Course on Hybrid Systems

Maria Prandini

Politecnico di Milano, Italy

CONTACT INFO

Organizer and lecturer:
Maria Prandini
Politecnico di Milano, Italy
maria.prandini@polimi.it

Additional lecturers:

Goran Frehse
University Grenoble Alpes, France
goran.frehse@univ-grenoble-alpes.fr

Olaf Stursberg
Universität Kassel, Germany
stursberg@uni-kassel.de

COURSE STRUCTURE AND SCHEDULE

Monday February 20
14.30 – 15.15 Introduction and motivation (1h)
15.15 – 16.00 Modeling (1h)
16.30 – 18.00 Modeling (2h)

Tuesday February 21
11.00 – 12.30 Stability (2h)
14.30 – 16.00 Stability (2h)
16.30 – 17.15 Observability (1h)

Wednesday February 22
11.00 – 12.30 Observability (2h)
14.30 – 16.00 Reachability (2h, Goran)
16.30 – 18.00 Reachability (2h, Goran)

Thursday February 23
10.00 – 12.30 Tools for reachability (3h with 15 minutes break, Goran)
14.30 – 16. 00 Control design (2h, Olaf)
16.30 – 18.00 Control design (2h, Olaf)

Friday February 24
10.00 – 12.30 Control design (3h with 15 minutes break, Olaf)
12.30 – 12.45 Conclusions

BIBLIOGRAPHY

Key references:

- John Lygeros
  Lecture Notes on Hybrid Systems
- H. Lin and P.J. Antsaklis
  Hybrid Dynamical Systems: An introduction to Control and Verification.
  Foundations and Trends in Systems and Control. Vol. 1, No. 1 (2014) 1–172
- Daniel Liberzon
  Switching in Systems and Control
  Birkhauser, 2003

Papers covering additional parts of the course will be listed on the website
http://home.deib.polimi.it/prandini/Hybrid_Systems_2017.html

FINAL EXAM

Two options:

- solve some assignments (questions and numerical exercises)
- develop a small project on a topic to be agreed upon.

WHAT IS A HYBRID SYSTEM?

A system with interacting continuous and discrete dynamics, combining

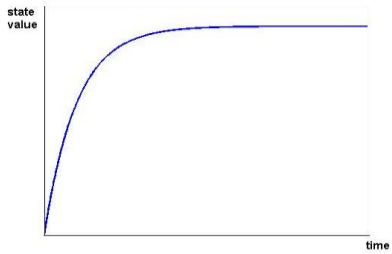
- time-driven continuous systems

  state takes values in a continuous set and changes as time progresses

- event-driven discrete systems

  state takes values in a discrete set and changes due to the occurrence of an event

TIME DRIVEN vs. EVENT-DRIVEN SYSTEMS

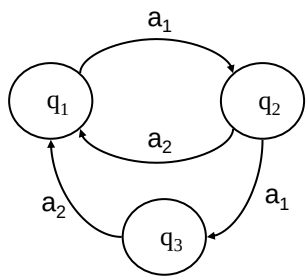


Time driven system

State space: $X = \Re$

Dynamics: $dx/dt = f(x)$

ordinary differential equation (ODE)



Event-driven system

State space: $Q = \{q_1, q_2 q_3\}$

Dynamics: $q(i+1) = \Phi(q(i), a(i))$

a(i) = i-th event, finite automaton

WHAT IS A HYBRID SYSTEM?

A system with interacting continuous and discrete dynamics, combining

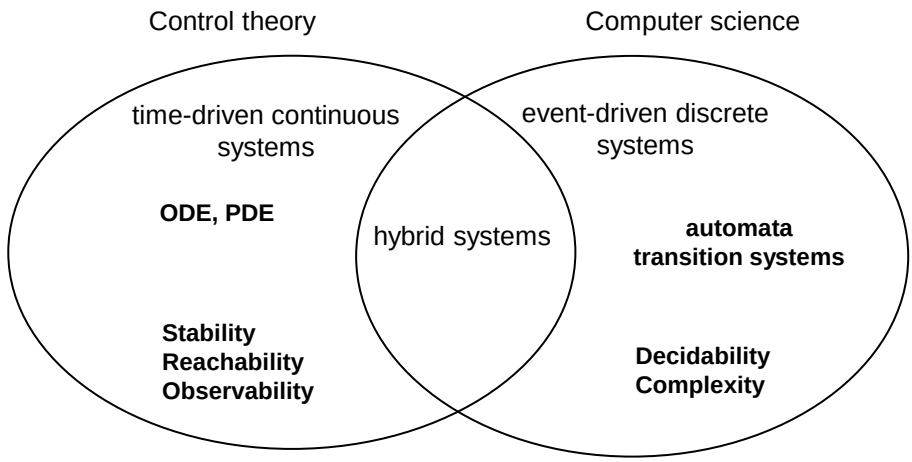
- time-driven continuous systems

  state takes values in a continuous set and changes as time progresses

- event-driven discrete systems

  state takes values in a discrete set and changes due to the occurrence of an event

HYBRID SYSTEMS THEORY



HYBRID BEHAVIOR ARISES IN…

Continuous systems with phased operation

dynamics inherently hybrid (mechanical systems with collisions, robotics, engine control, circuits with diodes, biological cell growth and division)

Continuous systems controlled by discrete logic

quantized control of a continuous system (thermostat, chemical plant with valves and pumps)
embedded systems where computational systems modeled as finite-state machines are coupled with plants and controllers modeled by continuous systems
networked control systems where sensors, controllers and actuators are connected by a shared network medium
switching control systems where some supervisor decides which controller in a given set to apply and when to switch to a different one

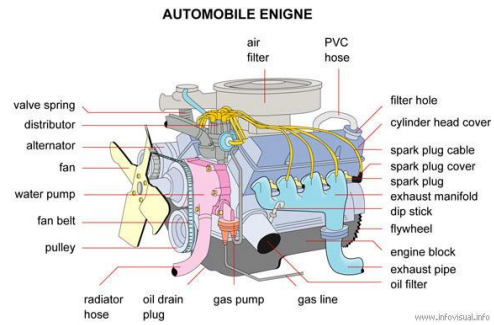
Coordination of multi-agent systems

resource allocation for competing agents with a continuous dynamics (air and ground transportation systems)

Hybrid specifications

controlled system is continuous but specifications are discrete or hybrid (process industry, manufacturing systems)

ENGINE CONTROL



a four-stroke gasoline engine is naturally modeled using four modes corresponding to the position of the pistons, while combustion and power train dynamics are continuous

ROBOTICS

a manipulator is governed by continuous dynamics, but impacts and load shifting cause discrete and asyncronous changes

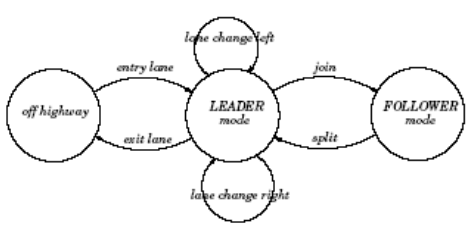
AIR TRAFFIC CONTROL

- predefined set of commands (speed change, short cut, detour...) issued by air traffic controllers to pilots to avoid conflicts
- sequencing of aircraft landing to some airport (distance to avoid turbulence caused by preceeding aircraft, timining constraints)

AUTOMATED HIGHWAY SYSTEM

Goal: increase highway throughput

Cars organized in platoons



Coordination and communication protocols

More conventional controllers govern engines and breaks

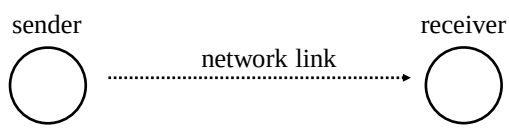
CHEMICAL PROCESS CONTROL

- a sequence of instructions is executed to produce some chemical substance (discrete specifications)
- each instruction possibly involves the control of continuous dynamics
  - keeping some reference temperature while a chemical reaction takes place
  - maintaining some level in a tank (valves and pumps implement a discrete control)

DATA COMMUNICATION NETWORK

Data flows are modeled as continuous variables

Traffic control mechanism induces discrete controls



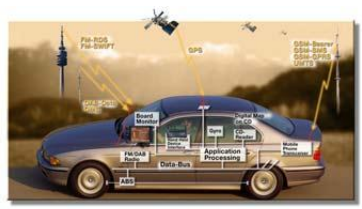
The network link may drop some packet

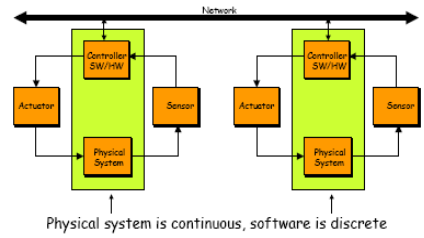
TCP/IP protocol (simplified version)

- sender increases the sending rate linearly until the first packet loss (no ack received in some time window)
- then the sending rate is reduced to half (congestion control)

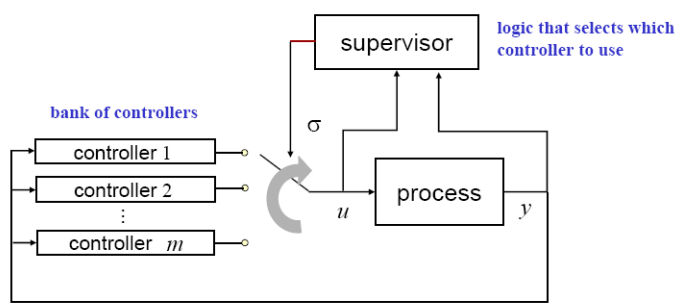
EMBEDDED SYSTEMS

- micro-processors with an inherently discrete behavior (e.g. due to finite precision computations and quantization of signals) embedded in a physical device
- integrated with the physical world (continuous environment) through actuators and sensors
- sharing data and resources by a networked architecture (networked embedded systems)



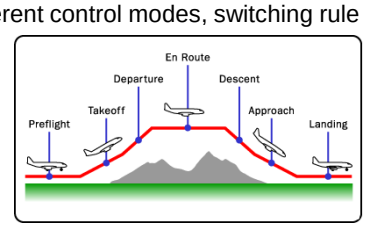


SWITCHING CONTROL

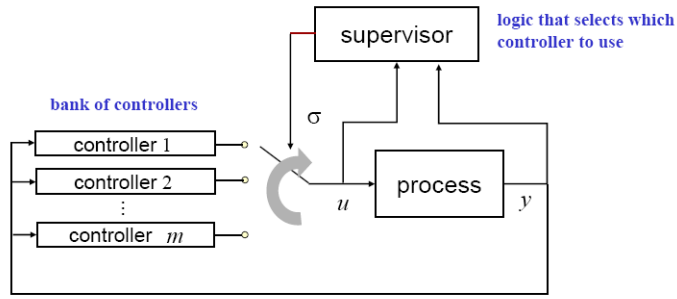


Control of a complex plant:

different controllers designed for different control modes, switching rule coded by a DES (e.g. flight control)



SWITCHING CONTROL



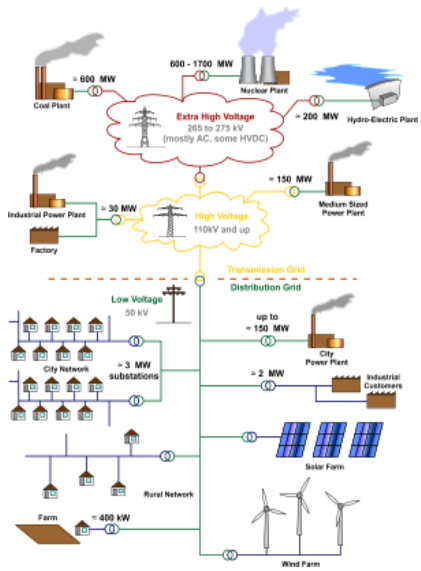
Control of a complex plant:

different controllers designed for different control modes, switching rule coded by a DES (e.g. flight control)

Adaptive switching control:

different controllers designed for different admissible models for the uncertain plant, switching rule based on data collected from plant

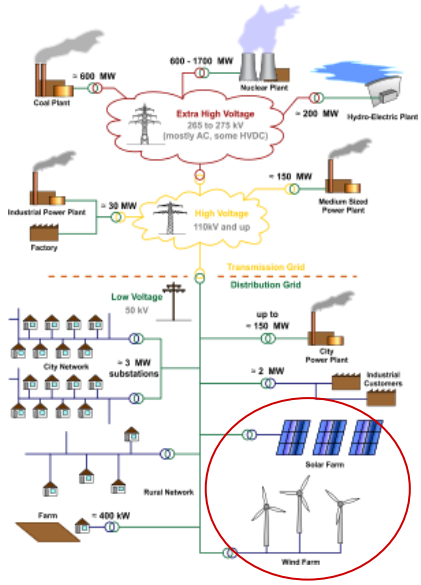
POWER NETWORKS



continuous dynamics
model the evolution of voltages, frequencies, flows, etc.

