1st HYCON PhD School on Hybrid Systems

www.unisi.it

www.ist-hycon.org

Models for Hybrid Systems

Bart De Schutter

TU Delft, The Netherlands

b.deschutter@its.tudelft.nl

HYSCOM

IEEE CSS Technical Committee on Hybrid Systems

Information Society Technologies

Siena, July 19-22, 2005 - Rectorate of the University of Siena

1st HYCON Summer School on Hybrid Systems
Siena, Italy, July 13–18, 2005

Models for Hybrid Systems

Bart De Schutter (b.deschutter@dcsc.tudelft.nl)
Delft Center for Systems and Control
Delft University of Technology

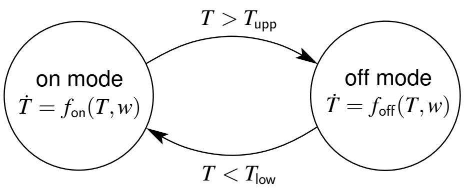
Models for Hybrid Systems

Overview

1. Hybrid systems

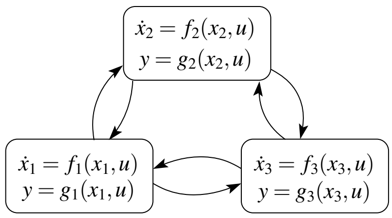
1.1 Informal definition

- Hybrid = combination of continuous and discrete dynamics
- Temperature control system:



1.2 More formal definition

- System can be in one of several modes
- In each mode: behavior described by system of difference or differential equations
- Mode switches due to occurrence of "events"

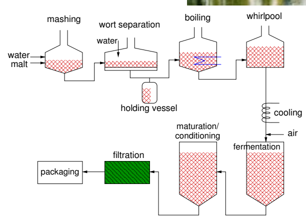


1.2 More formal definition (continued)

- At switching time instant:
  → possible state reset or state dimension change
- Mode transitions may be caused by
  - external control signal
  - internal control signal
  - dynamics of system itself (crossing of boundary in state space)

1.3 Examples

- Hierarchical control in process industry
- Telecommunication systems
- Manufacturing systems
- Air traffic coordination and control
- Batch processes
  (e.g. beer brewing)



Human intervention in smooth systems → hybrid

1.3 Examples (continued)

- Traffic control
- Automatic platooning
- Evolution of rigid bodies (contact/no contact)
- Electrical networks (switching, diodes)
- Fermentation process (lag, growth, stationary, inactivation)
- Saturation, hysteresis
- Actuator and sensor failures

Switching between dynamical regimes → hybrid

1.4 Challenges

- Analysis and control
- Nowadays:
  - often still heuristic & ad-hoc
  - often focus still exclusively on either continuous or discrete dynamics

  → structured approach necessary
- Consider hybrid nature of systems
- Combination of systems & control, computer science, mathematics, and simulation

2. Hybrid system models

2.1 Introduction

- Continuous-state / discrete-state
- Continuous-time / discrete-time
- Time-driven / event-driven
  - time-driven → state changes as time progresses, i.e., continuously (for CT), or at every tick of clock (for DT)
  - event-driven → state changes due to occurrence of event event:
    - start or end of an activity
    - asynchronous (occurrence times not necessarily equidistant)

Combinations → "hybrid"

2.2 Models for time-driven systems

- Continuous-time time-driven systems:

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

- Discrete-time (or sampled) time-driven systems:

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = g(x(k), u(k))$$

2.3 Models for event-driven systems

- ☞ Automata
- ☞ Petri nets
- (max,+) algebraic models
- Markov chains / Markov processes
- Extended state machines
- Generalized semi-Markov processes
- Networks of waiting queues
- ...

⇒ no general framework

Note: see also lecture on "Discrete-event modeling and diagnosis of quantized systems"

2.3 Models for event-driven systems (continued)

- No general framework (similar situation for hybrid systems)
- Basic trade-off:

modeling power ↔ decision power

⇒ application-specific

2.4 Models for hybrid systems

- Timed or hybrid Petri nets
- Differential automata
- Hybrid automata
- Brockett's model
- Mixed logical dynamic models
- Duration calculus
- Real-time temporal logics
- Timed communicating sequential processes
- Switched bond graphs
- ../..

2.4 Models for hybrid systems (continued)

- Computer simulation models
- Predicate calculus
- Piecewise-affine models
- Timed automata
- ...

