Models of Hybrid Systems

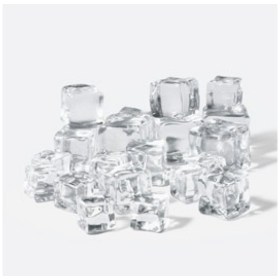
Michael S. Branicky
Department of Electrical Engineering and Computer Science
Case Western Reserve University

2nd HYCON PhD School on Hybrid Systems
University of Siena, ITALY
July 16, 2007

Continuous + Discrete = Hybrid (1)

Mixture of ... continuous & discrete inputs, outputs, states, dynamics

+

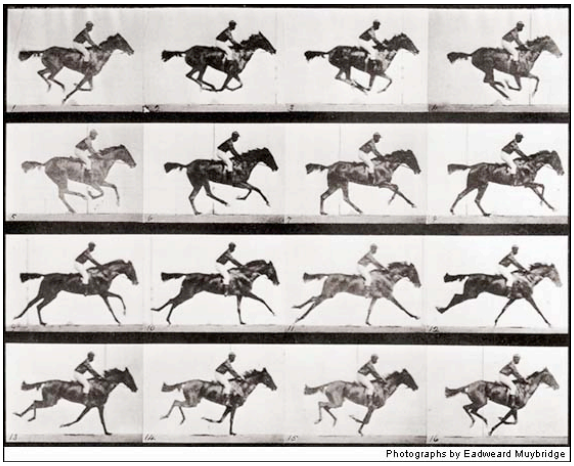


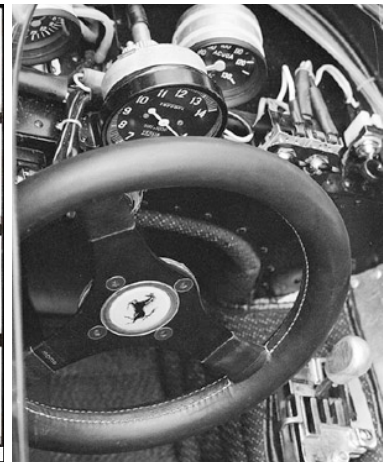
=

$\mathbf{R}^n$ $\quad Q \simeq \{1, 2, \ldots, N\}$ $\quad \mathbf{R}^n \times Q$

Continuous + Discrete = Hybrid (2)

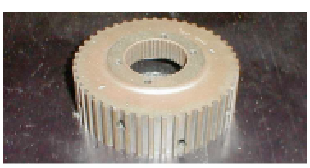
Mixture of . . . differential equations and discrete events / switching

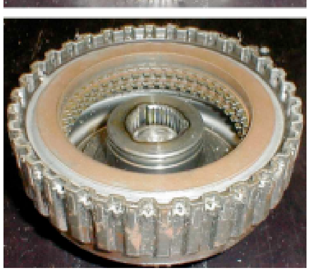


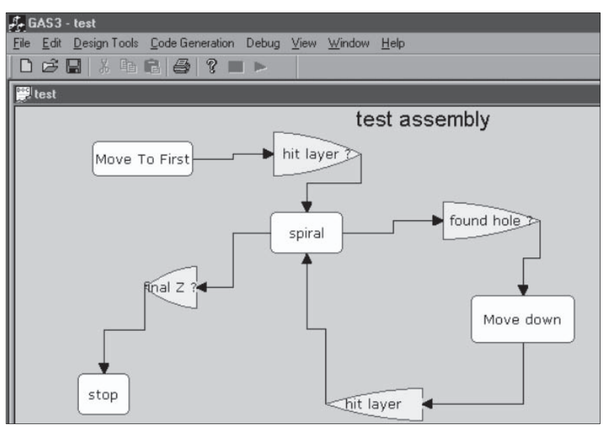


Continuous + Discrete = Hybrid (3)

Mixture of . . . continuous physical process with finite-state logic



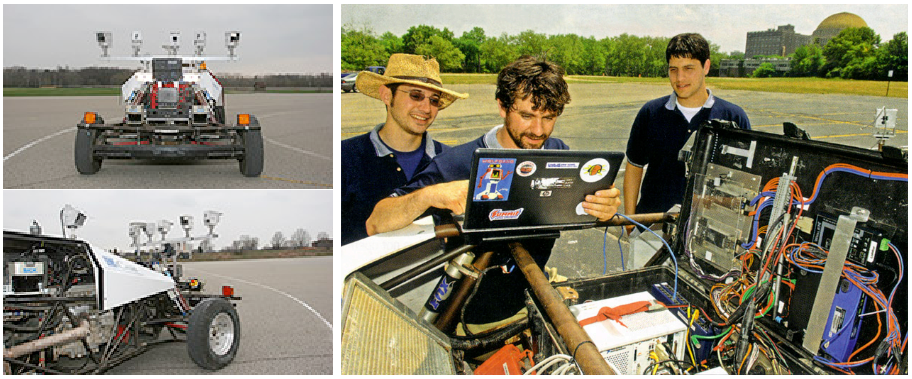




Force-guided robotic assembly [Branicky-Chhatpar, *HSCC*, 2002]

Continuous + Discrete = Hybrid (4)

Mixture of . . . control theory and computer science



Autonomous vehicle DEXTER [`urbanchallenge.case.edu`]

Outline

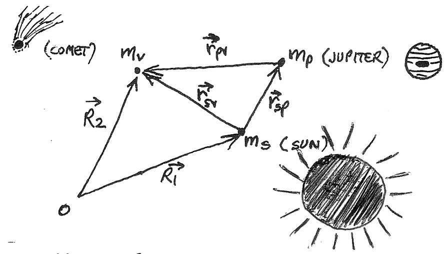
The First Hybrid Dynamicist

More Hybrid Systems Examples

Mathematical Models of HS

Laplace's Problem

Predict the motion of a comet about to pass near Jupiter (1845)



$$\ddot{\vec{R}}_1 = \frac{G m_v}{r_{sv}^3}\vec{r}_{sv} + \frac{G m_p}{r_{sp}^3}\vec{r}_{sp}$$

$$\ddot{\vec{R}}_2 = \frac{G m_s}{r_{sv}^3}\vec{r}_{vs} + \frac{G m_p}{r_{vp}^3}\vec{r}_{vp}$$

$\vec{r}_{sv} = \vec{R}_2 - \vec{R}_1$

$\vec{r}_{ij} = -\vec{r}_{ji}$

$$\ddot{\vec{r}}_{sv} + \frac{G(m_s+m_v)}{r_{sv}^3}\vec{r}_{sv} = \underbrace{-G m_p\left[\frac{\vec{r}_{pv}}{r_{pv}^3} + \frac{\vec{r}_{sp}}{r_{sp}^3}\right]}_{\text{THIRD-BODY EFFECTS}}$$

7

Laplace's Solution (1)

Two descriptions of motion plus
a logical choice of how to switch between them

$$\ddot{\vec{r}}_{sv} + \frac{G(m_s+m_v)}{r_{sv}^3}\vec{r}_{sv} = -G m_p\left[\frac{\vec{r}_{pv}}{r_{pv}^3} + \frac{\vec{r}_{sp}}{r_{sp}^3}\right]$$

$$\Longrightarrow \ddot{r}_{sv} - A_s = P_p$$

$$\ddot{\vec{r}}_{pv} + \frac{G(m_p+m_v)}{r_{pv}^3}\vec{r}_{pv} = -G m_s\left[\frac{\vec{r}_{sv}}{r_{sv}^3} - \frac{\vec{r}_{sp}}{r_{sp}^3}\right]$$

$$\Longrightarrow \ddot{r}_{pv} - A_p = P_s$$

WHEN v CLOSE TO p : $P_s \ll A_p$ (VECTORS CANCEL)
WHEN v FAR FROM p : $P_p \ll A_s$ ($m_s \gg m_p$)

SWITCH FROM SUN-CENTERED COORDS. WHEN $\frac{P_p}{A_s} > \frac{P_s}{A_p}$ $r_{SOI} \approx \left(\frac{m_p}{m_s}\right)^{2/5} r_{sp}$

INSIDE "SPHERE OF INFLUENCE" OF PLANET

8

Laplace's Solution (2)

Double-think: SOI is both infinitely large and infinitesimally small

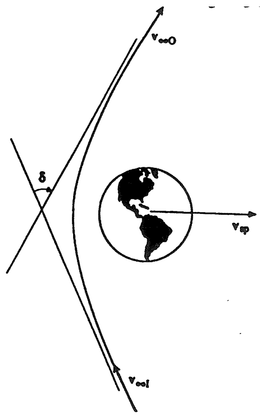


Fig. 7.7 Flyby Trajectory

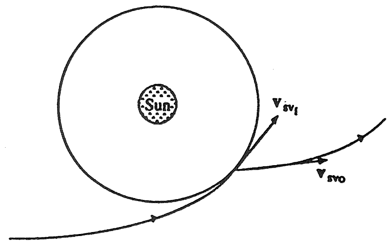


Fig. 7.9 Discontinuity in Velocity due to Flyby

e: 145, J: 677 (size in radii)   e: 0.006, J: 0.06 (fraction of area)

Laplace's Solution (3)

A different, logical(?) choice of when to switch

VEHICLE WITHIN
SOI OF EARTH IF

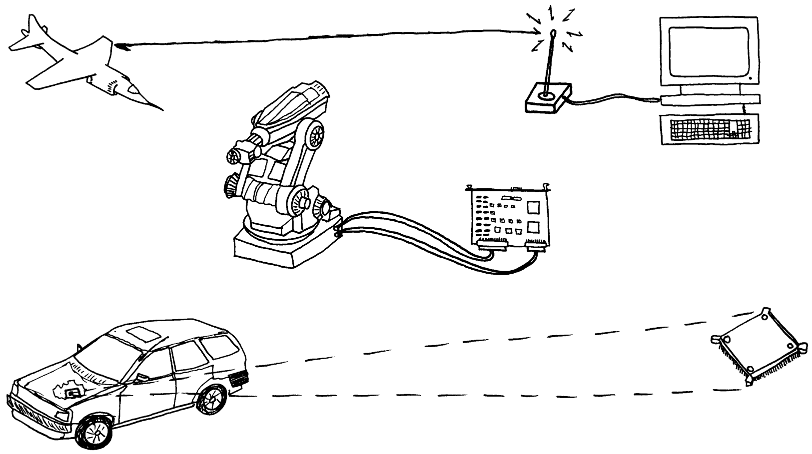
$$\frac{G m_e m_v}{r_{ev}^2} > \frac{G m_s m_v}{r_{sv}^2}$$

$$\Rightarrow \quad r_{SOI_e} \approx 42 \text{ earth radii}$$

DISTANCE $\quad r_{em} \approx 60$ earth radii

LAPLACE : $\quad r_{SOI_e} \approx 145$ earth radii

Hybrid Systems All Around Us



They drive on our streets, work in our factories, fly in our skies, ...