discrete dynamics
model changes in network topology

POWER NETWORKS

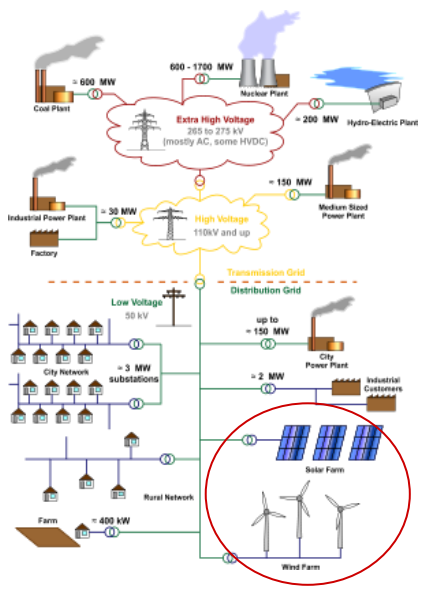


continuous dynamics
model the evolution of voltages, frequencies, flows, etc.

discrete dynamics
model changes in network topology

probability
models the uncertainty about power demand and (with the advent of renewables) power supply

POWER NETWORKS



continuous dynamics
model the evolution of voltages, frequencies, flows, etc.

discrete dynamics
model changes in network topology

probability
models the uncertainty about power demand and (with the advent of renewables) power supply

EC funded projects:
MoVeS “Modeling, verification and control of complex systems: From foundations to power network applications” 2010-2013
UnCoVerCPS “Unifying Control and Verification of Cyber-Physical Systems” 2015-2018

Examples of hybrid automaton

EXAMPLE: THERMOSTAT

Temperature in a room controlled by a thermostat switching a heater on and off

Dynamics of the temperature x (in °C):

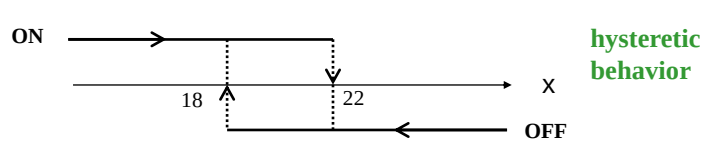
heater on: $\dot{x} = -0.2x + 6 \quad (x \to 30)$

heater off: $\dot{x} = -0.2x \qquad (x \to 0)$

Goal: regulate the temperature around 20°C

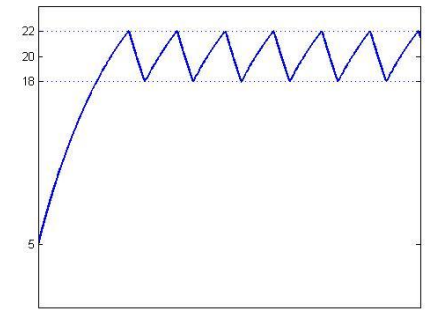
Strategy: turn the heater from OFF to ON as soon as x $\leq$ 18

turn the heater from ON to OFF as soon as x $\geq$ 22



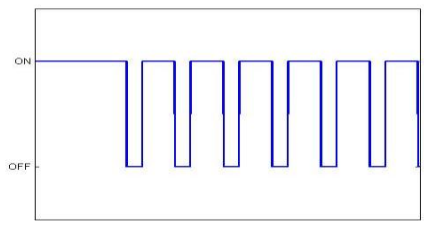
hysteretic behavior

EXAMPLE: THERMOSTAT



evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

"execution" or "solution" of the hybrid system



evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON

EXAMPLE: THERMOSTAT

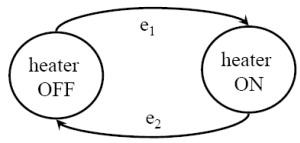
- Continuous dynamics

$$\dot{x} = -0.2x$$

$$\dot{x} = -0.2x + 6$$

linear ODEs describing the temperature evolution

- Discrete dynamics



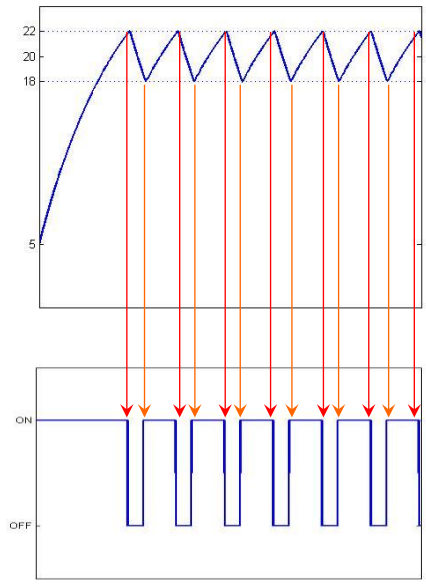
finite automaton describing the behavior of the thermostat

Q = {ON,OFF}

ON = Φ(OFF,e_1) $e_1 = [x \leq 18]$

OFF = Φ(ON,e_2) $e_2 = [x \geq 22]$

EXAMPLE: THERMOSTAT

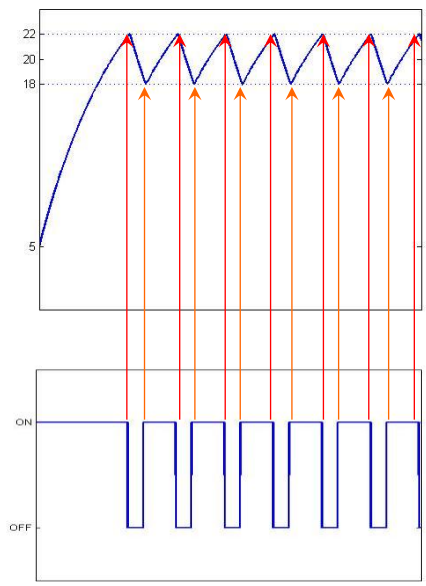


evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

x generates the events causing a transition
from OFF to ON $x \leq 18$
from ON to OFF $x \geq 22$

evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON

EXAMPLE: THERMOSTAT

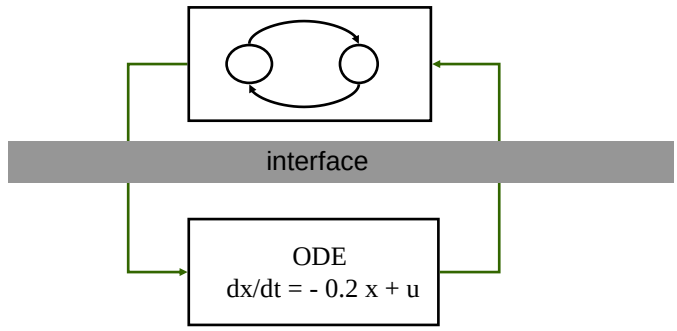


evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

A heater transition causes a switch in the dynamics governing x

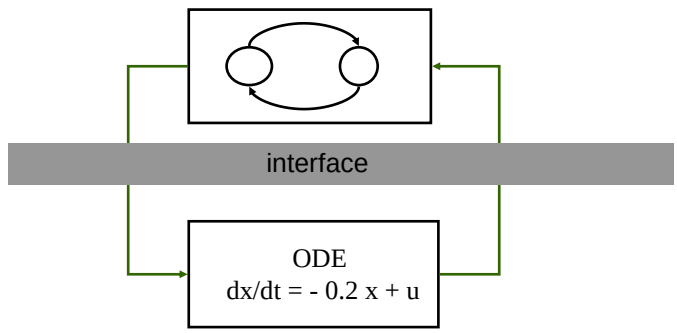
evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON

EXAMPLE: THERMOSTAT



continuous systems controlled by a discrete logic

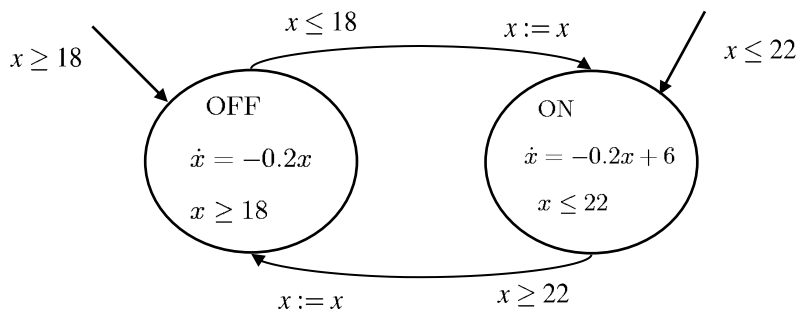
EXAMPLE: THERMOSTAT



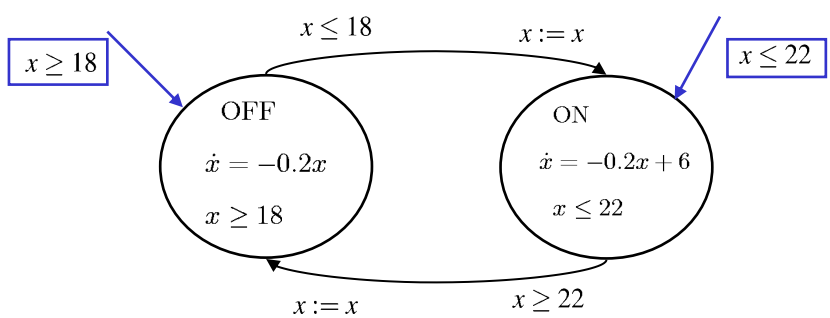
quantized control input (ON → heating power u = 6
OFF → heating power u= 0)

continuous systems controlled by a discrete logic

EXAMPLE: THERMOSTAT



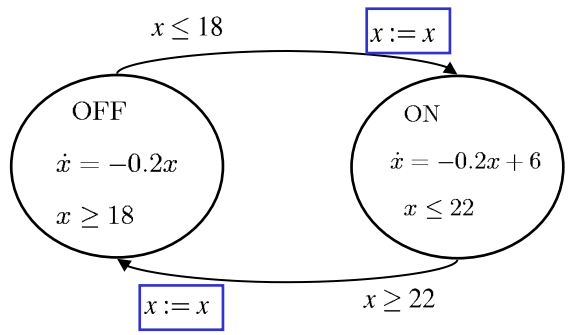
EXAMPLE: THERMOSTAT



Initialization of the system:

Init = {(OFF, x): x$\geq$ 18 } $\cup$ {(ON, x): x$\leq$ 22 }

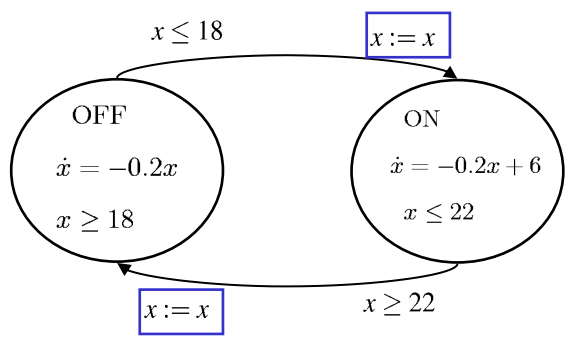
EXAMPLE: THERMOSTAT



Reset of the continuous state x due to the transition

$x := x \rightarrow x := x^-$

EXAMPLE: THERMOSTAT



Reset of the continuous state x due to the transition

$x := x \rightarrow x := x^-$

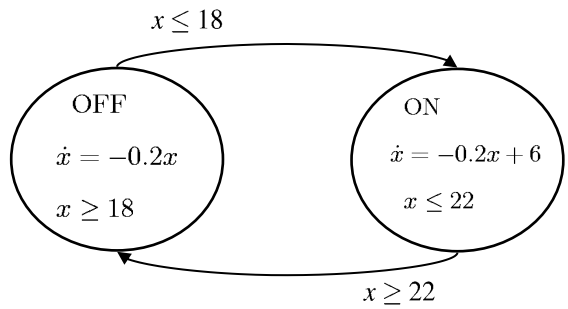
right continuous: $x(t) = x^+(t)$, $\forall\, t$