Note: focus in this lecture is on non-stochastic models
(see also lectures on "Stochastic Hybrid Systems")

2.5 Models for hybrid systems — Issues

$\Rightarrow$ no general modeling & analysis framework

modeling power $\leftrightarrow$ decision power

\+ computational complexity (NP-hard, undecidable)

$\Rightarrow$ special subclasses
hierarchical / modular approach

Intermezzo: Undecidable and NP-hard problems

- **Undecidable problems**
  $\rightarrow$ no algorithm at all can be given for solving the problem in general
- **NP-complete and NP-hard problems**
  - *decision problem*: solution is either "yes" or "no"
    e.g., traveling salesman decision problem:
    Given a network of cities, intercity distances, and a number B, does there exist a tour with length $\leqslant B$?
  - *search problem*
    e.g., traveling salesman problem:
    Given a network of cities, intercity distances, what is the shortest tour?

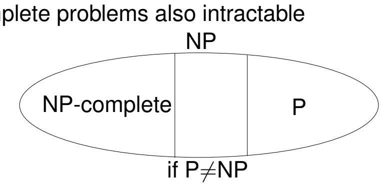
P and NP-complete decision problems

- time complexity function $T(n)$: largest amount of time needed to solve problem instance of size n (*worst case!*)
- polynomial time algorithm:
  $$T(n) \leqslant |p(n)| \qquad \text{for some polynomial } p$$
  $\rightarrow$ class P: solvable by polynomial time algorithm
- nondeterministic computer:
  - guessing stage (tour)
  - checking stage (compute length of tour + compare it with B)

  $\rightarrow$ class NP: "*nondeterministically polynomial*"
  i.e., time complexity of checking stage is polynomial

P and NP-complete decision problems

- Each problem in NP can be solved in exponential time: $T(n) \leqslant 2^{n^k}$
- NP-complete problems: "hardest" class in NP:
  - any NP-complete problem solvable in polynomial time
    $\Rightarrow$ every problem in NP solvable in polynomial time
  - any problem in NP intractable
    $\Rightarrow$ NP-complete problems also intractable



NP-hard problems

- decision problem is NP-complete ⇒ search problem is NP-hard
- NP-hard problems: at least as hard as NP-complete problems
  - NP-complete (decision problem)
    → solvable in polynomial time *if and only if* P = NP
  - NP-hard (search problem)
    → cannot be solved in polynomial time *unless* P = NP

Examples of NP-hard and undecidable problems

- Consider simple hybrid system:

$$x(k+1) = \begin{cases} A_1 x(k) & \text{if } c^T x(k) \geqslant 0 \\ A_2 x(k) & \text{if } c^T x(k) < 0 \end{cases}$$

  → deciding whether system is stable or not is NP-hard
- Given two Petri nets, do they have the same reachability set?
  → undecidable

Back to the main topic — Hybrid system models

- Many modeling frameworks for hybrid systems
  ⇒ trade-off: modeling power ↔ decision power, tractability
- Hybrid automata:
  - very general, high modeling power, but low decision power
  - analysis and control → computationally hard
    (NP-hard, undecidable problems)

Hybrid system models (continued)

- Computer simulation and verification tools: Modelica, HyTech, KRONOS, Chi, 20-sim, UPPAAL, ...
  + simulation models can represent plant with high degree of detail (high modeling power)
  - computationally very demanding for large systems
  - difficult to understand from simulation how behavior depends on model parameters
- In this lecture: special classes of hybrid systems for which *tractable* analysis and control design techniques are available (→ see next lectures)

3. Models for event-driven systems

3.1 Automata

Automaton

Automaton is defined by triple $\Sigma = (\mathcal{Q}, \mathcal{U}, \phi)$ with

- $\mathcal{Q}$: finite or countable set of discrete states
- $\mathcal{U}$: finite or countable set of discrete inputs (“input alphabet”)
- $\phi : \mathcal{Q} \times \mathcal{U} \mapsto P(\mathcal{Q})$: partial *transition function*.

where $P(\mathcal{Q})$ is power set of $\mathcal{Q}$ (set of all subsets)

Finite automaton: $\mathcal{Q}$ and $\mathcal{U}$ finite

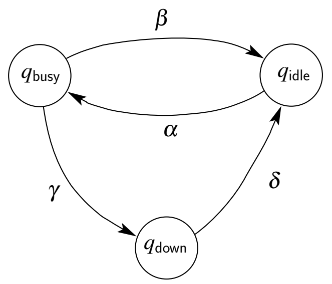
3.1 Automata (continued)