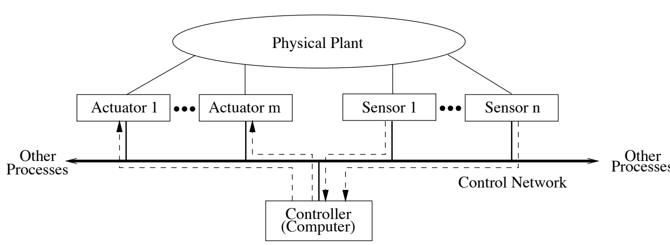
Networked Control Systems (1)

Sensors, actuators, and controllers connected over a network . . .
with feedback loops controlling physical systems closed among them

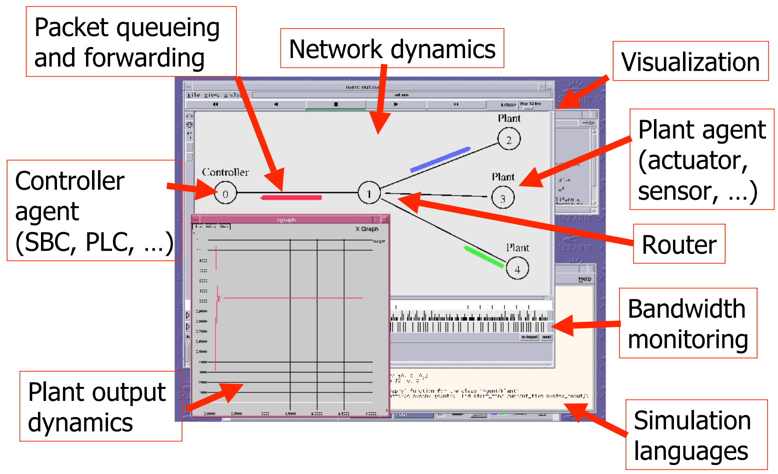
- continuous plants
- asynchronous or *event-driven* data transmission

  sampling, varying transmission delay, packet loss
- discrete implementation of network/protocols

  data packets, queuing, routing, scheduling, etc.



Networked Control Systems (2)



Co-simulation and co-design [Branicky-Liberatore-Phillips, *ACC*, 2003]

Other Examples

- systems with relays, switches, and hysteresis
- computer disk drives
- constrained robotic systems (locomotion, assembly, etc.)
- vehicle powertrains, transmissions, stepper motors
- mode-switched flight control, vehicle management systems

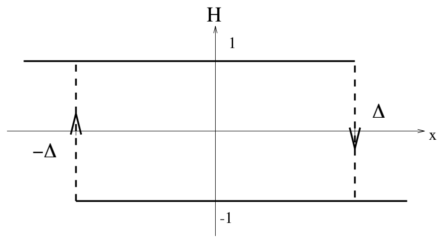
- automated highway systems (AHS)
- multi-vehicle formations and coordination
- power electronics
- analog/digital circuit co-design and verification
- biological applications

Systems with Switches and Relays

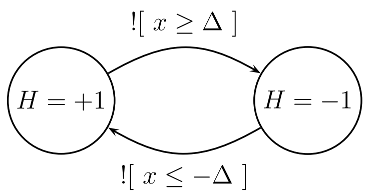
HVAC control with a thermostat:

$$\dot{x} = f(x, H(x - x_0), u)$$

- x, room temperature
- x_0, desired temperature
- f, dynamics of temperature
- u, control signal (e.g., the fuel burn rate)

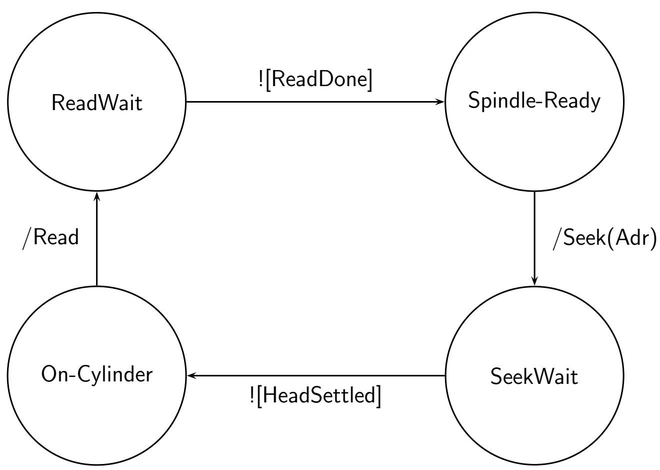


Hysteresis Function, H



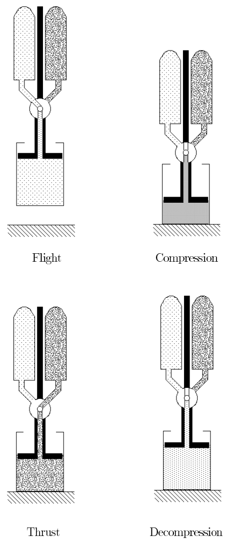
Associated Finite Automaton

Hard Disk Drive

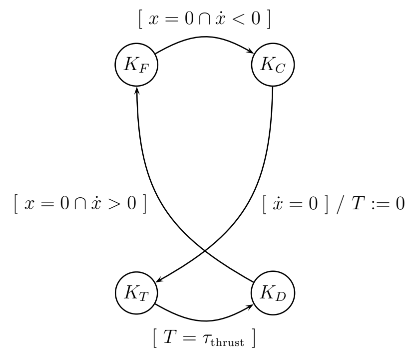


HS for main hard disk drive functionality [Gollu-Varaiya, *CDC*, 1989]

Raibert's Hopping Robot



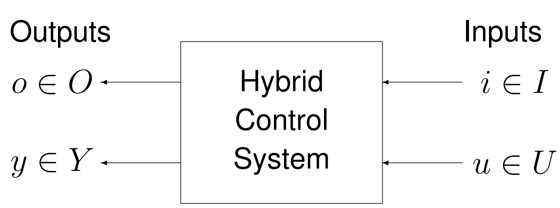
Dynamic Phases
[Back *et al.*, *HS I*, 1993]



Finite State Controller

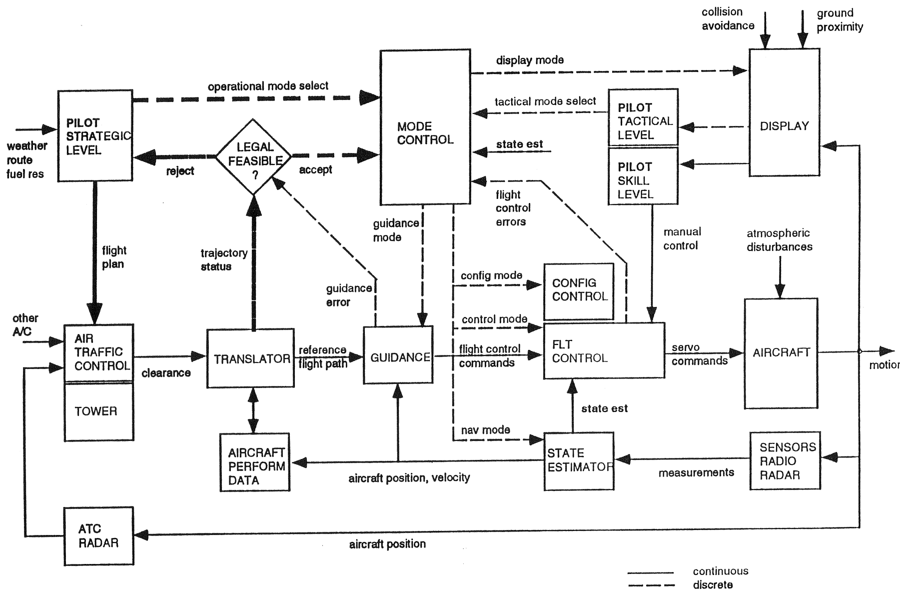
Vehicle Powertrains / Cruise Control

Continuous	**Discrete**
Throttle	Gear Position
Engine RPM	Cylinder Phases
Fuel/Air Mixture	Cylinder Firings
Belts, Cams	Microprocessors
Elevation	Road Condition



I, O are discrete (i.e., countable) sets of symbols
U, Y are continuums

Flight Vehicle Mgmt. Systems



[George Meyer, Plenary Lecture, *CDC*, 1994]