$x : [0,\infty) \rightarrow \Re$ piecewise continuous

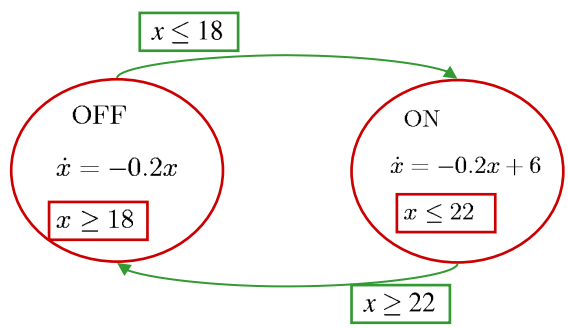
$x^- : (0,\infty) \rightarrow \Re \quad x^-(t) := \lim_{\varepsilon \rightarrow 0^+} x(t-\varepsilon)$

$x^+ : [0,\infty) \rightarrow \Re \quad x^+(t) := \lim_{\varepsilon \rightarrow 0^+} x(t+\varepsilon)$

EXAMPLE: THERMOSTAT



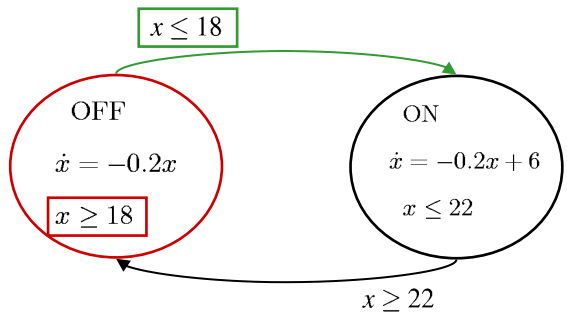
EXAMPLE: THERMOSTAT



If x exits the Domain (or “invariant set”), then a discrete transition must occur

Guard conditions enable the discrete transition

EXAMPLE: THERMOSTAT



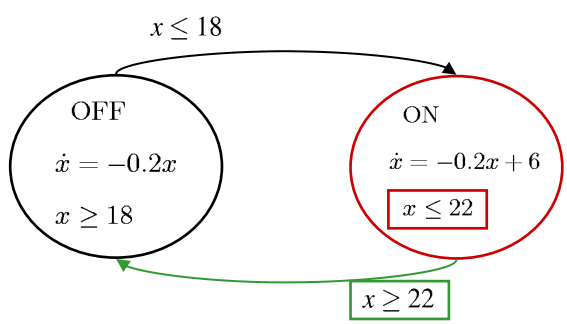
If x exits the Domain then a discrete transition must occur

Guard conditions enable the discrete transition

x = 18

→ transition from OFF to ON occurs when x = 18

EXAMPLE: THERMOSTAT



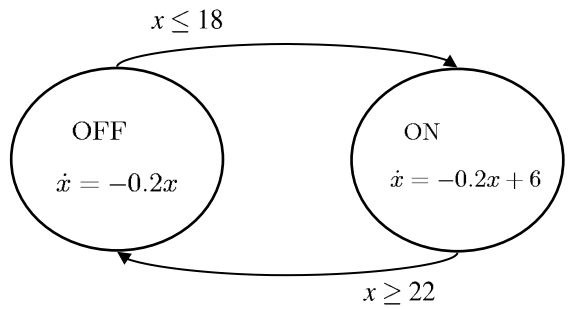
If x exits the Domain then a discrete transition must occur

Guard conditions enable the discrete transition

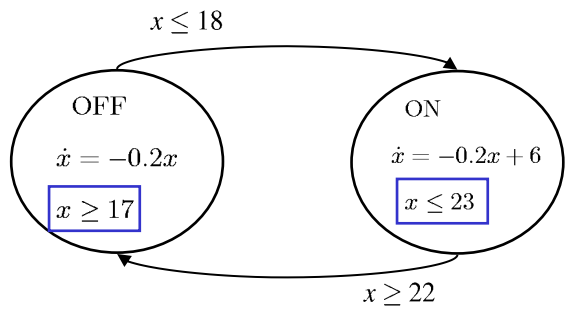
x = 22

→ transition from ON to OFF occurs when x = 22

EXAMPLE: THERMOSTAT



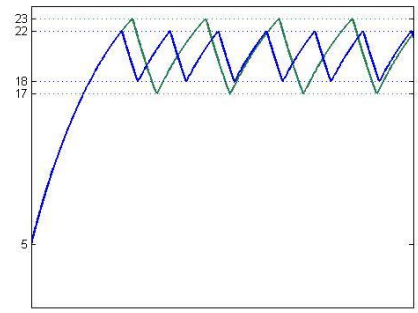
EXAMPLE: THERMOSTAT



Domain(OFF) ∩ Guard(OFF,ON) = [17, ∞) ∩ (-∞, 18] = [17,18]
→ transition from OFF to ON for x ∈ [17,18]

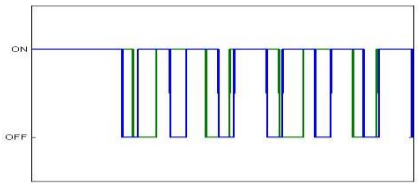
Domain(ON) ∩ Guard(ON,OFF) = (-∞, 23] ∩ [22, ∞) = [22,23]
→ transition from ON to OFF for x ∈ [22,23]

EXAMPLE: THERMOSTAT



evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

multiple executions starting from an initial condition
→ Non-deterministic hybrid system

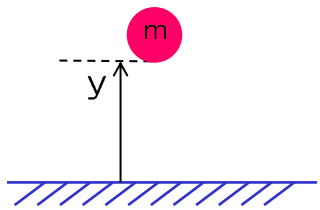


evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON

EXAMPLE: BOUNCING BALL

2 situations:
a) ball flying in the air
b) ball hitting the ground

EXAMPLE: BOUNCING BALL

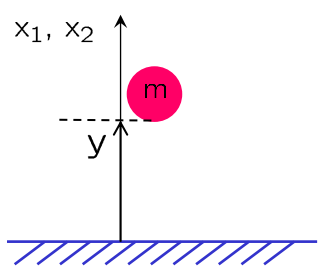


2 situations:

a) ball flying in the air

b) ball hitting the ground

EXAMPLE: BOUNCING BALL



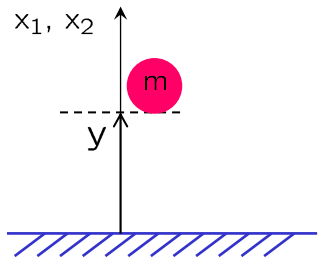
2 situations:

a) ball flying in the air

b) ball hitting the ground

State of the system given by position x_1 := y and velocity x_2 := dy/dt

EXAMPLE: BOUNCING BALL



2 situations:

a) ball flying in the air

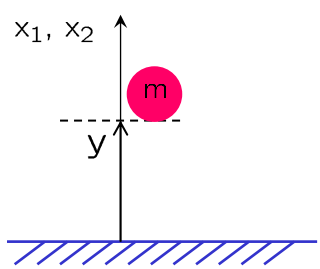
b) ball hitting the ground

State of the system given by position x_1 := y and velocity x_2 := dy/dt

$[x_1 > 0]$ or $[x_1 = 0$ and $x_2 \geq 0]$

$dx_1/dt = x_2$
$dx_2/dt = -g$

EXAMPLE: BOUNCING BALL



2 situations:

a) ball flying in the air

b) ball hitting the ground

State of the system given by position x_1 := y and velocity x_2 := dy/dt

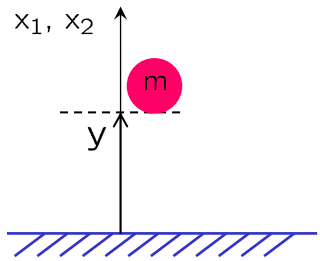
$[x_1 > 0]$ or $[x_1 = 0$ and $x_2 \geq 0]$

$dx_1/dt = x_2$
$dx_2/dt = -g$

$$\Updownarrow$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -g \end{bmatrix}$$

EXAMPLE: BOUNCING BALL



2 situations:

a) ball flying in the air

b) ball hitting the ground

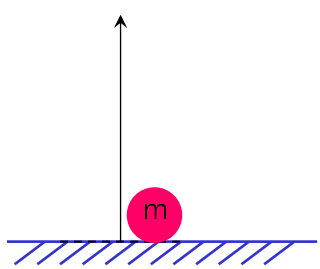
State of the system given by position x_1 := y and velocity x_2 := dy/dt

$[x_1 > 0]$ or $[x_1 = 0$ and $x_2 \geq 0]$

$dx_1/dt = x_2$
$dx_2/dt = -g$

$$x := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \dot{x} = f(x)$$

EXAMPLE: BOUNCING BALL



2 situations:

a) ball flying in the air

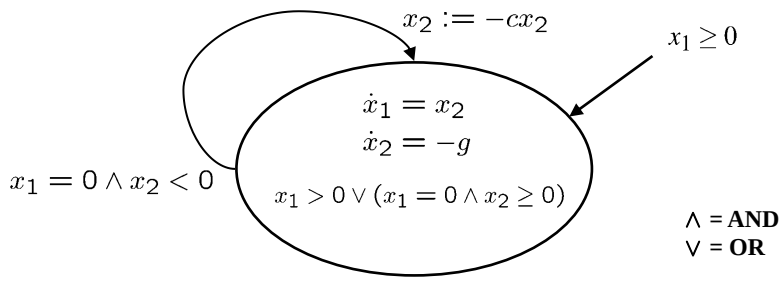
b) ball hitting the ground

State of the system given by position x_1 := y and velocity x_2 := dy/dt

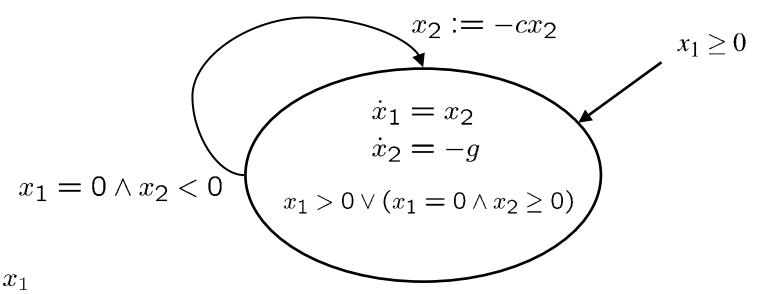
$x_1 = 0$ and $x_2 < 0$

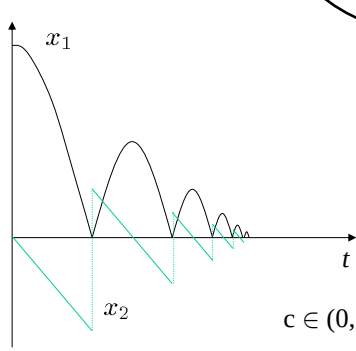
$x_1^+(t) = x_1^-(t) = 0$
$x_2^+(t) = -c\, x_2^-(t)$, $c \in [0,1]$

EXAMPLE: BOUNCING BALL



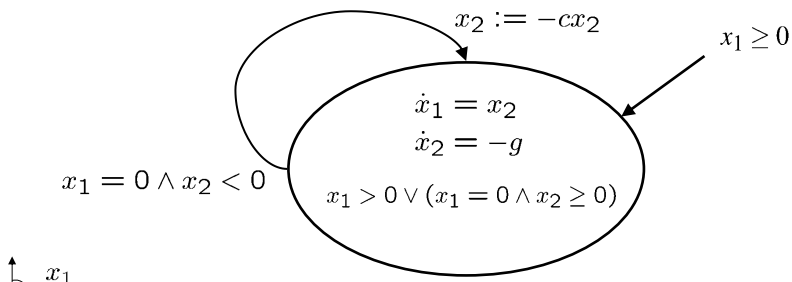
EXAMPLE: BOUNCING BALL

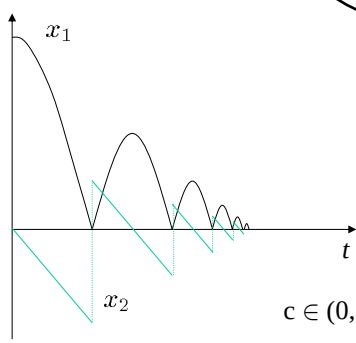




c ∈ (0,1)

EXAMPLE: BOUNCING BALL





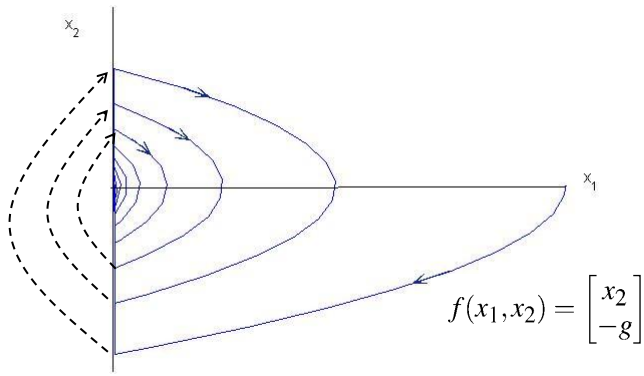
c ∈ (0,1)