Evolution of automaton

- Given state $q \in \mathcal{Q}$ and discrete input symbol $u \in \mathcal{U}$, transition function ϕ defines collection of next possible states: $\phi(q,u) \subseteq \mathcal{Q}$
- If each set of next states has 0 or 1 element:
  → “deterministic” automaton
- If some set of next states has more than 1 element:
  → “non-deterministic” automaton

3.1 Automata (continued)

Deterministic automaton

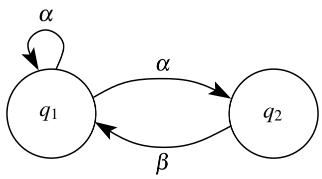


$\phi(q_{\text{busy}}, \beta) = \{q_{\text{idle}}\}$ $\phi(q_{\text{idle}}, \alpha) = \{q_{\text{busy}}\}$

$\phi(q_{\text{busy}}, \gamma) = \{q_{\text{down}}\}$ $\phi(q_{\text{down}}, \delta) = \{q_{\text{idle}}\}$

3.1 Automata (continued)

Non-deterministic automaton



$\phi(q_1, \alpha) = \{q_1, q_2\}$ $\phi(q_2, \beta) = \{q_1\}$

4. Hybrid automata

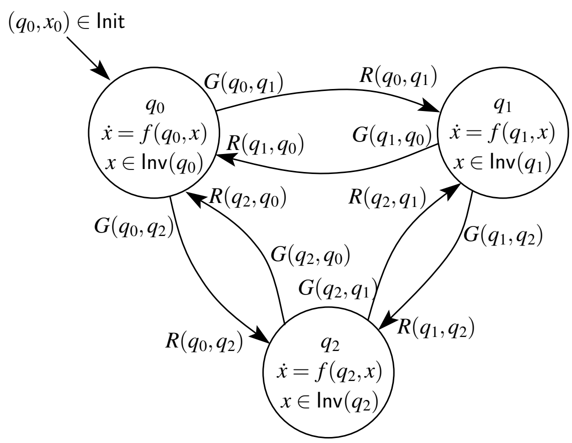
4.1 Definition

Hybrid automaton H is collection $H = (Q, X, f, \mathsf{Init}, \mathsf{Inv}, E, G, R)$ with

- $Q = \{q_1, \ldots, q_N\}$ is finite set of discrete states or *modes*
- $X = \mathbb{R}^n$ is set of continuous states
- $f : Q \times X \to X$ is vector field
- $\mathsf{Init} \subseteq Q \times X$ is set of initial states
- $\mathsf{Inv} : Q \to P(X)$ describes the *invariants*
- $E \subseteq Q \times Q$ is set of edges or *transitions*
- $G : E \to P(X)$ is *guard condition*
- $R : E \to P(X \times X)$ is *reset map*

Hybrid automaton $H = (Q, X, f, \mathsf{Init}, \mathsf{Inv}, E, G, R)$

- Hybrid state: (q, x)
- Evolution of continuous state in mode q: $\dot{x} = f(q, x)$
- Invariant Inv: describes conditions that continuous state has to satisfy at given mode
- Guard G: specifies subset of state space where certain transition is enabled
- Reset map R: specifies how new continuous states are related to previous continuous states



Evolution of hybrid automaton

- Initial hybrid state $(q_0, x_0) \in \mathsf{Init}$
- Continuous state x evolves according to

  $$\dot{x} = f(q_0, x) \quad \text{with } x(0) = x_0$$

  discrete state q remains constant: $q(t) = q_0$
- Continuous evolution can go on as long as $x \in \mathsf{Inv}(q_0)$
- If at some point state x reaches guard $G(q_0, q_1)$, then
  - transition $q_0 \to q_1$ is enabled
  - discrete state *may* change to q_1, continuous state then jumps from current value x^- to new value x^+ with $(x^-, x^+) \in R(q_0, q_1)$
- Next, continuous evolution resumes and whole process is repeated

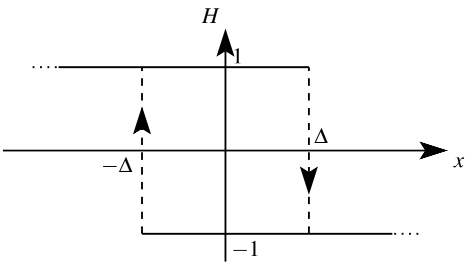
4.2 Examples of hybrid automata

1. Hysteresis
2. Water-level monitor

Hysteresis

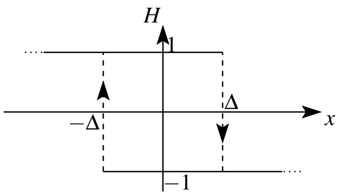
Control system with hysteresis element in the feedback loop :