View From Here

The remainder of this talk focuses on **mathematical models**

- From Continuous Toward Hybrid
  $\Longrightarrow$ Hybrid Dynamical Systems

- From Discrete Toward Hybrid
  $\Longrightarrow$ Hybrid Automata

From Continuous Toward Hybrid

Differential Equations[1]

+

Discrete Phenomena

$$\Longrightarrow$$

Hybrid Dynamical Systems

[1] It is easy to substitute "Difference Equations"

Base Continuous Model: ODEs

Ordinary differential equation (*ODE*):

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

$x(t) \in X \subset \mathbf{R}^n$ is a vector of *continuous states*
$f : X \longrightarrow \mathbf{R}^n$ is a *vector field* on $\mathbf{R}^n$

Autonomous/*time-invariant*: vector field doesn't depend explicitly on t

Non-autonomous or *time-varying*:

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

ODE with Inputs and Outputs

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t), u(t)) \end{aligned}$$

$x(t) \in X \subset \mathbf{R}^n, \quad u(t) \in U \subset \mathbf{R}^m, \quad y \in Y \subset \mathbf{R}^p$
$f : \mathbf{R}^n \times \mathbf{R}^m \longrightarrow \mathbf{R}^n, \quad h : \mathbf{R}^n \times \mathbf{R}^m \longrightarrow \mathbf{R}^p$

The functions $u(\cdot)$ and $y(\cdot)$ are the *inputs* and *outputs*, respectively

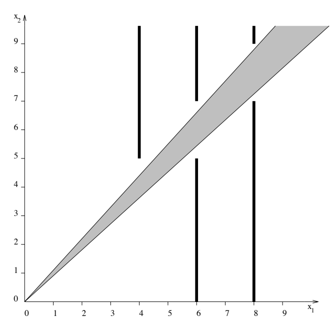
Whenever inputs are present, we say $f(\cdot)$ is a *controlled vector field*

Differential Inclusions

$$\dot{x}(t) \in F(x(t))$$

- Derivative belongs to a set of vectors in $\mathbf{R}^n$
- Models nondeterminism (controls, disturbances, uncertainty, ...)

Example 1 (Innacurate Clock)
A clock with time-varying rate between 0.9 and 1.1 can be modeled by $\dot{x} \in [0.9, 1.1]$, *which is a* rectangular *inclusion*



[van der Schaft-Schumacher, 1995]

Adding Discrete Phenomena

Continuous state dynamics given by

$$\dot{x}(t) = \xi(t), \qquad t \geq 0$$

Vector field $\xi(t)$ depends on x (and u) **plus discrete phenomena**:

- **autonomous switching**: vector field changes discontinuously
- **autonomous jumps**: continuous state changes discontinuously
- **controlled switching**: control switches vector field discontinuously
- **controlled jumps**: control changes cont. state discontinuously

Autonomous Switching

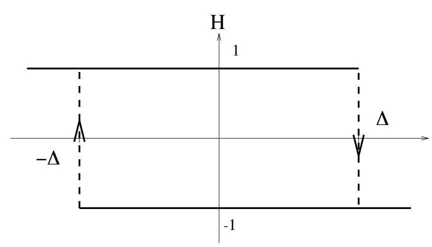
Vector field $\xi(\cdot)$ changes discontinuously when
the continuous state $x(\cdot)$ hits certain "boundaries"

Example 2 (HVAC) *Dynamics are given by*

$$\begin{aligned} \dot{x}(t) &= f_1(x(t)), & \text{furnace is On} \\ \dot{x}(t) &= f_0(x(t)), & \text{furnace is Off} \end{aligned}$$

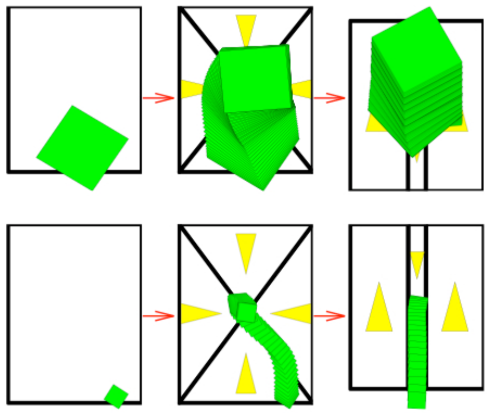
$x(t)$ *is temperature*

$$\dot{x}(t) = f_1(x(t))$$



$$\dot{x}(t) = f_0(x(t))$$

Piece-Wise Constant Vector Fields



Programmable vector fields for sorting parts (large, up; small, down)
Vector fields are merely sequenced in time (sensorless or open loop)

Figure from [Böhringer *et al.*, Computational Methods for Design and Control of MEMS Micromanipulator Arrays, *IEEE Computer Science and Engineering*, pp. 17–29, January–March 1997]

Switched Systems

A general *switched system*:[2]

$$\dot{x}(t) = f_{q(t)}(x(t))$$

where $q(t) \in Q \simeq \{1, \ldots, N\}$

E.g., $Q = \{0, 1\}$ for furnace Off, On

Important subclass: *switched linear systems*

$$\dot{x}(t) = A_q x(t), \qquad q \in \{1, \ldots, N\}$$

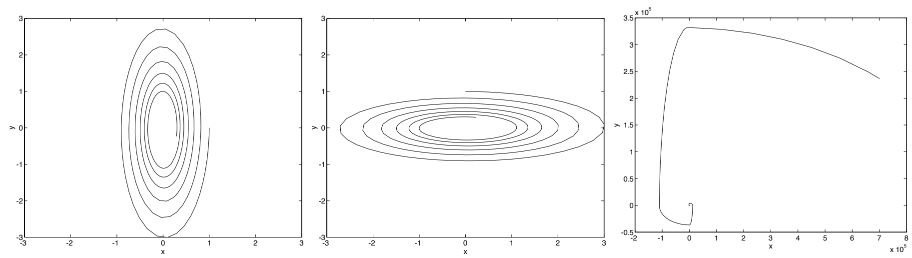
where each $A_q \in \mathbf{R}^{n \times n}$

[2]Note: Switching boundaries/manifolds have been suppressed; really, $q^+(t) = \nu(x(t), q(t))$ and hybrid state is (x, q)

Switched Linear Systems (1)

Example 3 (Unstable from Stable [Branicky, *IEEE T-AC*, 1998])

$$\dot{x}(t) = A_q x(t), \qquad A_0 = \begin{bmatrix} -0.1 & 1 \\ -10 & -0.1 \end{bmatrix}, \qquad A_1 = \begin{bmatrix} -0.1 & 10 \\ -1 & -0.1 \end{bmatrix}$$

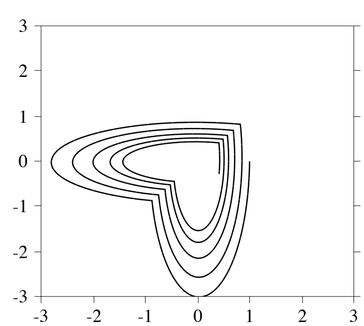


Trajectories: (left) A_0*, (center)* A_1*, (right)* A_i*,* $i =$ quadrant $\bmod 2$

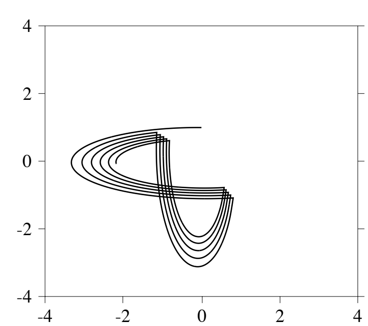
Switched Linear Systems (2)

Example 4 (Stable from Stable)

Two stable linear systems
Both "clockwise"
Switching on a line



Two stable linear systems
One anti-clockwise
Switching with a hybrid rule



Argues for "Multiple Lyapunov Functions" to prove stability [Branicky, *IEEE T-AC*, 1998]
Simulated using Omola/Omsim [Andersson, PhD, 1994; Branicky-Mattsson, *HS IV*, 1997]

Autonomous Jumps / Impulses

Continuous state $x(\cdot)$ jumps discontinuously on
hitting prescribed regions of the state space

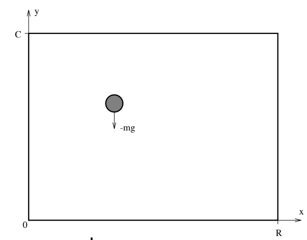
E.g., collisions (running animals, hopping robots, etc.)