Infinite number of transitions in finite time
→ Zeno hybrid system

(Paradox of Achilles and the turtle by Zeno of Elea born around 490 BC)

EXAMPLE: BOUNCING BALL

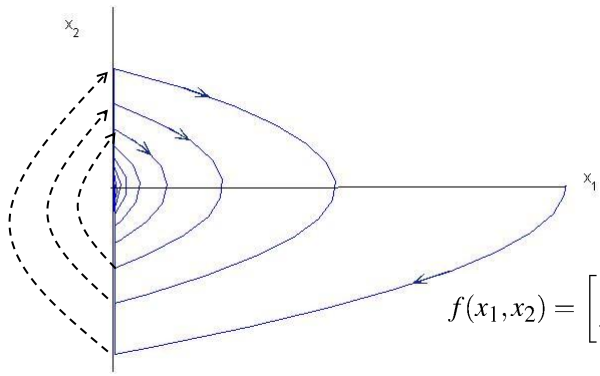
Phase plot:
plot of the evolution of the continuous state x in the state space X = $\Re^2$ (time info lost, suitable for qualitative analysis)



EXAMPLE: BOUNCING BALL

Phase plot:

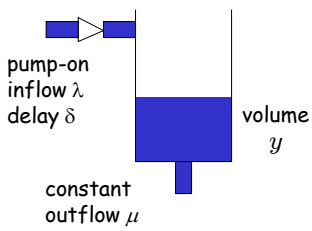
plot of the evolution of the continuous state x in the state space X = $\Re^2$ (time info lost, suitable for qualitative analysis)



$$f(x_1, x_2) = \begin{bmatrix} x_2 \\ -g \end{bmatrix}$$

$\{(x_1,x_2) \in \Re^2 : x_1 \geq 0\}$ invariant set

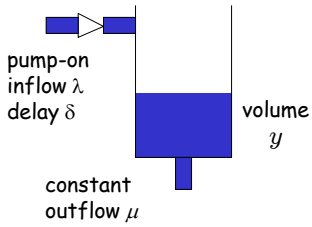
EXAMPLE: TANK SYSTEM

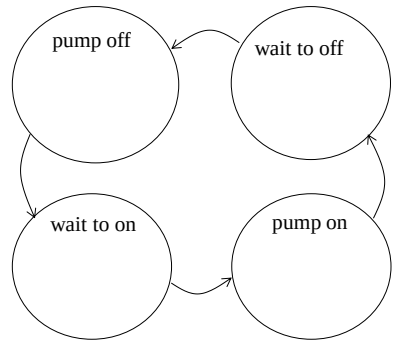


Goal: prevent the tank from emptying or filling up

Strategy: switch the pump on/off when the level drops below/goes beyond some threshold value

EXAMPLE: TANK SYSTEM

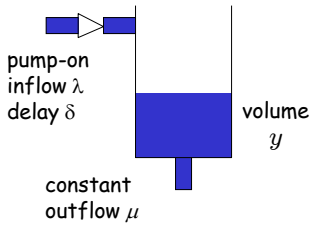


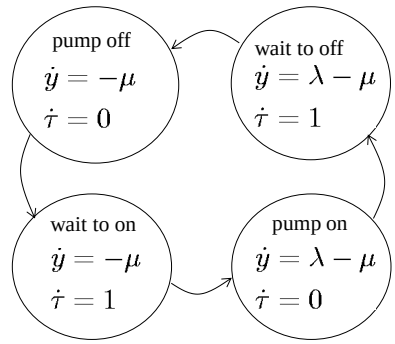


Goal: prevent the tank from emptying or filling up

Strategy: switch the pump on/off when the level drops below/goes beyond some threshold value

EXAMPLE: TANK SYSTEM

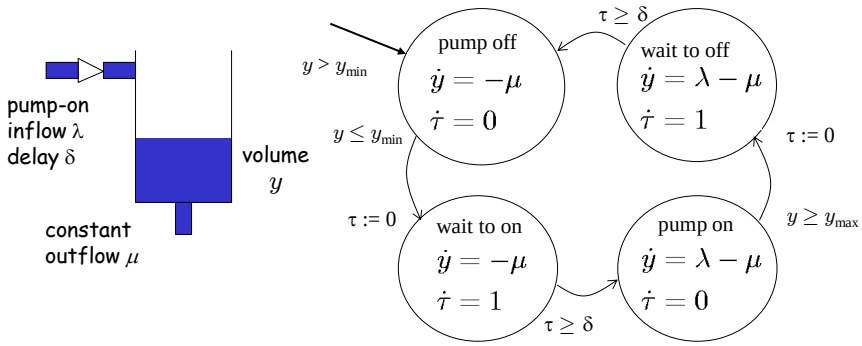




Goal: prevent the tank from emptying or filling up

Strategy: switch the pump on/off when the level drops below/goes beyond some threshold value

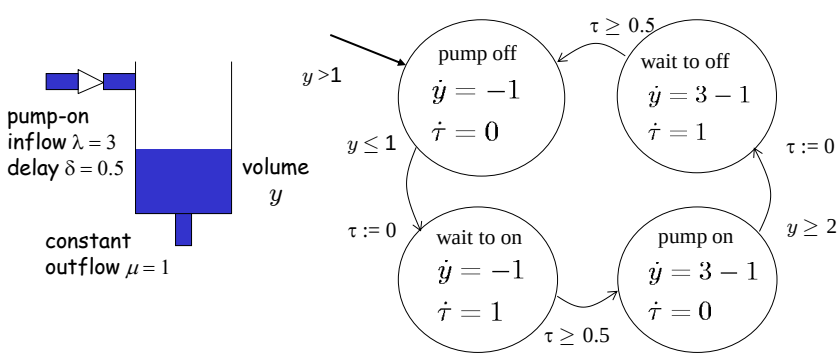
EXAMPLE: TANK SYSTEM



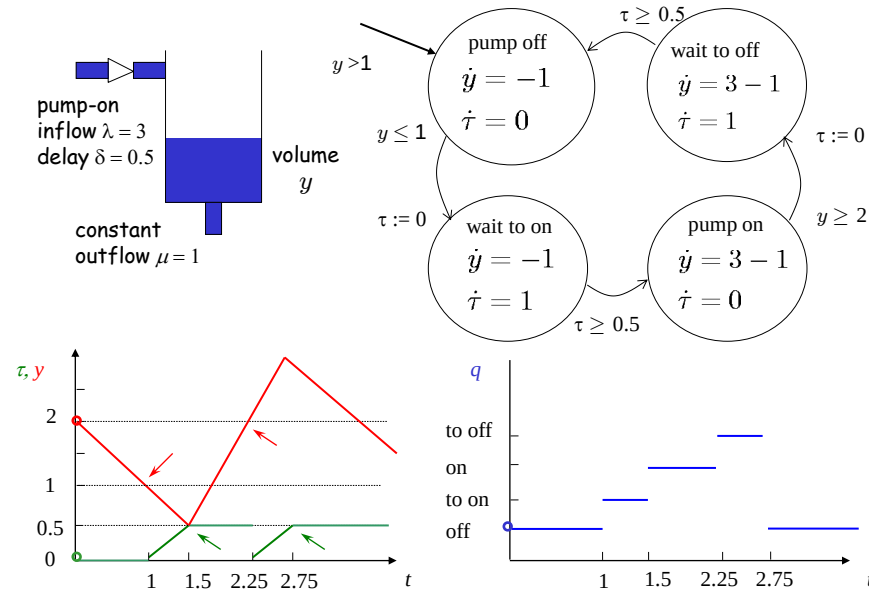
Goal: prevent the tank from emptying or filling up

Strategy: switch the pump on/off when the level drops below/goes beyond some threshold value

EXAMPLE: TANK SYSTEM



EXAMPLE: TANK SYSTEM



Formal modeling of hybrid systems

FORMAL MODELING OF HYBRID SYSTEMS

Brief review of

- Finite automata
- Ordinary differential equations

Definition of

- Hybrid automata

FORMAL MODELING OF HYBRID SYSTEMS

Brief review of

- Finite automata
- Ordinary differential equations

Definition of

- Hybrid automata

FINITE AUTOMATA

A finite automaton is a mathematical model of an event-driven discrete system with finite state space:

$M = (Q,\Sigma,R)$

$Q = \{q_1,q_2, \ldots,q_N\} \equiv$ finite set of states
$\Sigma = \{a, b, c,\ldots\} \equiv$ finite set (alphabet) of input symbols (events)
$R: Q \times \Sigma \rightarrow 2^Q \equiv$ transition (set-valued) function

where 2^Q denotes the power set of Q = set of all subsets of Q

$R(q,\sigma) \subset Q$ is the set of all states the system can transit to from state q under the input symbol σ

FINITE AUTOMATA

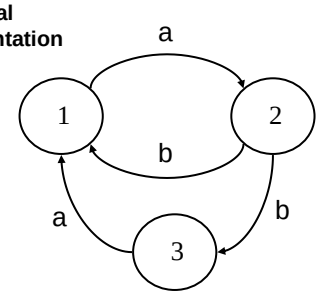
$M = (Q,\Sigma,R)$

$Q = \{q_1,q_2, \ldots,q_N\} \equiv$ finite set of states
$\Sigma = \{e_1, e_2,\ldots,e_m\} \equiv$ finite set (alphabet) of input symbols (events)
$R: Q \times \Sigma \rightarrow 2^Q \equiv$ transition (set-valued) function

Example: $Q = \{1, 2, 3\}$ $\Sigma = \{a, b\}$

s	σ	R(s,σ)
1	a	{2}
1	b	∅
2	a	∅
2	b	{1,3}
3	a	{1}
3	b	∅

Graphical representation



FINITE AUTOMATA: EXECUTION

M = (Q,Σ,R)

$Q = \{q_1, q_2, \ldots, q_N\} \equiv$ finite set of states
$\Sigma = \{e_1, e_2, \ldots, e_m\} \equiv$ finite set (alphabet) of input symbols (events)
$R: Q \times \Sigma \rightarrow 2^Q \quad \equiv$ transition (set-valued) function

Given an initial set of states *Init* $\subset$ Q,

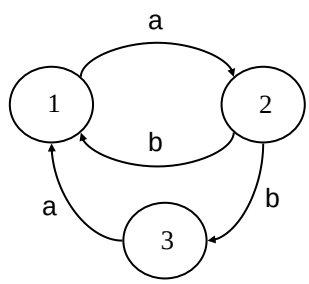
M accepts an input sequence $\{\sigma_0, \sigma_1, \sigma_2, \ldots\}$, with $\sigma_i \in \Sigma, \forall\, i \geq 0$, if there exists a sequence of states $\{s_0, s_1, s_2, \ldots\}, s_i \in Q, \forall\, i \geq 0$, such that:

- $s_0 \in$ *Init*
- $s_{i+1} \in R(s_i, \sigma_i) \; \forall\; i \geq 0$

$\{s_0, s_1, s_2, \ldots\}$ in the definition above is called an execution of M under the input sequence $\{\sigma_0, \sigma_1, \sigma_2, \ldots\}$

FINITE AUTOMATA: EXECUTION

- EXAMPLE



Init $= \{1\}$

1,2,1 is an execution under input a, b
1,2,3,1,2 under input a,b,a,a

a,b,b is not accepted as an input

FINITE AUTOMATA: LANGUAGE

M = (Q,Σ,R)

$Q = \{q_1, q_2, \ldots, q_N\} \equiv$ finite set of states
$\Sigma = \{e_1, e_2, \ldots, e_m\} \equiv$ finite set (alphabet) of input symbols (events)
$R: Q \times \Sigma \rightarrow 2^Q \quad \equiv$ transition (set-valued) function

A string with alphabet Σ
is a finite sequence of input symbols $\{\sigma_0, \sigma_1, \sigma_2, \ldots\}$ (ε: empty string).

Given *Init* ⊂ Q, the language L(M) accepted by M
is the set of all input strings $\{\sigma_0, \sigma_1, \sigma_2, \ldots\}$ that are accepted by M with *Init* as initial set of states

Formal language theory: is there M that accepts a given language? Smallest M?
Do two automata accept the same languange?