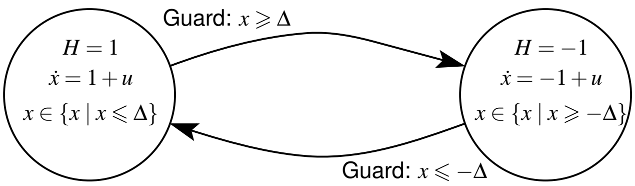
$$\dot{x} = H(x) + u$$



Hysteresis (continued)

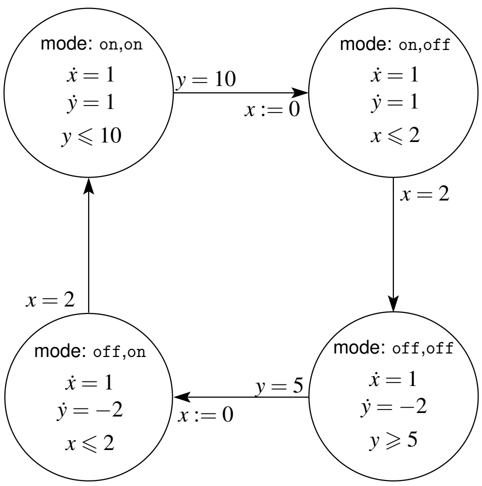
$$\dot{x} = H(x) + u$$





Water-level monitor

- variables:
  - $y(t)$: water level, continuous
  - $x(t)$: time elapsed since last signal was sent by monitor, cont.
  - $P(t)$: status of pump, $\in \{\texttt{on}, \texttt{off}\}$
  - $S(t)$: nature of signal last sent by monitor, $\in \{\texttt{on}, \texttt{off}\}$
- dynamics of system:
  - water level rises 1 unit per second when pump is on and falls 2 units per second when pump is off
  - when water level rises to 10 units, monitor sends switch-off signal; after delay of 2 seconds pump turns off
  - when water level falls to 5 units, monitor sends switch-on signal; after delay of 2 seconds pump switches on



y: water level
x: time since last signal

5. PWA systems and related model classes

1. Piecewise affine systems (PWA)
2. Mixed Logical Dynamical systems (MLD)
3. Linear Complementarity systems (LC)
4. Max-Min-Plus-Scaling systems (MMPS)
5. Equivalence of MLD, LC, ELC, PWA and MMPS systems

5.1 Piecewise affine (PWA) systems

- PWA systems are described by

$$\begin{aligned} x(k+1) &= A_i x(k) + B_i u(k) + f_i \\ y(k) &= C_i x(k) + D_i u(k) + g_i \end{aligned} \quad \text{for } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \Omega_i,\ i = 1, \ldots, N$$

- $\Omega_1, \ldots, \Omega_N$: convex polyhedra (i.e., given by finite number of linear inequalities) in input/state space, non-overlapping interiors
- PWA can be used as approximation of nonlinear model

$$\begin{aligned} x(k+1) &= \mathscr{N}_x(x(k), u(k)) \\ y(k) &= \mathscr{N}_y(x(k), u(k)) \end{aligned}$$

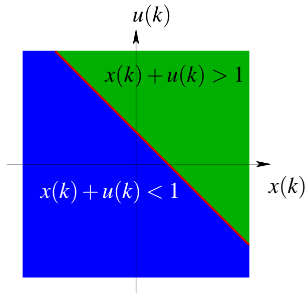
$\rightarrow$ "simplest" extension of linear systems that can still model non-linear & non-smooth processes with arbitrary accuracy + are capable of handling hybrid phenomena

Example of PWA model

Integrator with upper saturation:

$$x(k+1) = \begin{cases} x(k) + u(k) & \text{if } x(k) + u(k) \leqslant 1 \\ 1 & \text{if } x(k) + u(k) \geqslant 1 \end{cases}$$

$$y(k) = x(k)$$



5.2 Mixed Logical Dynamical (MLD) systems

Preliminaries

- Boolean operators:
  $\wedge$ (and), $\vee$ (or), $\sim$ (not), $\Rightarrow$ (implies), $\Leftrightarrow$ (iff), $\oplus$ (xor)