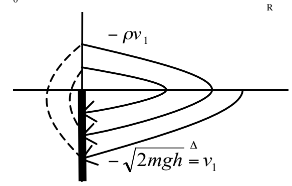
Example 5 (Bouncing Ball)

$$\begin{aligned} \dot{y}(t) &= v(t) \\ \dot{v}(t) &= -mg \\ v^+(t) &= -\rho v(t), \qquad x(t) \in M \end{aligned}$$

$M = \{(0, v) \mid v < 0\}$
$0 \leq \rho \leq 1$, *coefficient of restitution*

"If $y = 0$ and $v < 0$, $v := -\rho v$"





Networked Control System

Example 6 (NCS) *A linear, full-state feedback control system*

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ u(t) &= -Kx(t) \end{aligned}$$

Place a network between state measurement (at sensor node) and control computation/actuation (at another node)

x *is measured at time* t_i, *received after delay* d_i

$$\begin{aligned} \dot{x}(t) &= Ax(t) - BK\hat{x}(t) \\ \hat{x}^+(t) &= x(t_i), \qquad \text{when } t = t_i + d_i \end{aligned}$$

Note: augmented state measurement $\hat{x}$ is piecewise constant

Autonomous Jumps / Impulses (2)

General system subject to autonomous impulses:

$$\begin{aligned} \dot{x}(t) &= f(x(t)), & x(t) \notin A \\ x^+(t) &= G(x(t)), & x(t) \in A \end{aligned}$$

Autonomous jump set, A
Reset map, G

Linear system with equally spaced impulses [Branicky, *CDC*, 1997]

$$\begin{aligned} \dot{x}(t) &= P_1 x(t), & t \notin I \\ x^+(t) &= P_2 x(t), & t \in I = \{0, h, 2h, \ldots\} \end{aligned}$$

Stable if eigenvalues of $P_2 e^{P_1 h}$ have magnitude < 1

Controlled Switching

Vector field $\xi(\cdot)$ changes abruptly in response to a control command, usually with an associated cost

One is allowed to pick among a discrete number of vector fields:

$$\dot{x} = f_{q(t)}(x)$$

$q(t) \in Q \simeq \{1, 2, \ldots, N\}$ (or $Q \simeq \mathbf{Z}$)
$q(t)$ chosen by the controller

Note: If $q(t)$ were an explicit function of state, result would be a closed-loop system with autonomous switches

Controlled Switching Examples (1)

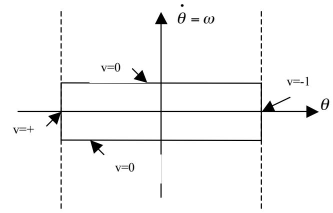
Example 7 (Satellite Control)

$$\ddot{\theta} = \tau_{\mathit{eff}}\, v$$

θ, $\dot{\theta}$, angular position and velocity

$v \in \{-1, 0, 1\}$, reaction jets

are full reverse, off, or full on



Example 8 (Manual Transmission [Brockett, 1993])

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= [-a(x_2/v) + u]/(1+v) \end{aligned}$$

x_1, ground speed

x_2, engine RPM

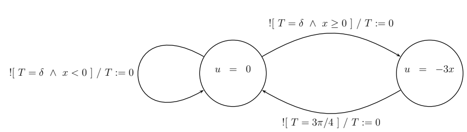
$u \in [0, 1]$, throttle position

$v \in \{1, 2, 3, 4\}$, gear shift position

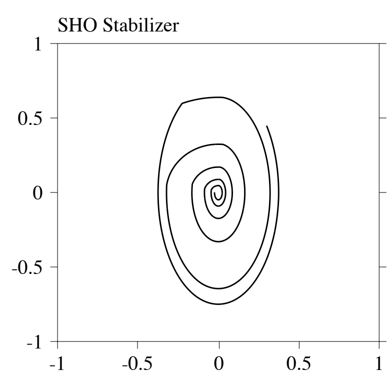
a is positive for positive argument

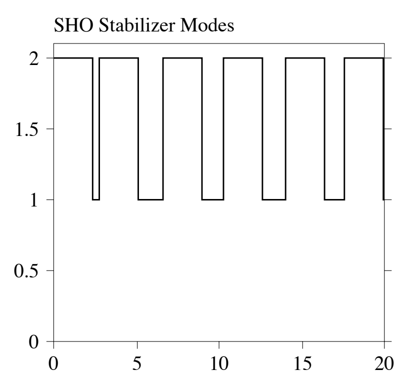
Switching Control Laws (1)

Example 9 (Pait's S.H.O. Stabilizer [Artstein, *HS III*, 1996])



$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= -x + u \\ x &\equiv \text{state measurement} \\ \\ \dot{T} &= 1 \text{ (always in model)} \end{aligned}$$

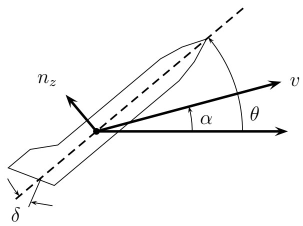


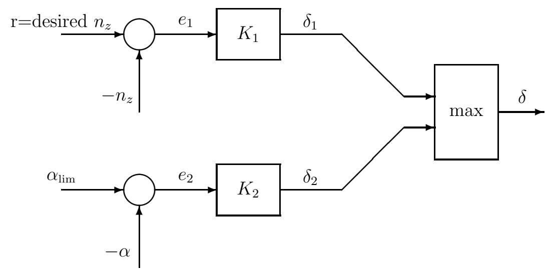


Switching Control Laws (2)

Example 10 (Max Controller [Branicky, *ACC*, 1994])
Control objective:

Good tracking of the pilot's input, n_z,
without violating angle-of-attack constraint

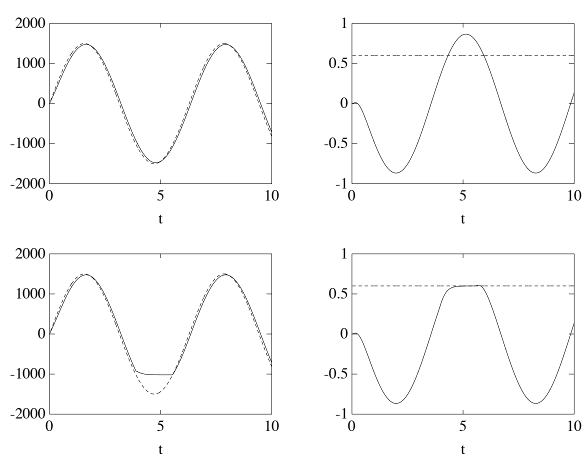




Longitudinal Aircraft View *Max Controller*

Switching Control Laws (3)

Outputs of tracking (top) and max controller (bottom)



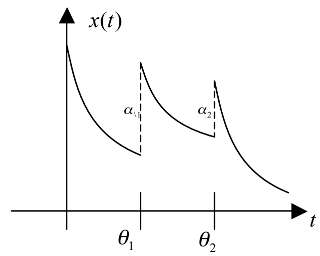
Left: normal acceleration n_z (solid), desired value r (dashed)
Right: angle of attack α (solid), α's limit (dashed)

Controlled Jumps / Impulses

Continuous state $x(\cdot)$ changes discontinuously in response to a control command, usually with an associated cost

Example 11 (Inventory Management)

$$\dot{x}(t) = -\mu(t) + \sum_i \delta(t - \theta_i)\alpha_i$$



x, stock
μ, *degradation/utilization*
$\theta_1 < \theta_2 < \ldots$, *"discrete" restocking times*
$\alpha_1, \alpha_2, \ldots$, *order amounts*

Note: *If stocking times/amounts explicit function of x, then controlled jumps become autonomous jumps*

Example 12 (Planetary Flybys) *Exploration spacecraft typically use close encounters with moons/planets to gain energy, change course*

At the level of the entire solar system, these maneuvers are planned by considering the flight path to be a sequence of parabolic curves, with resets of heading/velocity occurring at the "point" of encounter

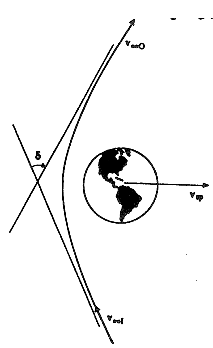


Fig. 7.7 Flyby Trajectory

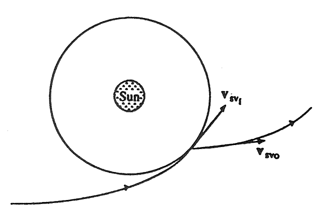


Fig. 7.9 Discontinuity in Velocity due to Flyby

Significant Hybrid Phenomena

Continuous dynamics and controls +

Type: Example	Discontinuity	
Source	Vector Field (Switching)	Continuous State (Jump/Impulse)
System (Autonomous)	Autonomous Switching: Hysteresis	Autonomous Jumps/Impulses: Collisions
Controller (Controlled)	Controlled Switching: Gearbox	Controlled Jumps/Impulses: Resets

+ interactions with finite automata
+ other models (**Tavernini**, Brockett, Nerode-Kohn, BGM, ASL, ...)

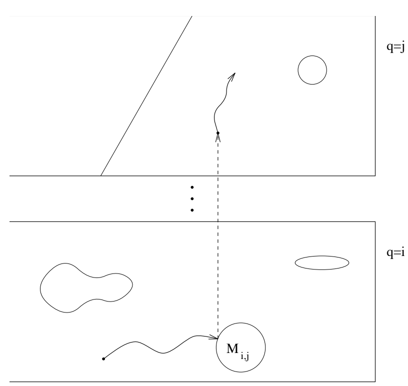
Tavernini's Model

Differential automaton [Tavernini, 1987]:
A triple $(S,\ f,\ \nu)$ where

- S = $\mathbf{R}^n \times Q$, *(hybrid) state space*
  $Q \simeq \{1, \ldots, N\}$, *discrete state space*
  $\mathbf{R}^n$, *continuous state space*

- $f(\cdot, q) = \mathbf{R}^n \to \mathbf{R}^n$, for each $q \in Q$, *continuous dynamics*

- $\nu : S \to Q$, *discrete transition function*



In our notation:

$$\begin{aligned} \dot{x} &= f(x, q) \\ q^+ &= \nu(x, q) \end{aligned}$$

Tavernini's Results

Assumptions:

- switching manifolds are given by the zeros of a smooth function
- separation of switching sets, separation from concatenated jumps

Results:

- Unique solution with finitely many switching points

$$s_0(t_0) = (x_0, q_0), \qquad s_1(t_1) = (x_1, q_1), \qquad s_2(t_2) = (x_2, q_2), \qquad \ldots$$

- Continuity in initial conditions[3]

$$|s_0 - s_0'| < \delta \Longrightarrow \begin{array}{rcl} |x(t) - x'(t)| & < & \epsilon_1,\ t < T \\ q_0 q_1 \cdots q_M & = & q_0' q_1' \cdots q_M' \\ |t_i - t_i'| & < & \epsilon_2,\ i \le M \end{array}$$

