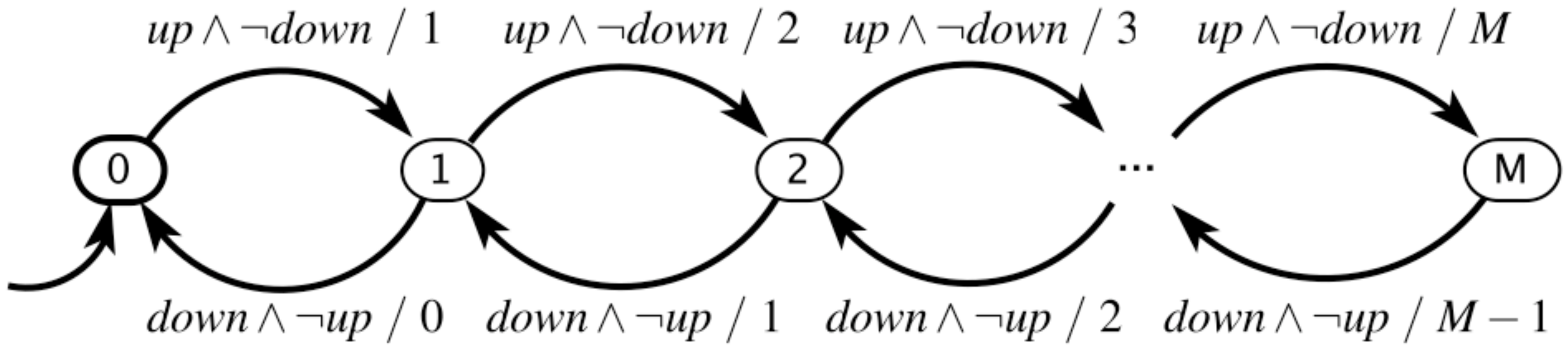
State Space

A practical parking garage has a finite number M of spaces, so the state space for the counter is

$$States = \{0, 1, 2, \cdots, M\} .$$



Finite State Machine (FSM)



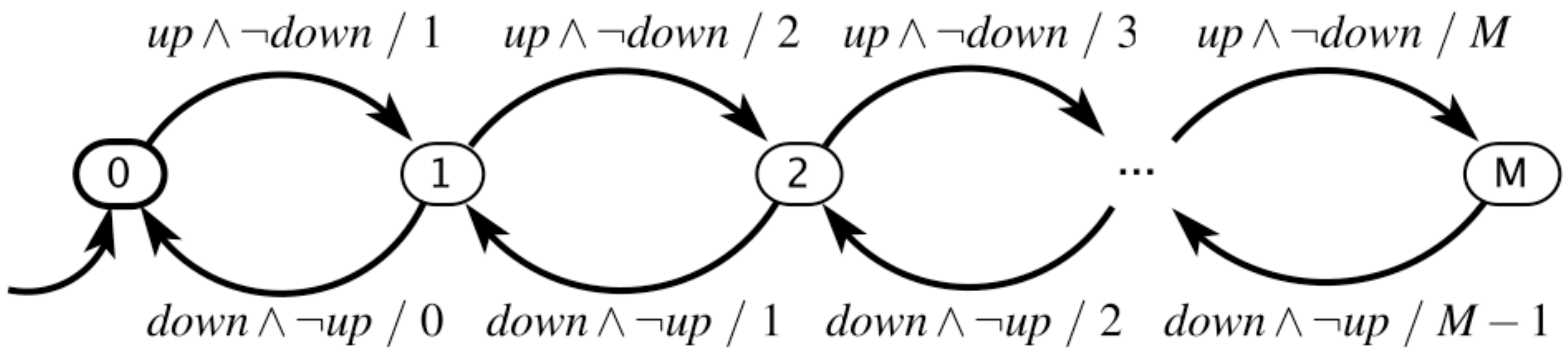
$up \wedge \neg down \ / \ 1$
$up \wedge \neg down \ / \ 2$
$up \wedge \neg down \ / \ 3$
$up \wedge \neg down \ / \ M$
0
1
2
...
M
$down \wedge \neg up \ / \ 0$
$down \wedge \neg up \ / \ 1$
$down \wedge \neg up \ / \ 2$
$down \wedge \neg up \ / \ M-1$

Guard g is specified using the predicate

$$up \wedge \neg down$$

which means that *up* has value *present* and *down* has value *absent*.