X_1	X_2	$X_1 \wedge X_2$	$X_1 \vee X_2$	$\sim X_1$	$X_1 \Rightarrow X_2$	$X_1 \Leftrightarrow X_2$	$X_1 \oplus X_2$
T	T	T	T	F	T	T	F
T	F	F	T	F	F	F	T
F	T	F	T	T	T	F	T
F	F	F	F	T	T	T	F

- Properties:
  - $X_1 \Rightarrow X_2$ is same as $\sim X_1 \vee X_2$
  - $X_1 \Rightarrow X_2$ is same as $\sim X_2 \Rightarrow \sim X_1$
  - $X_1 \Leftrightarrow X_2$ is same as $(X_1 \Rightarrow X_2) \wedge (X_2 \Rightarrow X_1)$

- Associate with literal X_i logical variable $\delta_i \in \{0,1\}$:
  $\delta_i = 1$ iff $X_i =$ T, $\delta_i = 0$ iff $X_i =$ F
  $\rightarrow$ compound statement can be transformed into *linear integer program*
- Examples:
  - $X_1 \wedge X_2$ equivalent to $\delta_1 = \delta_2 = 1$
  - $X_1 \vee X_2$ equivalent to $\delta_1 + \delta_2 \geqslant 1$
  - $\sim X_1$ equivalent to $\delta_1 = 0$
  - $X_1 \Rightarrow X_2$ equivalent to $\delta_1 - \delta_2 \leqslant 0$
  - $X_1 \Leftrightarrow X_2$ equivalent to $\delta_1 - \delta_2 = 0$
  - $X_1 \oplus X_2$ equivalent to $\delta_1 + \delta_2 = 1$
- For $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $x \in \mathscr{X}$ with $\mathscr{X}$ bounded, define

$$M \stackrel{\text{def}}{=} \max_{x \in \mathscr{X}} f(x) \qquad m \stackrel{\text{def}}{=} \min_{x \in \mathscr{X}} f(x)$$

- Equivalences:
  - $[f(x) \leqslant 0] \wedge [\delta = 1]$ true iff $f(x) - \delta \leqslant -1 + m(1-\delta)$
  - $[f(x) \leqslant 0] \vee [\delta = 1]$ true iff $f(x) \leqslant M\delta$
  - $\sim[f(x) \leqslant 0]$ true iff $f(x) \geqslant \varepsilon$ (with ε machine precision)
  - $[f(x) \leqslant 0] \Rightarrow [\delta = 1]$ true iff $f(x) \geqslant \varepsilon + (m - \varepsilon)\delta$
  - $[f(x) \leqslant 0] \Leftrightarrow [\delta = 1]$ true iff $\begin{cases} f(x) \leqslant M(1-\delta) \\ f(x) \geqslant \varepsilon + (m-\varepsilon)\delta \end{cases}$
- Product $\delta_1 \delta_2$ can be replaced by auxiliary variable $\delta_3 = \delta_1 \delta_2$
  Since $[\delta_3 = 1] \Leftrightarrow [\delta_1 = 1] \wedge [\delta_2 = 1]$,

$$\delta_3 = \delta_1 \delta_2 \quad \text{is equivalent to} \quad \begin{cases} -\delta_1 + \delta_3 \leqslant 0 \\ -\delta_2 + \delta_3 \leqslant 0 \\ \delta_1 + \delta_2 - \delta_3 \leqslant 1 \end{cases}$$

- Product $\delta f(x)$ can be replaced by auxiliary real variable $y = \delta f(x)$ with $[\delta = 0] \Rightarrow [y = 0]$, $[\delta = 1] \Rightarrow [y = f(x)]$, or equivalently

$$\begin{cases} y \leqslant M\delta \\ y \geqslant m\delta \\ y \leqslant f(x) - m(1-\delta) \\ y \geqslant f(x) - M(1-\delta) \end{cases}$$

Mixed logical dynamical (MLD) systems

- $$\begin{aligned} x(k+1) &= Ax(k)+B_1u(k)+B_2\delta(k)+B_3z(k) \\ y(k) &= Cx(k)+D_1u(k)+D_2\delta(k)+D_3z(k) \\ E_1x(k)+E_2u(k)+E_3\delta(k)+E_4z(k) &\leqslant g_5, \end{aligned}$$
- $x(k) = [x_r^T(k)\ x_b^T(k)]^T$ with $x_r(k)$ real-valued, $x_b(k)$ boolean
  $z(k)$: real-valued auxiliary variables
  $\delta(k)$: boolean auxiliary variables
- Applications: PWA systems, systems with discrete inputs, qualitative inputs, bilinear systems, finite state machines
- Reference: A. Bemporad and M. Morari, “Control of systems integrating logic, dynamics, and constraints,” *Automatica*, vol. 35, no. 3, pp. 407–427, March 1999.