- Numerical integration approaches true solution[4]

$$|s'(t;h) - s(t)| \to 0 \quad \text{as} \quad h \to 0$$

[3]On an open, dense set S^0

[4]With initial error (from $s_0' \neq s_0$); uniformly, in S^0

Hybrid Dynamical Systems (HDS)

An indexed collection of DSs plus a map for "jumping" among them

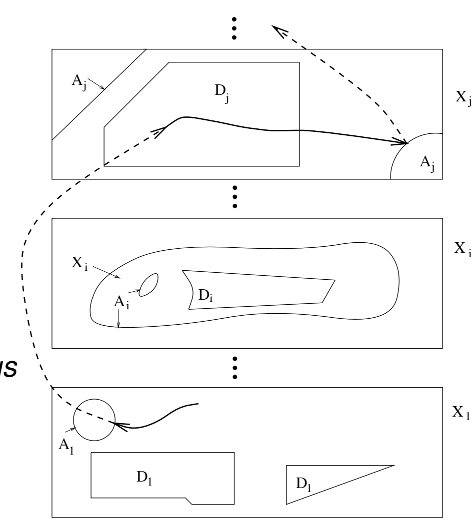
$$H = (Q, \mathbf{\Sigma}, \mathbf{A}, \mathbf{G})$$

- Q, countable *discrete states*
- $\mathbf{\Sigma} = \{\Sigma_q\}_{q \in Q}$, set of DSs

  $f_q : X_q \to \mathbf{R}^{d_q}$, $X_q \subset \mathbf{R}^{d_q}$,

  continuous state spaces
- A_q, *autonomous jump sets*
- $G_q : A_q \to S$, *autonomous jump transition maps*

Hybrid state space:
$S = \bigcup_{q \in Q} X_q \times \{q\}$



Hybrid Dynamical Systems: Notes

- *ODEs and Automata*

  ODEs: $|Q| = 1$, $A = \emptyset$
  (Later) Finite Automata: $|Q| = N$, each $f_q \equiv 0$

- *Outputs*: add continuous/discrete output maps for each q

- *Changing State Space*

  inelastic collisions, component failures, aircraft modes, . . .

- *State Space Overlaps*, e.g., hysteresis

- *Transition Delays*

  Add *autonomous jump delay map*, $\Delta_a : A \times V \longrightarrow \mathbf{R}_+$
  Associates (possibly zero) delay to each jump
  Aggregate transients, activation delay, etc.

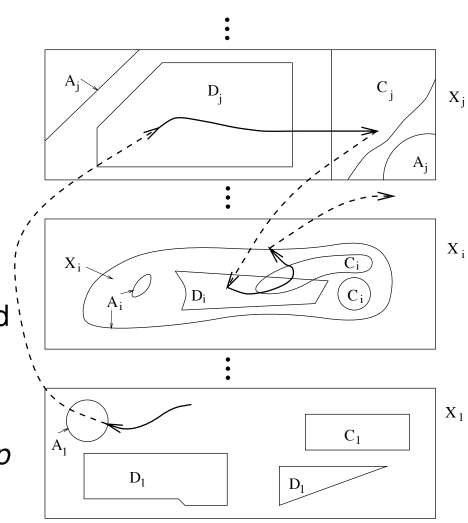
Adding Control: Controlled HDS

$$H_c = (Q, \mathbf{\Sigma}, \mathbf{A}, \mathbf{G}, \mathbf{C}, \mathbf{F})$$

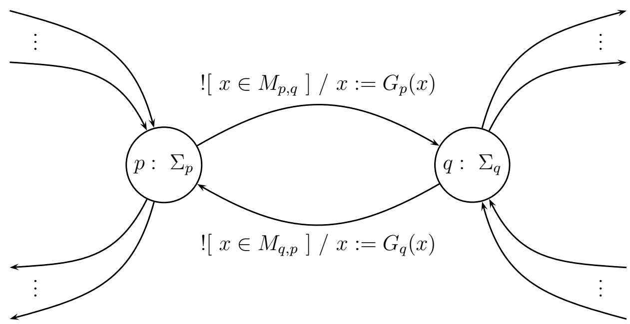
- $\mathbf{\Sigma_q}$, *controlled* ODEs

  $f_q \;:\; X_q \times U_q \;\to\; \mathbf{R}^{d_q}$
  $U_q \;\subset\; \mathbf{R}^{m_q}$, *continuous control spaces*

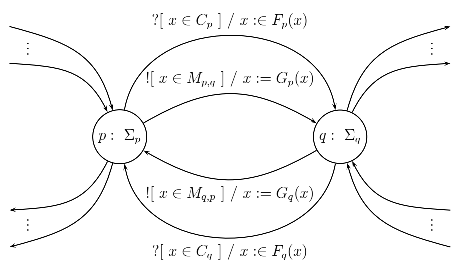
- $G_q : A_q \times V_q \to S$, modulated by *discrete decisions* V_q
- C_q, *controlled jump sets*
- $F_q : C_q \to 2^S$, *controlled jump destination maps* (set-valued)



(C)HDS: Automaton View

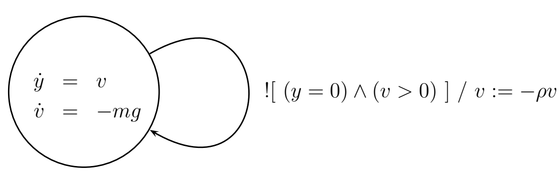


![condition]: *must* be taken
?[condition]: *may* be taken
":∈", reassignment to value in set

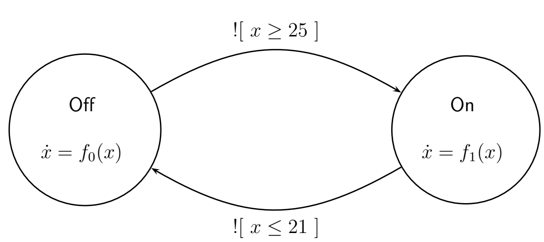


Hybrid Automata: Examples (1)

Example 13 (Bouncing Ball Revisited)

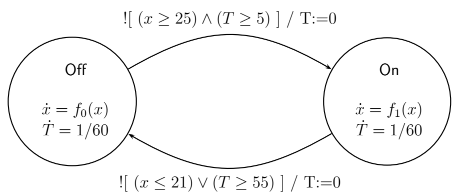


Example 14 (HVAC Revisited) *Goal of A.C.: temp. at* $23 \pm 2\,^{\circ}$*C*

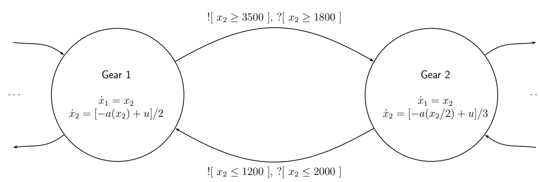


Hybrid Automata: Examples (2)

Example 15 (HVAC++) *Add that A.C.*
(i) is never ON *more than 55 minutes straight*
(ii) must remain Off *for at least 5 minutes*



Example 16 (Audi A4 Tiptronic Transmission)



From Discrete to Hybrid

Automata[5]

+

Continuous Phenomena

$\Longrightarrow$

Hybrid Automata

[5] It is easy to substitute "Automata" with "Petri Nets"

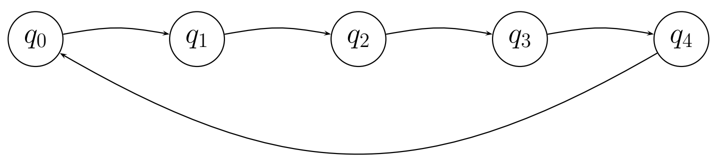
Base Discrete Model: FA / FSM

Inputless finite automaton (FA) or finite state machine (FSM):

$$q(k+1) = \nu(q(k))$$

$q(k) \in Q$, a finite set
i.e., dynamical system with discrete state space

Example 17 (Finite Counter)
State space $Q = \{q_0, q_1, \ldots, q_{N-1}\}$ *and* $\nu(q_i) = q_{i+1 \bmod N}$

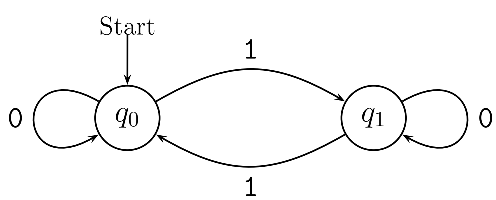


Starting from initial state, q_0*,* **trajectory** *or* **run** *is:*

$$q_0,\ q_1,\ q_2,\ q_3,\ q_4,\ q_0,\ q_1,\ q_2,\ q_3,\ q_4,\ q_0, \cdots$$

Deterministic FA Example

Example 18 (Parity of Binary String Input)
DFA keeps track of input's parity by counting `1`*s, modulo 2*



$$Q = \{q_0, q_1\}$$
$$q(k+1) = \nu(q(k), i(k))$$

ν	0	1
q_0	q_0	q_1
q_1	q_1	q_0

On input `1101`*, run is* q_0, q_1, q_0, q_0, q_1

"Unrolled" View: $q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1$

Exercise 1 *Draw a DFA whose states track number of* `0`*s mod. 3*

Exercise 2* *Draw one whose states track binary number seen mod. 3*

Automaton Preliminaries

symbol: abstract entity of automata theory, e.g., letter or digit

alphabet: finite set of symbols

$E = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots, \mathtt{z}\}$ — English alphabet
$D = \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \ldots, \mathtt{9}\}$ — Decimal digits
$B = \{\mathtt{0}, \mathtt{1}\}$ — Binary alphabet
Latin 1 — IS0 8859-1 (Unicode characters)

string / ***word*** **(over alphabet** I**):** finite sequence of symbols from I

`cat` and `jazz` and `zebra`; `w` and `qqq` — strings over E
`0` and `1` and `1101` — strings over B

empty string, ε**:** string consisting of zero symbols

concatenation **operator:** strings can be juxtaposed

$\mathtt{cat} \cdot \mathtt{jazz} = \mathtt{catjazz}$
$00 \cdot 11 = 0011 \neq 1100 = 11 \cdot 00$
$\mathtt{q}^3 = \mathtt{qqq}, \qquad \mathtt{q}^0 = \varepsilon$