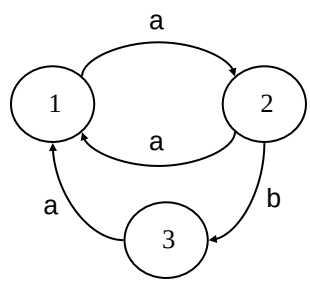
FINITE AUTOMATA: DETERMINISTIC VS. NONDETERMINISTIC

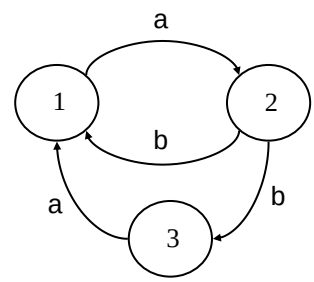
A finite automaton M = (Q,Σ,R) is deterministic if

$|R(q,\sigma)| \leq 1$ for all $q \in Q$, $\sigma \in \Sigma$, $|R(q,\sigma)|$ = cardinality of $R(q,\sigma)$

→ at most one execution given an initial state and an input string

Otherwise, it is nondeterministic.





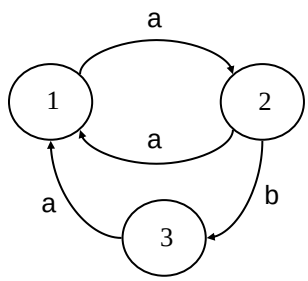
FINITE AUTOMATA: DETERMINISTIC VS. NONDETERMINISTIC

A finite automaton M = (Q,Σ,R) is deterministic if

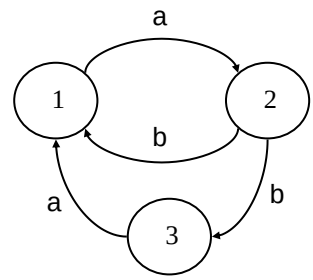
$|R(q,\sigma)| \leq 1$ for all $q \in Q$, $\sigma \in \Sigma$, $|R(q,\sigma)|$ = cardinality of $R(q,\sigma)$

→ at most one execution given an initial state and an input string

Otherwise, it is nondeterministic.



deterministic

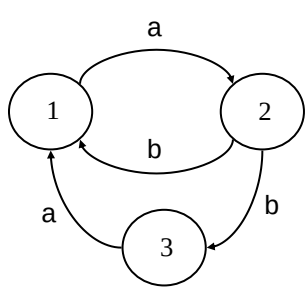


nondeterministic

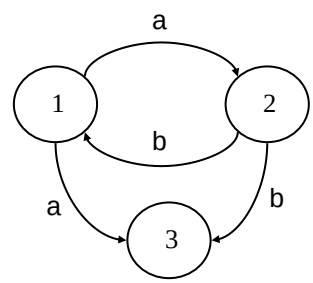
FINITE AUTOMATA: BLOCKING STATE

A finite automaton M = (Q,Σ,R) has a blocking state $q \in Q$ if

$R(q,\sigma) = \emptyset$, for all $\sigma \in \Sigma$



no blocking state



blocking state: q = 3

FORMAL MODELING OF HYBRID SYSTEMS

Brief review of

- Finite automata
- Ordinary differential equations

Definition of

- Hybrid automata

ORDINARY DIFFERENTIAL EQUATIONS

An ordinary differential equation is a mathematical model of a continuous state continuous time system:

$$\dot{x}(t) = f(x(t))$$

$X = \Re^n$	$\equiv$ state space
$f: \Re^n \rightarrow \Re^n$	$\equiv$ vector field (assigns a "velocity" vector to each x)

ORDINARY DIFFERENTIAL EQUATIONS

An ordinary differential equation is a mathematical model of a continuous state continuous time system:

$$\dot{x}(t) = f(x(t))$$

$X = \Re^n$	$\equiv$ state space
$f: \Re^n \rightarrow \Re^n$	$\equiv$ vector field (assigns a "velocity" vector to each x)

Given an initial set of states *Init* $\subset$ X,

an execution (solution in the sense of Caratheodory) over the time interval [0,T) is a function x: [0,T) $\rightarrow \Re^n$ such that:

- x(0)$\in$ *Init*
- x is continuous and piecewise differentiable
- $x(t) = x(0) + \int_0^t f(x(\tau))d\tau,\ \forall t \in [0,T)$

If x is a solution, dx/dt(t) = f(x(t)) for all t for which x is differentiable

ODE SOLUTION: WELL-POSEDNESS?

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0$$

- Existence?

  local existence (solution exists over [0,δ))

  → non blocking

  global existence (solution exists over [0,∞))

  → non-zeno (solution can be extended over an arbitrarily long time horizon)

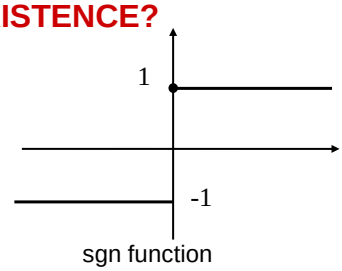
- Uniqueness?

  → deterministic

Both critical for simulation...

ODE SOLUTION: EXISTENCE?

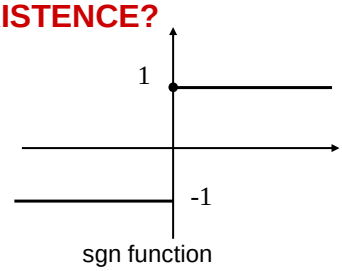
- $\dot{x} = -sgn(x), \quad x(0) = 0$



sgn function

ODE SOLUTION: EXISTENCE?

- $\dot{x} = -sgn(x), \quad x(0) = 0$

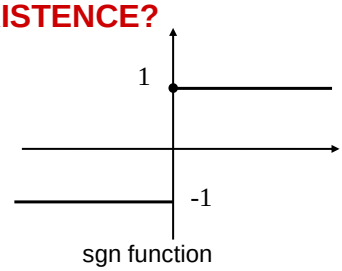


sgn function

No solution if x(0) = 0,
because on any interval [0,δ), x cannot:
remain zero, become positive, or become negative

ODE SOLUTION: EXISTENCE?

- $\dot{x} = -sgn(x), \quad x(0) = 0$



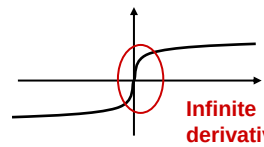
sgn function

No solution if x(0) = 0,
because on any interval [0,δ), x cannot:
remain zero, become positive, or become negative

Theorem [local existence]
If f:$\Re^n \to \Re^n$ is continuous, then $\forall x_0$ there exists at least a solution with $x(0)=x_0$ defined on some [0,δ).

ODE SOLUTION: UNIQUENESS?

- $\dot{x} = x^{1/3}, \quad x(0) = 0$



x(t)=0 and x(t) = $(2/3t)^{3/2}$ are two solutions

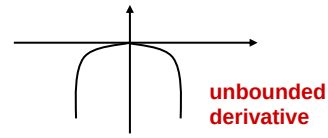
Def. f:$\Re^n \to \Re^n$ is Lipschitz continuous, if in any bounded set A of $\Re^n$ there exists a constant L such that

$||f(x_1)-f(x_2)|| \leq L\ ||x_1-x_2||, \ \forall\ x_1,x_2 \in A$

Theorem [local existence and uniqueness]
If f:$\Re^n \to \Re^n$ is Lipschitz continuous, then $\forall\ x_0$ there exists a single solution with $x(0)=x_0$ defined on some [0,δ).

ODE SOLUTION: GLOBAL EXISTENCE AND UNIQUENESS?

- $\dot{x} = -x^2, \quad x(0) = -1$



x(t)=1/(t-1) local solution on [0,1) (tends to $-\infty$ as $t \uparrow 1$)

finite escape

Def. f:$\Re^n \to \Re^n$ is globally Lipschitz continuous, if there exists a constant L such that

$||f(x_1)-f(x_2)|| \leq L\ ||x_1-x_2||, \forall\ x_1,x_2 \in \Re^n$

Theorem [global existence and uniqueness]

If f:$\Re^n \to \Re^n$ is globally Lipschitz continuous, then $\forall\ x_0$ there exists a single solution with $x(0)=x_0$ defined on $[0,\infty)$.

FORMAL MODELING OF HYBRID SYSTEMS

Brief review of

- Finite automata
- Ordinary differential equations

Definition of

- Hybrid automata

HYBRID AUTOMATON

A hybrid automaton is a mathematical model for a dynamical system whose state s =(q,x) consists of two components:

- continuous state x taking values in $X = \Re^n$
- discrete state (mode) q taking values in $Q = \{q_1, q_2, \ldots, q_N\}$

HYBRID AUTOMATON

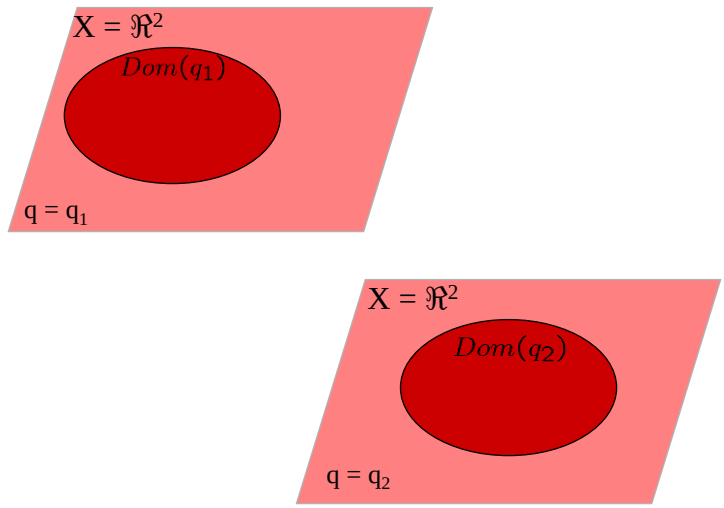
A hybrid automaton is a mathematical model for a dynamical system whose state s =(q,x) consists of two components:

- continuous state x taking values in $X = \Re^n$
- discrete state (mode) q taking values in $Q = \{q_1, q_2, \ldots, q_N\}$

Hybrid state space:

$S = Q \times X$ (N copies of X)
for each value of q, the admissible values for x are only a subset of X

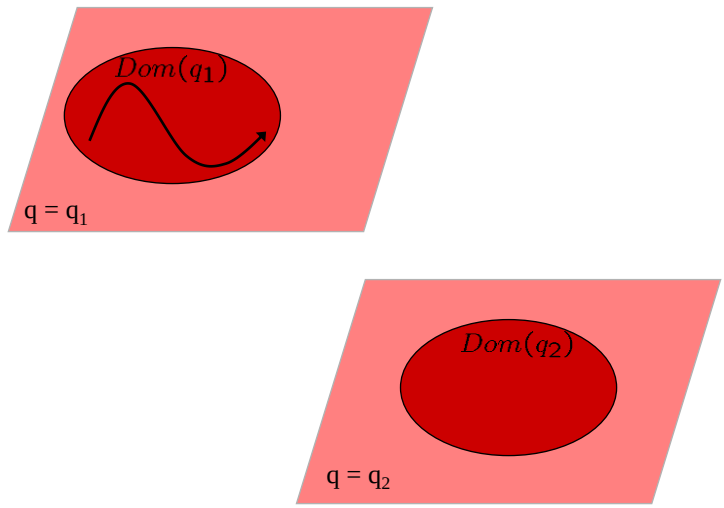
→ Dom(q) = domain for x within mode q



HYBRID AUTOMATON

Dynamics of q and x are correlated:

- x follows an ODE within Dom(q). The ODE depends on the value of mode q



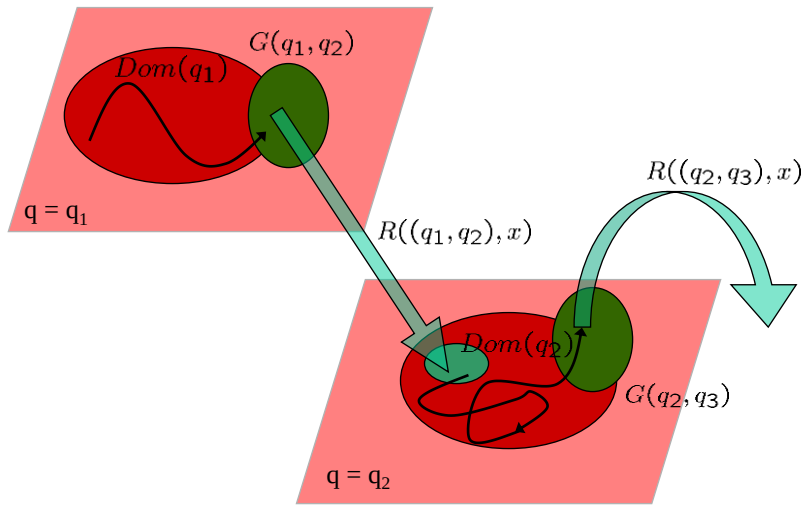
HYBRID AUTOMATON

Dynamics of q and x are correlated:

- x follows an ODE within Dom(q). The ODE depends on the value of mode q
- q follows a finite automaton with transitions triggered by events given by x entering subsets of X called Guards

  → G(q,q') = values for x enabling the transition from q to q'
- when a transition from q to q' takes place, x is reset within Dom(q') by some Reset map

  → R((q,q'),x) = reset values for x^+ when a transition from q to q' occurs at x



HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

H = (Q,X,f,*Init*,Dom,E,G,R)

- Q = $\{q_1, q_2, \ldots\}$ is a set of discrete states (modes)
- X = $\Re^n$ is the continuous state space

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
  - for each $q \in Q$, $f(q,\cdot)$ is the vector field of the ODE governing the evolution of x
  - for each $q \in Q$, $f(q,\cdot)$ is assumed to be globally Lipschitz continuous

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- Dom: $Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- Dom: $Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X
- $E \subseteq Q \times Q$ is a set of transitions (edges)
  - each $e \in E$ is of the form e=(q,q') and represents a transition from q to q' (edge in the directed graph with vertex set Q)

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- $Dom: Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X
- $E \subseteq Q \times Q$ is a set of transitions (edges)
- $G: E \rightarrow 2^X$ is a set of guards (guard condition)
  - For each e=(q,q'), whenever x reaches G(e) from within Dom(q), transition e is enabled

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- $Dom: Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X
- $E \subseteq Q \times Q$ is a set of transitions (edges)
- $G: E \rightarrow 2^X$ is a set of guards (guard condition)

- $R: E \times X \rightarrow 2^X$ is a set of reset maps
  - for each $e=(q,q') \in E$ and $x \in Dom(q)$, $R(e,x) \subseteq Dom(q')$ is the set of values that x can be reset to after the transition e

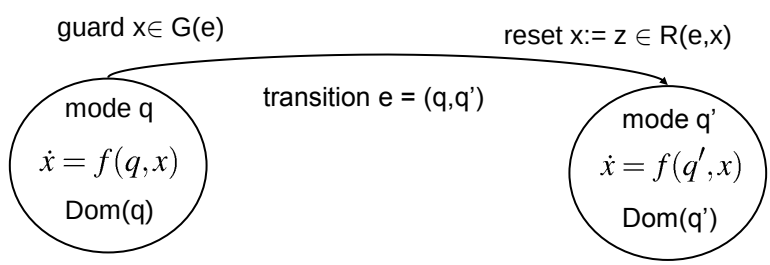
HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

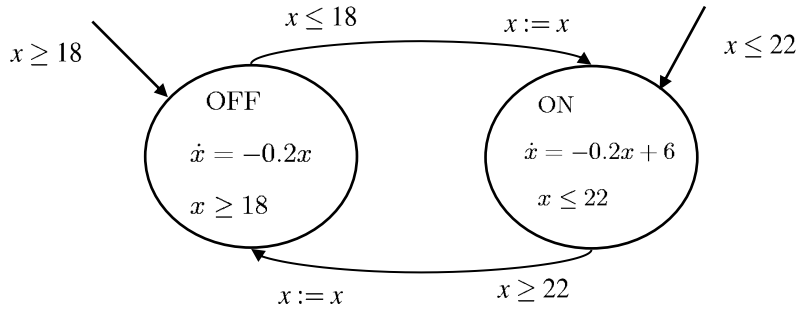
$$H = (Q, X, f, Init, Dom, E, G, R)$$

- $Q = \{q_1, q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- $Dom: Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X
- $E \subseteq Q \times Q$ is a set of transitions (edges)
- $G: E \rightarrow 2^X$ is a set of guards (guard condition)
- $R: E \times X \rightarrow 2^X$ is a set of reset maps

HYBRID AUTOMATA: GRAPHICAL REPRESENTATION



EXAMPLE: THERMOSTAT



H = (Q,X,f,*Init*,Dom,E,G,R)

Q = {OFF, ON}; X = $\Re$;

f(OFF,x) = -0.2 x; f(ON, x) = -0.2 x+6

Init = {(OFF,x) : x$\geq$ 18} $\cup$ {(ON,x) : x$\leq$ 22}

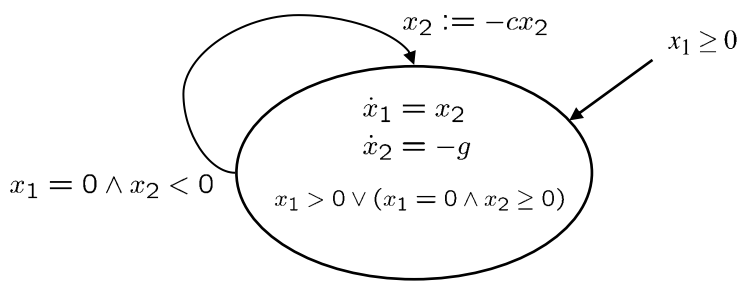
Dom(OFF) = [18,∞); Dom(ON) = (-∞, 22]

E = {(OFF,ON),(ON,OFF)}

G((OFF,ON)) = (-∞,18]; G((ON,OFF)) = [22,∞)

R(e,x) = {x} for any e$\in$ E

EXAMPLE: BOUNCING BALL



H = (Q,X,f,*Init*,Dom,E,G,R)

Q = {fly}; X = $\Re^2$;

f(fly,x) = [x_2, -g]^T

Init = {(fly,(x_1,x_2)) : $x_1 \geq 0$}

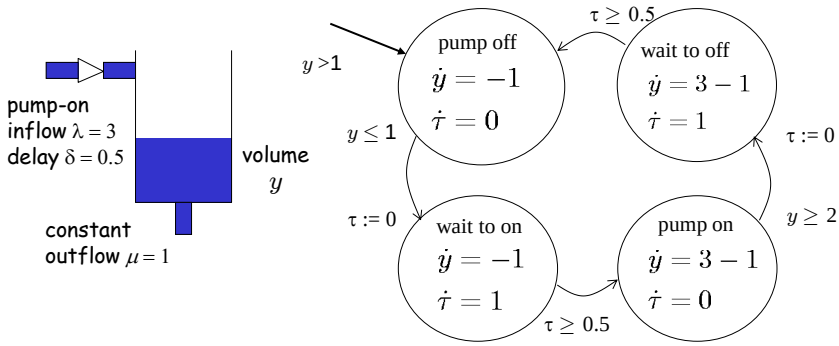
Dom(fly) = {(x_1,x_2) : x_1>0}$\cup$ {(x_1,x_2) : x_1=0, $x_2 \geq 0$}

E = {(fly,fly)}

G(e) = G((fly,fly)) = {(x_1,x_2) : x_1=0, x_2 < 0}

R(e,(x_1,x_2)) = {(x_1, -c x_2)}

EXAMPLE: TANK SYSTEM



Exercise.

Hybrid automaton: execution

HYBRID AUTOMATA: EXECUTION

To define an execution of a hybrid system, we need to introduce first the notions of

- hybrid time set
- hybrid trajectory
  (which includes hybrid time set among its components)

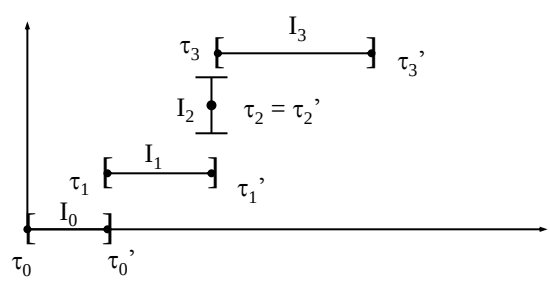
We shall see that
an execution of a hybrid automaton H is a hybrid trajectory that is "consistent" with the definition of H

HYBRID TIME SET

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

$\tau = \{I_i, i=0,1,\ldots, M\}$ such that

- $I_i = [\tau_i, \tau_i']$ for $i < M$
- $I_M = [\tau_M, \tau_M']$ or $I_M = [\tau_M, \tau_M')$ if $M<\infty$
- $\tau_i' = \tau_{i+1}$
- $\tau_i \leq \tau_i'$

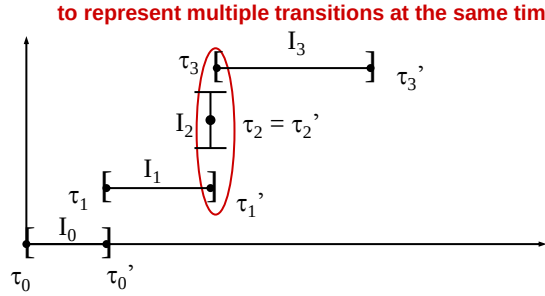


HYBRID TIME SET

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

$\tau = \{I_i, i=0,1,\ldots, M\}$ such that

- $I_i = [\tau_i, \tau_i']$ for $i < M$ — **τ_i represent times of discrete transitions**
- $I_M = [\tau_M, \tau_M']$ or $I_M = [\tau_M, \tau_M')$ if $M<\infty$
- $\tau_i' = \tau_{i+1}$ — **consecutive intervals, without gaps**
- $\tau_i \leq \tau_i'$ — **intervals can be degenerate to represent multiple transitions at the same time**



HYBRID TIME SET

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

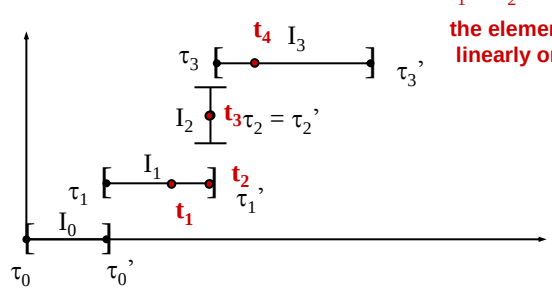
$\tau = \{I_i, i=0,1,\ldots, M\}$ such that

- $I_i = [\tau_i, \tau_i']$ for $i < M$
- $I_M = [\tau_M, \tau_M']$ or $I_M = [\tau_M, \tau_M')$ if $M<\infty$
- $\tau_i' = \tau_{i+1}$
- $\tau_i \leq \tau_i'$

$t_1 \prec t_2$: "t_1 precedes t_2"

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

the elements of τ are linearly ordered



HYBRID TIME SET: LENGTH

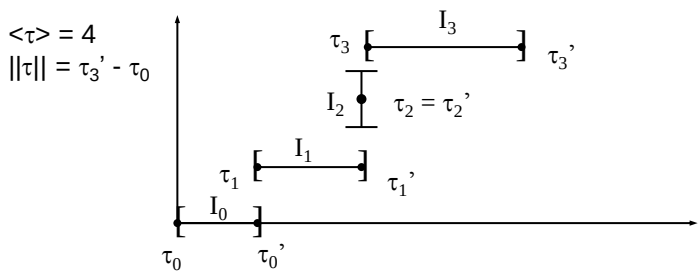
Two notions of length for a hybrid time set $\tau = \{I_i, i=0,1,\ldots, M\}$:

- Discrete extent:

$$<\tau> = M+1$$ **number of discrete transitions**

- Continuous extent:

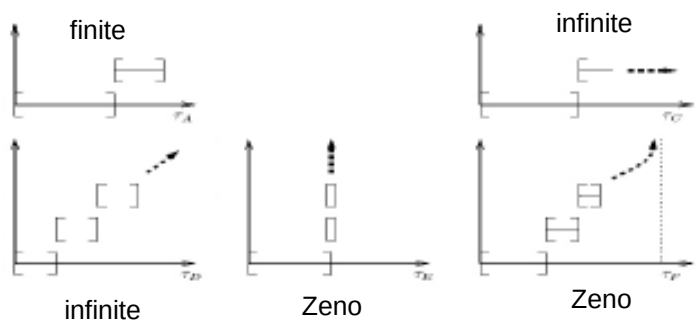
$$||\tau|| = \Sigma_{i=0,1,..,M} |\tau_i' - \tau_i|$$ **total duration of intervals in τ**



HYBRID TIME SET: CLASSIFICATION

A hybrid time set $\tau = \{I_i, i=0,1,\ldots, M\}$ is

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



HYBRID TRAJECTORY

A hybrid trajectory is a triple (τ, q, x) that consists of:

- A hybrid time set $\tau = \{I_i, i=0,1,\ldots, M\}$
- Two sequences of functions $q = \{q_i(\cdot), i=0,1,\ldots, M\}$ and $x = \{x_i(\cdot), i=0,1,\ldots, M\}$ such that