Garage Counter Mathematical Model

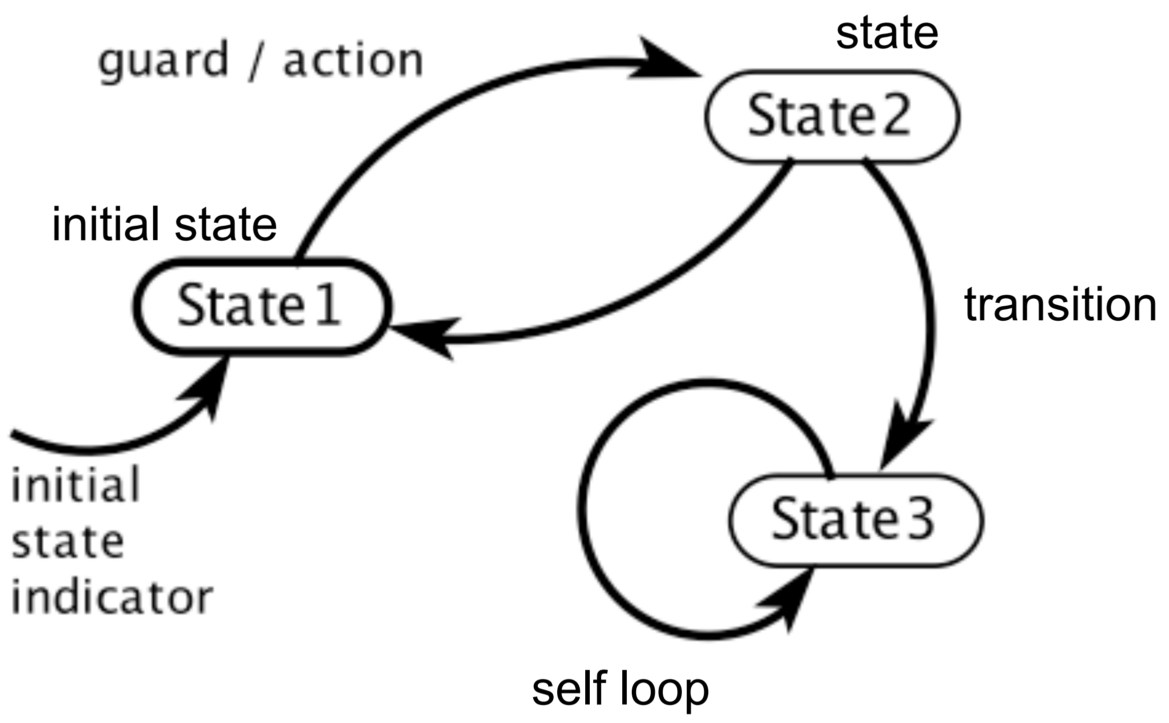

$up \wedge \neg down$ / 1
$up \wedge \neg down$ / 2
$up \wedge \neg down$ / 3
$up \wedge \neg down$ / M
0
1
2
...
M
$down \wedge \neg up$ / 0
$down \wedge \neg up$ / 1
$down \wedge \neg up$ / 2
$down \wedge \neg up$ / $M-1$

Formally: $(States, Inputs, Outputs, update, initialState)$, where

- $States = \{0, 1, \cdots, M\}$
- $Inputs$ is a set of input valuations
- $Outputs$ is a set of output valuations
- $update : States \times Inputs \to States \times Outputs$
- $initialState = 0$

update function defined by labeled edges

FSM Notation



Guards for *Pure* Signals

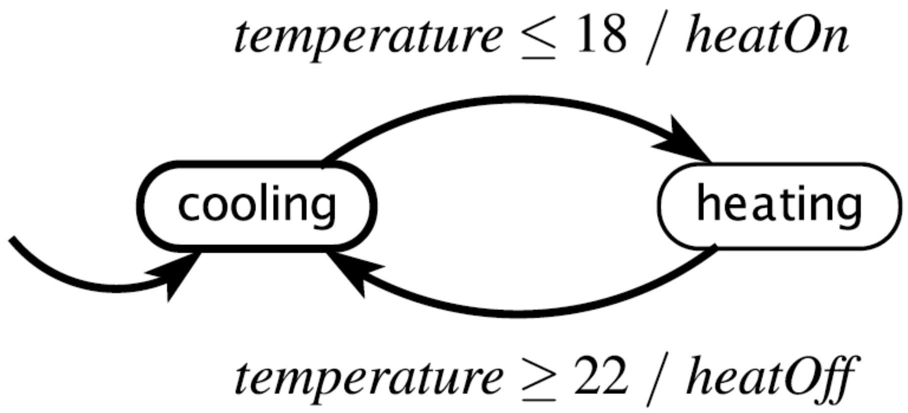
true	Transition is always enabled.
p_1	Transition is enabled if p_1 is $present$.
$\neg p_1$	Transition is enabled if p_1 is $absent$.
$p_1 \wedge p_2$	Transition is enabled if both p_1 and p_2 are $present$.
$p_1 \vee p_2$	Transition is enabled if either p_1 or p_2 is $present$.
$p_1 \wedge \neg p_2$	Transition is enabled if p_1 is $present$ and p_2 is $absent$.

Guards for Signals with Numerical Values

p_3	Transition is enabled if p_3 is *present* (not *absent*).
$p_3 = 1$	Transition is enabled if p_3 is *present* and has value 1.
$p_3 = 1 \wedge p_1$	Transition is enabled if p_3 has value 1 and p_1 is *present*.
$p_3 > 5$	Transition is enabled if p_3 is *present* with value greater than 5.

Example: Thermostat

input: *temperature* : $\mathbb{R}$
outputs: *heatOn*, *heatOff* : pure



$temperature \leq 18$ / $heatOn$
cooling
heating
$temperature \geq 22$ / $heatOff$

From this picture, one can construct the formal mathematical model.