Deterministic Finite Automata (DFA)

A *DFA* is a four-tuple $\mathcal{A} = (Q, I, \nu, q_0)$, where

- Q is a finite set of *states*
- I is an alphabet, called the *input alphabet*
- ν is the *transition function* mapping $Q \times I$ into Q
- $q_0 \in Q$ is the *initial state*

Dynamics:

- Machine starts in state q_0
- One move: DFA in q receives symbol a and enters state $\nu(q, a)$
- On input word $w = a_1 a_2 \cdots a_n$: DFA in r_0 successively processes symbols and sequences through states $r_1, r_2, \ldots, r_n$, such that

$$r_{k+1} = \nu(r_k, a_k)$$

This sequence is a *run* of DFA over w

$$r_0 \xrightarrow{\mathtt{a}_1} r_1 \xrightarrow{\mathtt{a}_2} r_2 \xrightarrow{\mathtt{a}_3} \cdots \xrightarrow{\mathtt{a}_{n-1}} r_{n-1} \xrightarrow{\mathtt{a}_n} r_n$$

Languages

***language* (over alphabet I):** a set of strings over I

English language, L_E, is a language over E
cat $\in L_E$, qqq $\notin L_E$

Languages over B:

$$\begin{aligned} B_{\text{length 2}} &= \{00, 01, 10, 11\} \\ B_{\text{even length}} &= \{\varepsilon\} \cup B_{\text{length 2}} \cup B_{\text{length 4}} \cup \cdots \\ B_{\text{odd parity}} &= \{1, 01, 10, 001, 010, 100, 111, \ldots\} \end{aligned}$$

***Kleene closure,* I^*:** set of all strings over alphabet I

$B^* \equiv \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
$B^+ \equiv \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\} = B^* - \{\varepsilon\}$

***empty language,* $\emptyset$:** language without any strings (note $\emptyset \neq \{\varepsilon\}$)

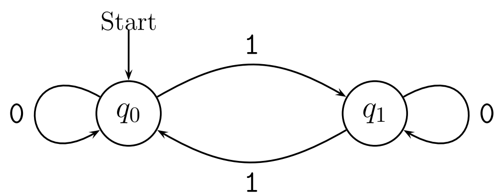
***concatenation* of languages:** $ST = \{st \mid s \in S, t \in T\}$

Exercise 3 *$S = \{0\}^+ = 0^+$, $T = \{1\}^+ = 1^+$; what is ST?*

Marked States

DFA plus a set of marked or accepting or *final states*, F

language of FA: set of strings having a run that ends in a state of F
a.k.a. set of *accepted* strings



If $F = \{q_1\}$, 111 is accepted, ε is not; accepted language is $B_{\text{odd parity}}$
If $F = \{q_0, q_1\}$: accepted language is B^*
If $F = \{q_1\}$: it would be $B^* - B_{\text{odd parity}}$
If $F = \emptyset$: it is $\emptyset$

Exercise 4 *Draw DFA accepting: (i) $B_{\text{even length}}$, (ii) $B_{\text{length 2}}$*

Nondeterministic FA (NFA)

An *NFA* $N = (Q, I, \hat{\nu}, Q_0, F)$ allows

- a *set* of start states, $Q_0 \subseteq Q$
- set-valued transition function, $\hat{\nu} : Q \times I \rightarrow 2^Q$
- at any stage automaton may be in a set of states

Dynamics:

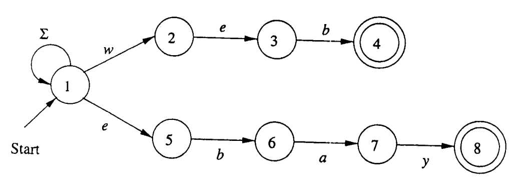
- One move: NFA in q receives symbol a and nondeterministically enters any one of the states in the set $\hat{\nu}(q, a)$
- On input $w = a_1 a_2 \cdots a_n$: NFA in state r_0 nondeterministically sequences through $r_1, r_2, \ldots, r_n$ such that

$$r_{k+1} \in \hat{\nu}(r_k, a_k)$$

  Sequence is *run* of NFA over w: $r_0 \xrightarrow{\mathtt{a}_1} r_1 \xrightarrow{\mathtt{a}_2} \cdots r_{n-1} \xrightarrow{\mathtt{a}_n} r_n$
- In general, NFA has many runs over each string; DFA, only one

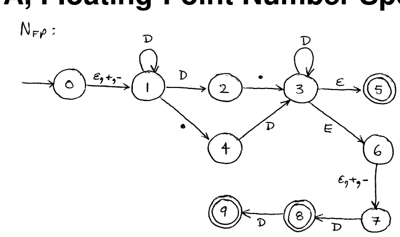
NFA Examples

Example 19 (Pattern Search [Hopcroft, Motwani, Ullman])



Accepts strings ending in `web` *or* `ebay`

Example 20 (ε-NFA, Floating-Point Number Specification)



optional sign, digit before or after decimal, optional exponent (+sign)

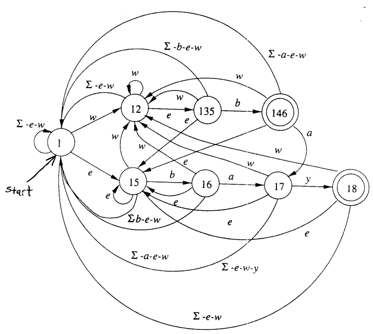
Subset Construction

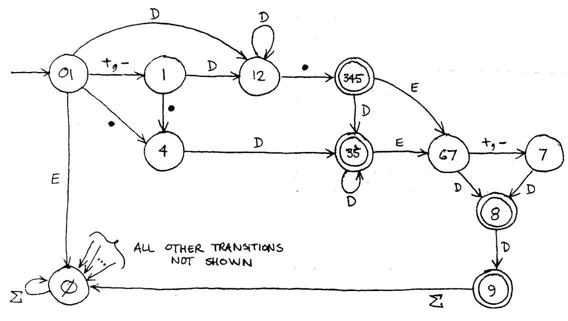
Convert any NFA into a DFA

$$N = (Q_N, I, \hat{\nu}, Q_0, F_N) \Longrightarrow D = (Q_D, I, \nu_D, q_0, F_D)$$

Idea: Keep track of *set* of states NFA can be in

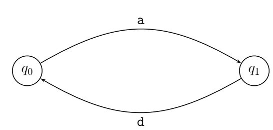
$$Q_D = 2^{Q_N},\ q_0 = Q_0; \qquad \nu_D(R, i) = \bigcup_{r \in R} \hat{\nu}(r, i); \qquad F_D = \{R \in 2^Q \mid R \cap F_N \neq \emptyset\}$$





ω-Automata

Machines that process infinite sequence of symbols
Appropriate for modeling reactive processes (e.g., OS, server)



***ω-string* / *ω-word* (over alphabet I):** infinite-length sequence of symbols from I

$$\mathtt{a}^\omega \equiv \mathtt{aaaa}\cdots \qquad \mathtt{d}^\omega \equiv \mathtt{dddd}\cdots \qquad (\mathtt{ad})^\omega \equiv \mathtt{adad}\cdots$$

***ω-languages*:** sets of ω-words.

***ω-automata*:** act as finite automata (can be deterministic or not)

For server, the run over $(\mathtt{ad})^\omega$ is $q_0, q_1, q_0, q_1, q_0, q_1, \ldots$

Difference is acceptance conditions; flavors: Büchi, Muller, Rabin, etc.
They involve states visited infinitely often, e.g., q_0 above

Adding Continuous Phenomena

Finite automata **plus continuous phenomenon**

- *Global Time*: add a universal clock (with unity rate)
- *Timed Automata*: add a set of such clocks and ability to reset them
- *Skewed-Clock Automata*: each clock variable has a different rational rate (uniform over all locations)[6]
- *Multi-Rate Automata*: each variable can take on different, rational rates in each location
- *Multi-Rectangular Automata*: same, but rectangular inclusions

$\Longrightarrow$ “Linear” Hybrid Automata

[6]“Discrete states” $\equiv$ *modes*, *phases*, or *locations*

Global Time

FA usually: “abstract time,” only ordering of symbols/“events” matters

Add time: associate time t_k at which kth transition occurs

$$\begin{aligned} q(t_{k+1}) &= \nu(q(t_k), i(t_k)) \\ o(t_k) &= \eta(q(t_k), i(t_k)) \end{aligned}$$

Make continuous-time: variables are piecewise continuous functions

$$\begin{aligned} q^+(t) &= \nu(q(t), i(t)) \\ o(t) &= \eta(q(t), i(t)) \end{aligned}$$

$q(t)$ changes only when input symbol $i(t)$ changes

Timed Automata (1)

timed word: sequence of symbols $+$ their increasing times of occurrence

$w = (i_1, t_1), (i_2, t_2), \ldots, (i_N, t_N)$
$i_k \in I; \quad t_k \in \mathbf{R}_+, t_{k+1} > t_k$