  $q_i: I_i \to Q$

  $x_i: I_i \to X$

HYBRID TRAJECTORY

A hybrid trajectory is a triple (τ, q, x) that consists of:

- A hybrid time set $\tau = \{I_i, i=0,1,\ldots, M\}$
- Two sequences of functions $q = \{q_i(\cdot), i=0,1,\ldots, M\}$ and $x = \{x_i(\cdot), i=0,1,\ldots, M\}$ such that

  $q_i: I_i \to Q$

  $x_i: I_i \to X$

Remarks:

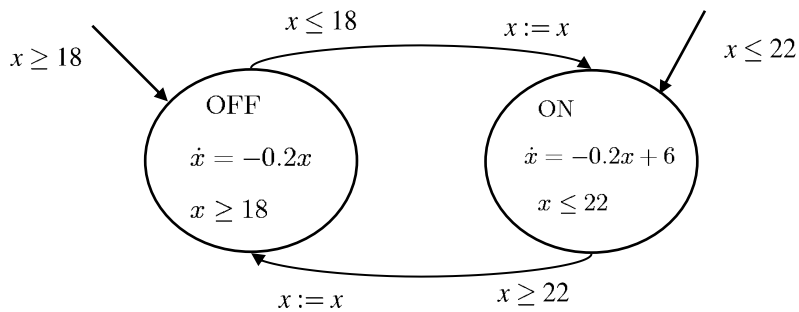
- mixture of the two notions of continuous and discrete evolution
- the hybrid signals x and q can take multiple values at the same time instant

HYBRID AUTOMATA: EXECUTION

A hybrid trajectory (τ, q, x) is an execution (solution) of the hybrid automaton $H = (Q,X,f,Init,Dom,E,G,R)$ if it satisfies the following conditions:

- Initial condition: $(q_0(\tau_0), x_0(\tau_0)) \in Init$
- Continuous evolution:
  for all i such that $\tau_i < \tau_i'$
  - q_i: $I_i \rightarrow Q$ is constant
  - x_i: $I_i \rightarrow X$ is the solution to the ODE associated with $q_i(\tau_i)$
  - $x_i(t) \in Dom(q_i(\tau_i))$, $t \in [\tau_i, \tau_i')$
- Discrete evolution:
  - $(q_i(\tau_i'), q_{i+1}(\tau_{i+1})) \in E$ transition is feasible
  - $x_i(\tau_i') \in G(q_i(\tau_i'), q_{i+1}(\tau_{i+1}))$ guard condition satisfied
  - $x_{i+1}(\tau_{i+1}) \in R((q_i(\tau_i'), q_{i+1}(\tau_{i+1})), x_i(\tau_i'))$ reset condition satisfied

EXAMPLE: THERMOSTAT



$H = (Q,X,f,Init,Dom,E,G,R)$

$Q = \{OFF, ON\}$; $X = \Re$;

$f(OFF,x) = -0.2\,x$; $f(ON, x) = -0.2\,x+6$

$Init = \{(OFF,x) : x \geq 18\} \cup \{(ON,x) : x \leq 22\}$

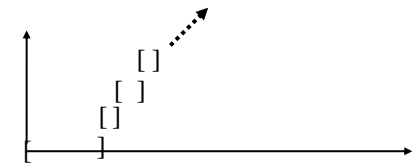
$Dom(OFF) = [18,\infty)$; $Dom(ON) = (-\infty, 22]$

$E = \{(OFF,ON),(ON,OFF)\}$

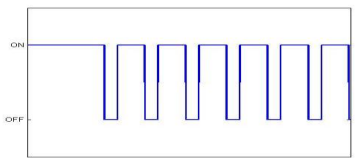
$G((OFF,ON)) = (-\infty,18]$; $G((ON,OFF)) = [22,\infty)$

$R(e,x) = \{x\}$ for any $e \in E$

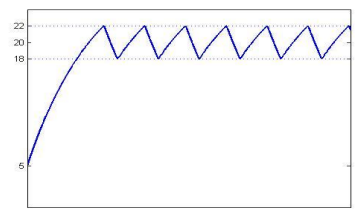
EXAMPLE: THERMOSTAT



infinite hybrid time set

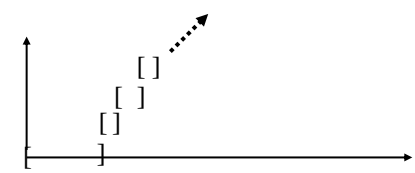


evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON



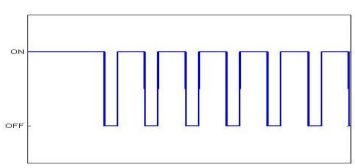
evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

EXAMPLE: THERMOSTAT

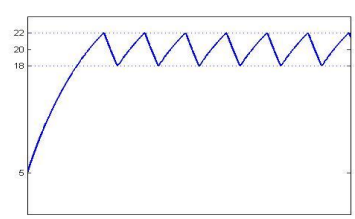


execution accepted by the thermostat hybrid automaton

Infinite hybrid time set



evolution of the heater status starting from the initial condition x(0) = 5 with the heater ON



evolution of the temperature x starting from the initial condition x(0) = 5 with the heater ON

HYBRID AUTOMATA EXECUTIONS

What can go wrong?

HYBRID AUTOMATA EXECUTIONS

What can go wrong?

Problems of the ODE solution (existence, uniqueness, finite escape) avoided by the globally Lipschitz assumption

HYBRID AUTOMATA EXECUTIONS

What can go wrong?

Problems of the ODE solution (existence, uniqueness, finite escape) avoided by the globally Lipschitz assumption

Problems due the hybrid nature!!!

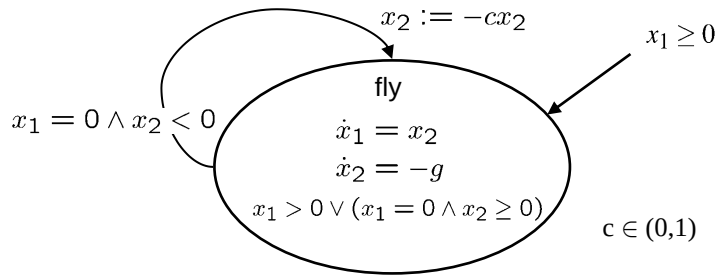
- Zeno
- chattering
- blocking
- nondeterministic

HYBRID EXECUTION: ZENO

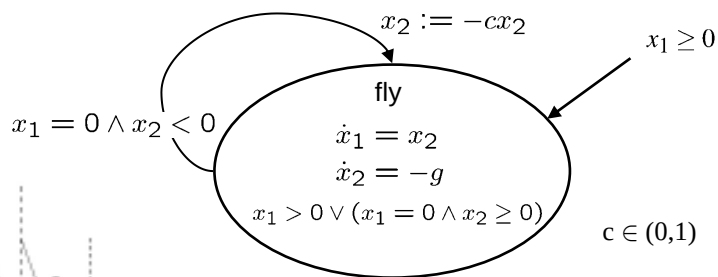
Let (τ, q, x) be an execution of H.
(τ, q, x) is called Zeno execution if

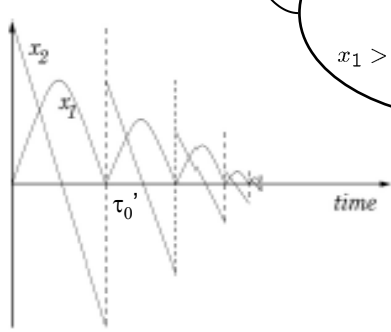
$\tau = \{I_i, i=0,1,\ldots, M\}$ is Zeno (infinite number of discrete transitions in finite time)

EXAMPLE: BOUNCING BALL



EXAMPLE: BOUNCING BALL

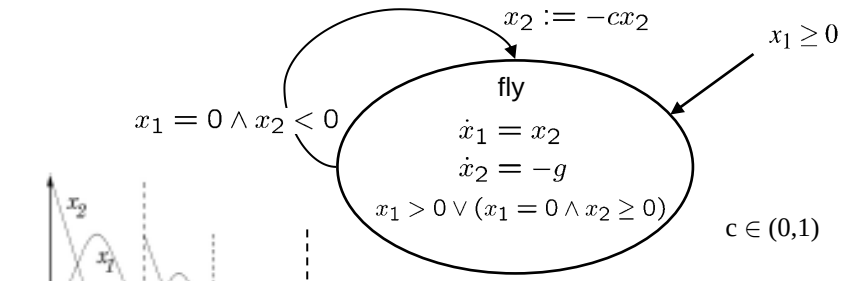




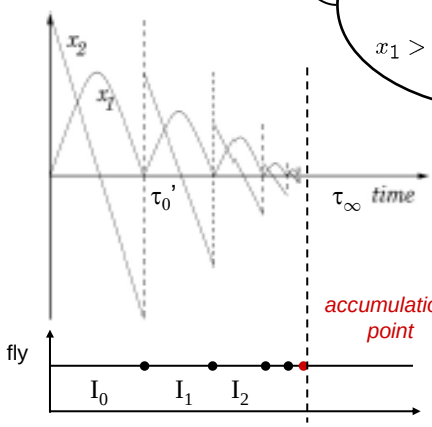
$x_1(0) = 0, x_2(0) > 0$

$\tau_0' = 2\, x_2(0)/g$

EXAMPLE: BOUNCING BALL



$c \in (0,1)$



$x_1(0) = 0, x_2(0) > 0$
$\tau_0' = 2\, x_2(0)/g$

$\tau_\infty = \tau_0' + c\,\tau_0' + c^2\tau_0' \ldots$
$= 1/(1-c)\,\tau_0'$
$= 1/(1-c)\,(2\, x_2(0)/g)$

HYBRID EXECUTION: CHATTERING

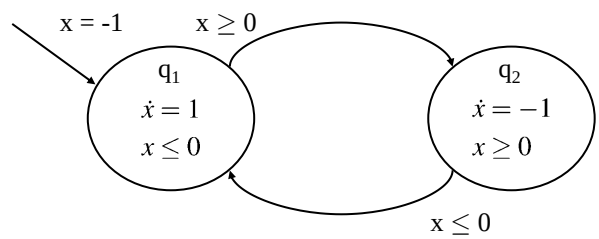
Let (τ, q, x) be an execution of H.
(τ, q, x) is called chattering execution if

- $\tau = \{I_i, i=0,1,\ldots, M\}$ is Zeno (infinite number of discrete transitions in finite time)
- After some $k \geq 0$, all intervals I_i, $i \geq k$, are singletons

HYBRID EXECUTION: CHATTERING

Let (τ, q, x) be an execution of H.
(τ, q, x) is called chattering execution if

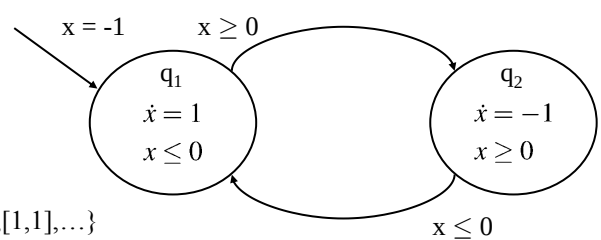
- $\tau = \{I_i, i=0,1,\ldots, M\}$ is Zeno (infinite number of discrete transitions in finite time)
- After some $k \geq 0$, all intervals I_i, $i \geq k$, are singletons



HYBRID EXECUTION: CHATTERING

Let (τ, q, x) be an execution of H.
(τ, q, x) is called chattering execution if

- $\tau = \{I_i, i=0,1,\ldots, M\}$ is Zeno (infinite number of discrete transitions in finite time)
- After some $k \geq 0$, all intervals I_i, $i \geq k$, are singletons



Execution:

$\tau = \{[0,1],[1,1],[1,1],\ldots\}$

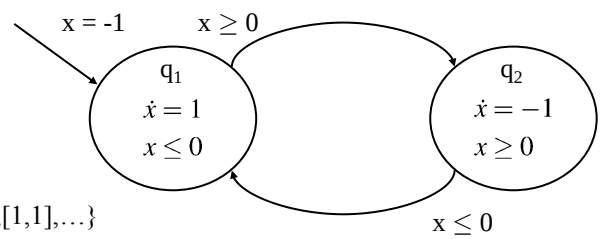
$q = \{q_1, q_2, q_1, q_2, \ldots\}$

$x = \{t\text{-}1, 0, 0, 0, \ldots\}$

HYBRID EXECUTION: CHATTERING

Let (τ, q, x) be an execution of H.
(τ, q, x) is called chattering execution if

- $\tau = \{I_i, i=0,1,\ldots, M\}$ is Zeno (infinite number of discrete transitions in finite time)
- After some $k \geq 0$, all intervals I_i, $i \geq k$, are singletons