For server: $w = (\mathtt{a}, 0),\ (\mathtt{d}, 2),\ (\mathtt{a}, 3),\ (\mathtt{d}, 4),\ (\mathtt{a}, 5),\ (\mathtt{d}, 8),\ (\mathtt{a}, 9),\ (\mathtt{d}, 16)$

symbol sequence: $\sigma \;=\; \mathtt{a},\ \mathtt{d},\ \mathtt{a},\ \mathtt{d},\ \mathtt{a},\ \mathtt{d},\ \mathtt{a},\ \mathtt{d}$

time sequence: $\tau \;=\; 0,\ 2,\ 3,\ 4,\ 5,\ 8,\ 9,\ 16$

$w = (\sigma, \tau)$; Untime$(w) = \sigma$

timed ω-word: infinite sequence plus time *progresses* without bound

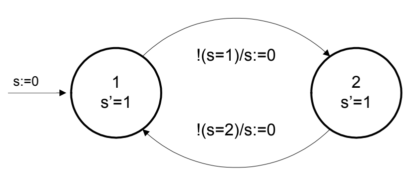
Not valid: $1/2, 3/4, 7/8, 15/16, 31/32, \ldots$
Condition avoids so-called *Zeno behavior*

timed language: set of timed words

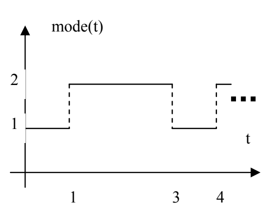
$$L_{\text{bounded response time}} = \{(\sigma, \tau) \mid \sigma_{2i-1} = \mathtt{a}, \sigma_{2i} = \mathtt{d}, \tau_{2i} < \tau_{2i-1} + 2\}$$
$$\text{Untime}(L_{\text{bounded response time}}) = \{(\mathtt{ad})^\omega\}$$

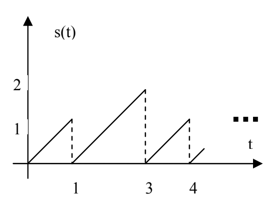
Timed Automata (2)

timed automaton: same structure as FA adding
(i) finite number of real-valued clocks (all unity rate)
(ii) ability to reset clocks, test clock constraints when traversing edges



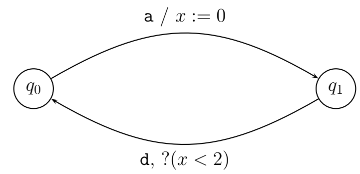
Notes
- s is a clock.
- !(s=2) means you must traverse the edge when s is equal to 2.
- s:=0 denotes setting the clock to 0.
- You could add output to the edges.





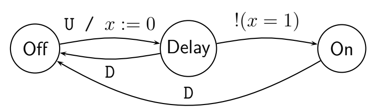
Timed Automata (3)

Example 21 (Bounded Response Time [Alur-Dill, *TCS*, 1994])
Every "arrival" needs to "depart" within two seconds



Accepted word: $(\mathtt{a}, 0),\ (\mathtt{d}, 1.5),\ (\mathtt{a}, 2),\ (\mathtt{d}, 3.5),\ (\mathtt{a}, 4),\ (\mathtt{d}, 5.5),\ \cdots$
Not accepted: $(\mathtt{a}, 0),\ (\mathtt{d}, 1.5),\ (\mathtt{a}, 2),\ (\mathtt{d}, 4.5),\ \cdots$
Not accepted: $(\mathtt{a}, 0),\ (\mathtt{a}, 1.5),\ \cdots$

Example 22 (Switch with Delay [Maler-Yovine, 1996])
`U`, `D` *switch "On", "Off"; models: transistors, relays, pneumatic valves*



Timed Automata Theory[7]

clock constraint has form $\chi := (x \leq c) \mid (c \leq x) \mid \neg\chi_0 \mid \chi_1 \wedge \chi_2$

x, clock variable; c, rational constant; χ_i, valid clock constraints

Can build up more complicated tests:

$$\begin{array}{rcl} (x = c) & \Longleftarrow & (x \leq c) \wedge (c \leq x) \\ (x < c) & \Longleftarrow & (x \leq c) \wedge \neg(x = c) \\ \chi_1 \vee \chi_2 & \Longleftarrow & \neg(\neg\chi_1 \wedge \neg\chi_2) \\ \text{True} & \Longleftarrow & (x \leq c) \vee (c \leq x) \end{array}$$

Rich and beautiful theory:

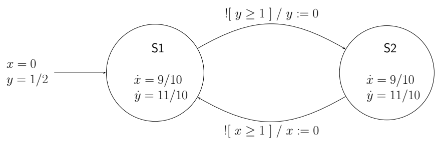
- Closure properties, decidability results
- E.g., a timed automaton can be mimicked by an ω-automata (called a *region* automata because it operates on clock regions), leading to an effective decision problem for language emptiness

[7] Seminal reference: R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994

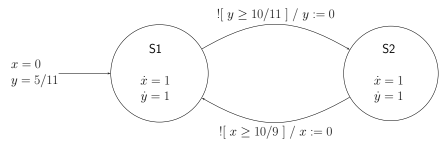
Skewed-Clock Automata

timed automaton: $\dot{x}_i = 1$ for all clocks and all locations

skewed-clock automaton: $\dot{x}_i = k_i$ where each k_i is a rational number



Skewed-Clock Automaton



Equivalent Timed Automaton

Remark 1 *Skewed-clock automata are equivalent to timed automata*

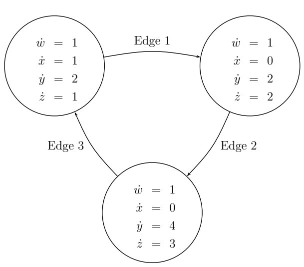
Proof 1 *Timed automaton is a special skewed-clock automaton wherein each $k_i = 1$*

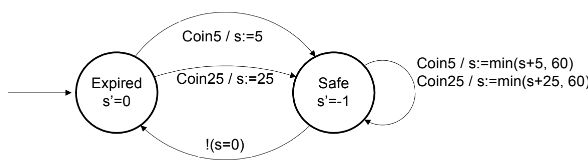
For converse:

1. $k_i = 0$: $x_i(t)$ *remains constant and any conditions involving it are uniformly true or false (and thus may be reduced or removed using the rules of logic)*
2. $k_i \neq 0$: Note that $x_i(t) = x_i(0) + k_i t$, *so* $x_i(t)/k_i = x_i(0)/k_i + t$
   Thus, divide every constant that x_i *is compared to by* k_i,
   and then use associated clock $\tilde{x}_i = x_i/t$, *with* $\dot{\tilde{x}}_i = 1$

Multi-Rate Automata

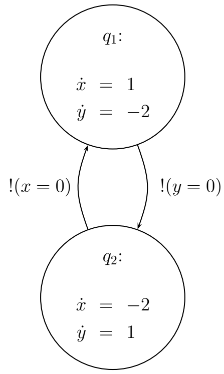
multi-rate automaton: $\dot{x}_i = k_{i,q}$ at location q (each $k_{i,q}$ is rational)

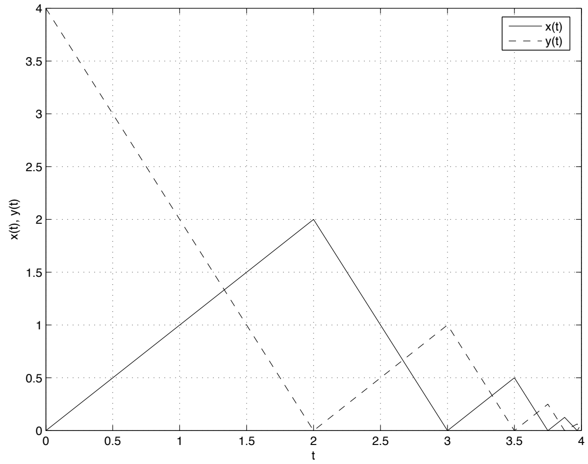




- Some vars. have the same rates in all states, e.g., w
- Some vars. are *stopwatches* (derivative either 0 or 1), e.g., x
- Not all dynamics change at every transition
- Parking meter has "non-linear" (non-TA) dynamics
- Skewed-clock automaton is special case with $k_{i,q} = k_i$ for all q

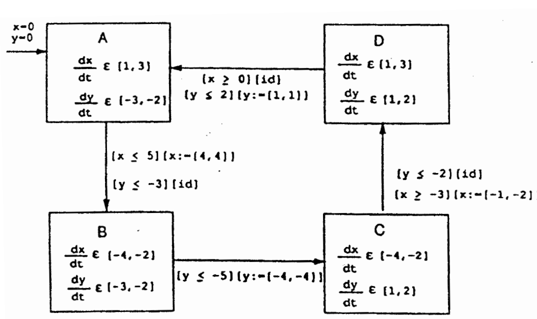
Zeno Behavior





Start in q_1 at $(x, y) = (0, 4)$
Events pile up at $t = 4$

Multi-Rectangular Automata



Rectangle in $\mathbf{R}^n$: $[r_1, s_1] \times [r_2, s_2] \times \cdots \times [r_n, s_n]$

E.g., in $\mathbf{R}^2$: $[0, 1] \times [1, 3]$; $[-\infty, \infty] \times [0, 1]$; $[-2, -2] \times [3, 5]$

Initial continuous states: $init(q_0)$ is a rectangle
Continuous dynamics: the inclusions, $flow(q)$, are rectangles
Guard conditions, $guard(e)$: rectangles
Reset relations, $reset(e)$: rectangle or identity (“id”) for each variable

Initialized Multi-R— Automata

initialized multi-r— automaton: variable must be reset when traversing an edge if its dynamics changes while crossing that edge

Example: multi-rectangular automation on previous page
Counterexample: multi-rate automation w/Zeno behavior

Remark 2 (Henzinger-Kopke-Puri-Varaiya, 1998)
An initialized multi-rate automaton can be converted into a timed automaton

Proof 2 *Idea: Use same trick as in Remark 1, as many times for each variable as it has different rates (the fact that the automaton is "initialized" is crucial)*

Remark 3 (Henzinger-Kopke-Puri-Varaiya, 1998)
An initialized multi-rectangular automaton can be converted to an initialized multi-rate automaton (and hence a timed automaton)

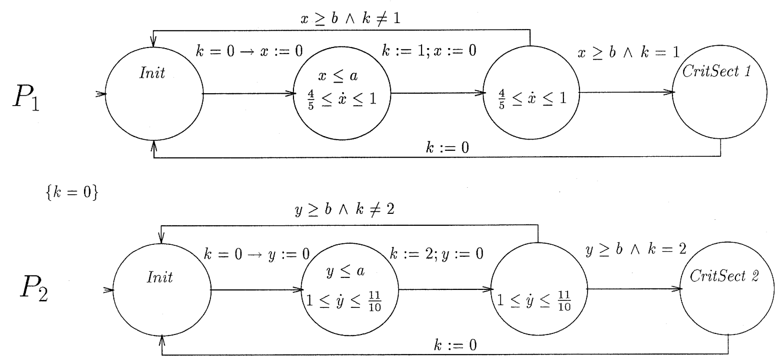
Proof 3 *Idea: replace each continuous variable, say x, with two variables, say x_l and x_u, that track lower and upper bounds on its value, resp.; then, invoke Remark 2*

Linear Hybrid Automata (LHA)

Solutions are linear (not vector field!)

- discrete transition system on finite set, Q, of modes/locations (FA)
- finite number of real-valued vars., with "nice" rate/jump constraints[8]

Example 23 (Fischer's MEX Protocol [Henzinger *et al.*])



[8]So the reachable set at each step is a union of polyhedra [Alur *et al.*, *Theoretical Computer Science*, 138:3–34, 1995]

LHA: Technical Definition (1)

Expressions over a set of variables Z

Linear Expression: linear combination of the vars. with rational coeffs.

$$1/2x + 24/5y, \qquad z + 5t - 6 + y$$

Linear Inequality: inequality between linear expressions

$$x \geq 0, \qquad 4 + 2t \leq 2/3x$$

Convex Predicate: a finite conjunction (“and”) of linear inequalities

$$(x \geq 3)\ \&\&\ (3y \geq z + 5/3)$$

Predicate: a finite disjunction (“or”) of convex predicates

$$((x \geq 3)\&\&(3y \geq z + 5/3))\ \ ||\ \ ((x \geq 0)\&\&(y < 1))$$

LHA: Technical Definition (2)

$$\begin{aligned} X &= \{x_1, x_2, \ldots, x_n\} & \longleftarrow \text{continuous variables} \\ \dot{X} &= \{\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n\} & \longleftarrow \text{continuous updates} \\ X' &= \{x'_1, x'_2, \ldots, x'_n\} & \longleftarrow \text{discrete updates (i.e., resets)} \end{aligned}$$

$init(q)$ is a predicate on X
$inv(q)$ is a convex predicate on X (the *invariant* for each q)
$flow(q)$ is a convex predicate on $\dot{X}$

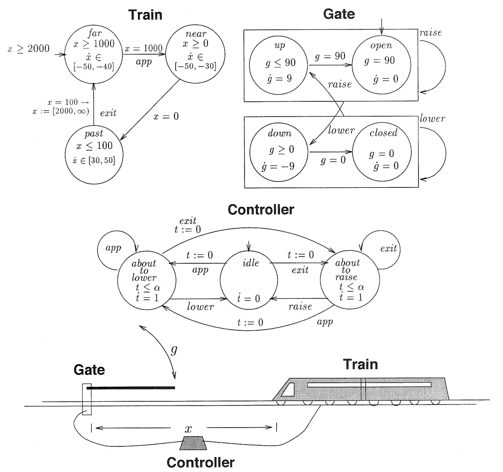
$\dot{x} \in [10,\ 20]$ is equivalent to $(\dot{x} \geq 10)\&\&(\dot{x} \leq 20)$

$reset(e)$ is a convex predicate on $X \cup X'$

$$1 <= x', \qquad x' < 2, \qquad t' >= x + 3, \qquad y' = 0$$

If inv, $flow$, $reset$ are predicates (vs. convex predicates),
we have “or” transitions involved
To handle this, split the states/edges to model the disjunctions

HyTech Train-Gate Example



$\alpha < 49/5$	Number of locations	Number of transitions	CPU time
When the train is within 10 meters to the gate, the gate is always fully closed.	36	90	0.2 sec.

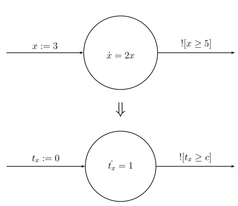
[Source: Henzinger, Ho, Wong-Toi. HyTech Demo. `embedded.eecs.berkeley.edu/research/hytech`]

Non-Linear Hybrid Automata

Non-Linear: anything not linear by HyTech's definition

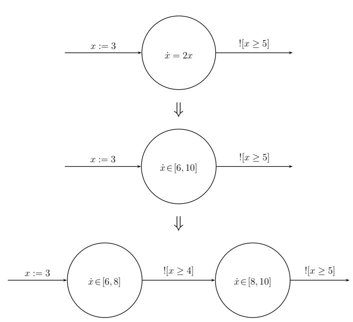
Two ways to deal with this

1. Easy way out!
   (a) Reduce or transform your HA into a LHA: *clock translation*
   (b) Approximate it by a LHA: *linear phase portrait approximation*
2. Harder: develop richer theory, comp. tools for a larger class of HA



$$5 = e^{2c} \cdot 3$$
$$c = \ln(5/3)/2$$

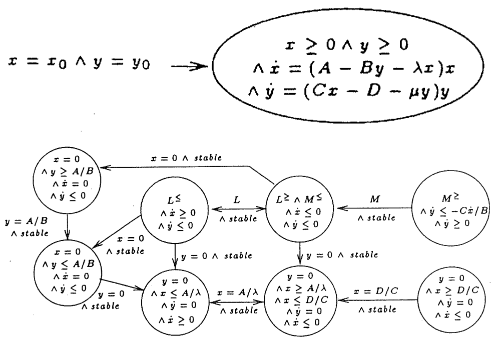
Clock Translation



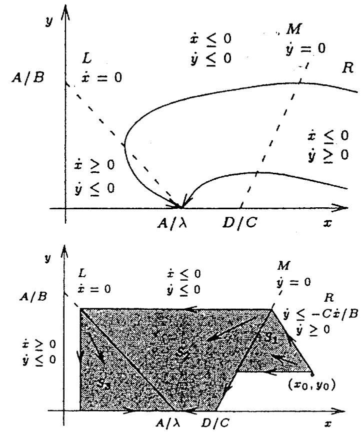
LPP Approx., Successive Refinement

Phase Portrait Approximation

Predator-Prey Equations: nonlinear (top) and linear (bottom)



Hybrid Automata



Phase Portraits

[Henzinger *et al.*, Algorithmic Analysis of Nonlinear Hybrid Systems, *IEEE Trans. Auto. Cont.*, 43(4):540–554, 1998]

Summary

- Broad Hybrid Systems Modeling Definition / Motivation
- The First Hybrid Dynamicist: Laplace
- Many Hybrid Systems Examples
- Mathematical Models of HS
  - From Continuous Side:

    ODEs $+$ Discrete Phenomena
    $\Longrightarrow$ Hybrid Dynamical Systems

  - From Discrete Side:

    FA $+$ Continuous Phenomena
    $\Longrightarrow$ Hybrid Automata

Going Further

Early HS models: Witsenhausen, Tavernini, Brockett, Nerode-Kohn, Antsaklis-Stiver-Lemmon, Back *et al.* ⟵ all reviewed/compared in [Branicky, ScD Thesis, 1995]

Early related work:

- variable-structure systems (Utkin), systems with impulse effect, jump-linear systems, cell-to-cell mapping (Hsu), iterated function systems
- DES (Ramadge-Wonham), statecharts (Harel), reactive systems (Manna-Pnueli)

More recent HS frameworks:

- hybrid I/O automata: Lynch, Segala, Vaandrager, *et al.*
- linear complementarity: Heemels, van der Schaft, Schumacher, *et al.*
- mixed logical dynamical systems: Bemporad, Morari, *et al.*
- hybrid Petri nets, stochastic hybrid systems, . . .

HS simulation, verification, specification languages/tools:

- Omola/Omsim; SHIFT, Ptolemy; Modelica; . . .
- HyTech, UPAAL, KRONOS, CheckMate, d/dt, Charon, PHAVer, HYSDEL, . . . [`wiki.grasp.upenn.edu/~graspdoc/wiki/hst`]

References

[1] MS Branicky. Introduction to hybrid systems. In D Hristu-Varsakelis and WS Levine (eds), *Handbook of Networked and Embedded Control Systems*, Boston: Birkhauser, 2005

[2] MS Branicky. EECS 381/409: Discrete event and hybrid systems. Course notes, Case Western Reserve University, 1998–2005

[3] MS Branicky, VS Borkar, SK Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automatic Control*, 43(1):31–45, 1998

[4] MS Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* ScD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995

These/other references available via `dora.case.edu/msb`