Execution:
$\tau = \{[0,1],[1,1],[1,1],\ldots\}$
$q = \{q_1, q_2, q_1, q_2, \ldots\}$
$x = \{t\text{-}1, 0, 0, 0, \ldots\}$

same problem of the sign function, can be solved by using a different notion of solution to ODE

ZENO HYBRID AUTOMATA

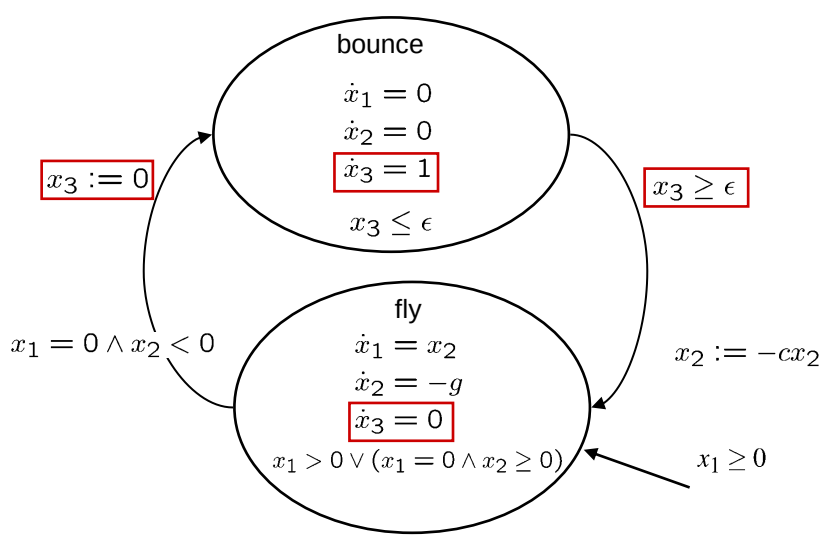
A hybrid automaton is Zeno if it accepts some Zeno execution

- difficult to simulate and analyse
- regularization methods to eliminate Zeno behavior

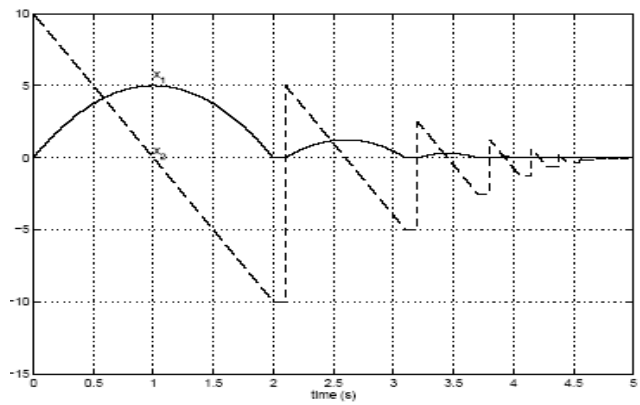
TEMPORAL REGULARIZATION: BOUNCING BALL

- each bounce takes ε >0 time units
- an infinite number of discrete transitions cannot occur in finite time

TEMPORAL REGULARIZATION: BOUNCING BALL

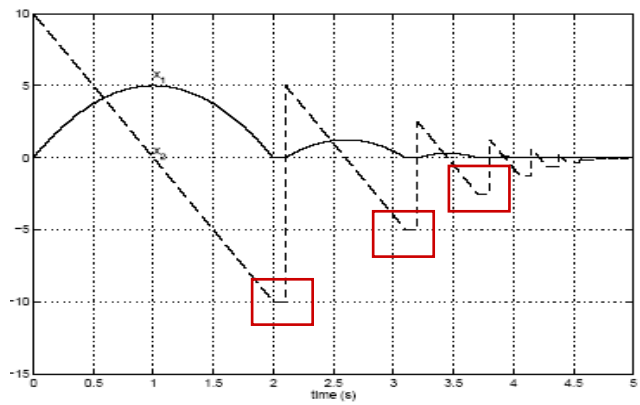


TEMPORAL REGULARIZATION: BOUNCING BALL



$\varepsilon = 0.1$

TEMPORAL REGULARIZATION: BOUNCING BALL



$\varepsilon = 0.1$

TEMPORAL REGULARIZATION: BOUNCING BALL

Executions get extended beyond the Zeno time τ_∞

In the limit, as $\varepsilon \rightarrow 0$, the execution of the regularized hybrid automaton converges to

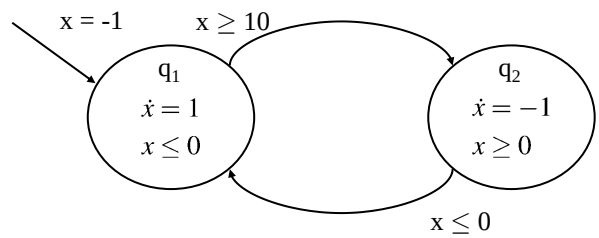
- the Zeno execution for $t < \tau_\infty$
- $x_1(t) = x_2(t) = 0$ for $t \geq \tau_\infty$

HYBRID AUTOMATA: BLOCKING vs. NON-BLOCKING

A hybrid automaton H = (Q,X,f,*Init*,Dom,E,G,R) is non-blocking if for all initial states $(q,x) \in$ *Init* there is an infinite execution starting at (q,x)

HYBRID AUTOMATA: BLOCKING vs. NON-BLOCKING

A hybrid automaton H = (Q,X,f,*Init*,Dom,E,G,R) is non-blocking if for all initial states (q,x)$\in$ *Init* there is an infinite execution starting at (q,x)



The execution starting from (q_1,-1) gets stuck at x =0 after 1 time unit

→ blocking hybrid automaton

HYBRID AUTOMATA: BLOCKING vs. NON-BLOCKING

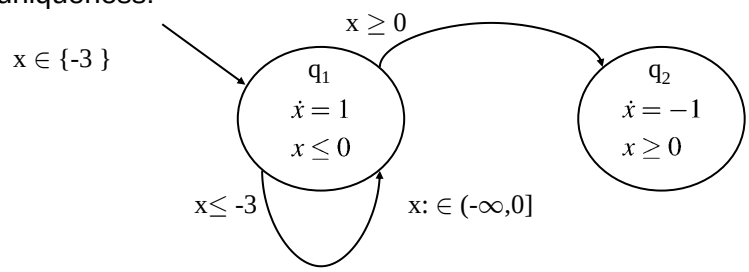
A hybrid automaton H = (Q,X,f,*Init*,Dom,E,G,R) is non-blocking if for all initial states (q,x)$\in$ *Init* there is an infinite execution starting at (q,x)

Note: this concerns with global existence of a solution, not uniqueness!

HYBRID AUTOMATA: BLOCKING vs. NON-BLOCKING

A hybrid automaton H = (Q,X,f,*Init*,Dom,E,G,R) is non-blocking if for all initial states (q,x)$\in$ *Init* there is an infinite execution starting at (q,x)

Note: this concerns with global existence of a solution, not uniqueness!



Multiple infinite executions starting from (q_1,-3) → non blocking

NON-BLOCKING HYBRID AUTOMATA

Given a hybrid automaton H = (Q,X,f,*Init*,Dom,E,G,R), transition states are those states from which continuous evolution is impossible:

Trans := {(q',x')$\in$ Q$\times$ X: $\forall$ δ>0, $\exists$ t$\in$ [0,δ) such that x(t)$\notin$Dom(q')}

where x(t) is the solution of dx/dt =f(q',x) with x(0)=x'

A hybrid automaton is non-blocking if:

1. f(q,·) is Lipschitz for each q$\in$ Q
2. for each (q,x) $\in$ Trans, there exists q' such that (q,q')$\in$ E and x $\in$ G(q,q')

(a discrete transition should be possible from the Trans states)

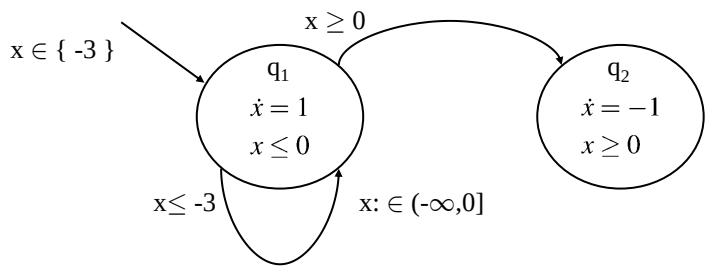
Remark: not a necessary condition, since transition states are not necessarily reached from the initial states

NON-BLOCKING HYBRID AUTOMATA

$\{(q_2,x): x \leq 0\} \subseteq$ Trans

trivial for $x < 0$ (already outside $Dom(q_2)$),
for $x = 0$ it would exit the domain

No discrete transition possible from q_2



Still… non blocking!

HYBRID AUTOMATA: DETERMINISTIC vs. NONDETERMINISTIC

An execution is maximal if "it cannot be extended any further"

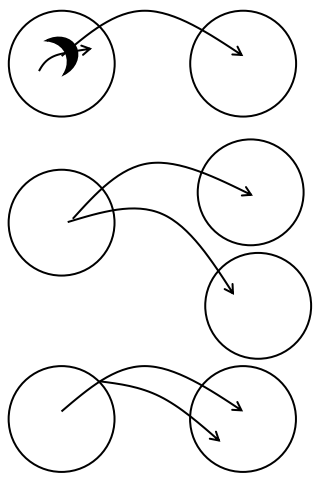
A hybrid automaton is deterministic if for each initial state $(q,x) \in$ *Init* there exists at most one maximal execution starting at (q,x).

Remarks:

1. infinite executions are maximal.
2. to make the notion of maximal execution precise, we should introduce the notion of prefix and a partial order on the hybrid time sets and on the set of executions of a hybrid automaton (see the Notes by John Lygeros)

HYBRID AUTOMATA: DETERMINISTIC vs. NONDETERMINISTIC

What causes non-determinism?



choice between continuous evolution and discrete jump

jump to multiple modes

multiple reset positions

DETERMINISTIC HYBRID AUTOMATA

A hybrid automaton is deterministic if

1. If $x \in G(q,q')$ for some $(q,q') \in E$, then $(q,x) \in$ Trans
   (jump only when continuous evolution not possible)
2. If (q,q'), $(q,q'') \in E$, then $G(q,q') \cap G(q,q'') = \emptyset$
   (no multiple jumps possible)
3. If $(q,q') \in E$ and $x \in G(q,q')$, then $R((q,q'),x)$ is a singleton
   (only one reset position when jumping)

EXISTENCE AND UNIQUENESS OF EXECUTIONS

A hybrid automaton has a unique infinite execution for each initial state if it is non-blocking and deterministic.

Sufficient condition given by the putting together the sufficient conditions for the hybrid automaton to be non-blocking and deterministic

HYBRID AUTOMATA: FORMAL DEFINITION

A hybrid automaton H is a collection

$$H = (Q,X,f,Init,Dom,E,G,R)$$

- $Q = \{q_1,q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $f: Q \times X \rightarrow \Re^n$ is a set of vector fields on X
- $Init \subseteq Q \times X$ is a set of initial states
- $Dom: Q \rightarrow 2^X$ assigns to each $q \in Q$ a domain Dom(q) of X
- $E \subseteq Q \times Q$ is a set of transitions (edges)
- $G: E \rightarrow 2^X$ is a set of guards (guard condition)
- $R: E \times X \rightarrow 2^X$ is a set of reset maps

No input and output variables….

OPEN HYBRID AUTOMATA

An open hybrid automaton H is a collection

$$H = (Q,X,V,W,f,h,\mathit{Init},\mathrm{Dom},E,G,R)$$

- $Q = \{q_1, q_2, \ldots\}$ is a set of discrete states (modes)
- $X = \Re^n$ is the continuous state space
- $V = \Sigma \times U$ is the input space (discrete & continuous)
- $W = \Psi \times Y$ is the output space (discrete & continuous)
- $f: Q \times X \times V \rightarrow \Re^n$ is a set of vector fields on X
- $h: Q \times X \rightarrow W$ is an output map
- $\mathit{Init} \subseteq Q \times X$ is a set of initial states
- $\mathrm{Dom}: Q \rightarrow 2^{X \times V}$ assigns to each $q \in Q$ a domain Dom(q) of $X \times V$
- $E \subseteq Q \times Q$ is a set of transitions (edges)
- $G: E \rightarrow 2^{X \times V}$ is a set of guards (guard condition)
- $R: E \times X \times V \rightarrow 2^X$ is a set of reset maps