Example of an MLD system

- Consider PWA system:
  $$x(k+1) = \begin{cases} 0.8x(k)+u(k) & \text{if } x(k) \geqslant 0 \\ -0.8x(k)+u(k) & \text{if } x(k) < 0 \end{cases}$$
  where $x(k) \in [-10, 10]$ and $u(k) \in [-1, 1]$
- Associate binary variable $\delta(k)$ to condition $x(k) \geqslant 0$ such that $[\delta(k) = 1] \Leftrightarrow [x(k) \geqslant 0]$ or
  $$-m\delta(k) \leqslant x(k) - m$$
  $$-(M+\varepsilon)\delta \leqslant -x-\varepsilon$$
  where $M = -m = 10$, and ε is machine precision
- PWA system can be rewritten as
  $$x(k+1) = 1.6\delta(k)x(k) - 0.8x(k) + u(k)$$

- $x(k+1) = 1.6\delta(k)x(k) - 0.8x(k) + u(k)$
- Define new variable $z(k) = \delta(k)x(k)$ or
  $$\begin{aligned} z(k) &\leqslant M\delta(k) \\ z(k) &\geqslant m\delta(k) \\ z(k) &\leqslant x(k) - m(1-\delta(k)) \\ z(k) &\geqslant x(k) - M(1-\delta(k)) \end{aligned}$$
- PWA system now becomes
  $$x(k+1) = 1.6z(k) - 0.8x(k) + u(k)$$
  subject to linear constraints above $\rightarrow$ MLD

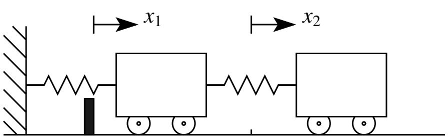
5.3 Linear Complementarity (LC) systems

- LC systems:
  $$\begin{aligned} x(k+1) &= Ax(k)+B_1u(k)+B_2w(k) \\ y(k) &= Cx(k)+D_1u(k)+D_2w(k) \\ v(k) &= E_1x(k)+E_2u(k)+E_3w(k)+g_4 \\ 0 \leqslant v(k) &\perp w(k) \geqslant 0 \end{aligned}$$
- $v(k)$, $w(k)$: “complementarity variables” (real-valued)
- Applications: constrained mechanical systems, electrical networks with ideal diodes, boost converter, dynamical systems with PWA relations, variable-structure systems, projected dynamical systems
- Example: two-carts system (*continuous*-time LC system)

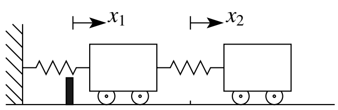
Example of an LC system

Two-carts system

- Two carts connected by spring
- Left cart attached to wall by spring;
  motion constrained by completely inelastic stop
  Stop is placed at equilibrium position of left cart
- Masses of carts and spring constants = 1

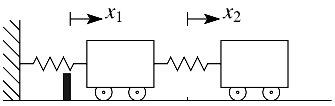


Two-carts system (continued)



- x_1, x_2: deviations of left and right cart from equilibrium position
- x_3, x_4: velocities of left and right cart
- z: reaction force exerted by stop
- Evolution: $$\begin{aligned} \dot{x}_1(t) &= x_3(t) \\ \dot{x}_2(t) &= x_4(t) \\ \dot{x}_3(t) &= -2x_1(t)+x_2(t)+z(t) \\ \dot{x}_4(t) &= x_1(t)-x_2(t) \end{aligned}$$

Two-carts system (continued)



To model stop:

- define $w(t) = x_1(t)$
- $w(t) \geqslant 0$ (since is position of left cart w.r.t. stop)
- force exerted by stop can act only in positive direction $\rightarrow z(t) \geqslant 0$
- if left cart not at stop $(w(t) > 0)$, reaction force vanishes: $z(t) = 0$
- if $z(t) > 0$ then cart must necessarily be at the stop: $w(t) = 0$

$$0 \leqslant w(t) \perp z(t) \geqslant 0$$

$\rightarrow$ (continuous-time) LC system

5.4 Max-Min-Plus-Scaling (MMPS) systems

- Max-min-plus-scaling expression:

$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l | \beta f_k$$

  with α, $\beta \in \mathbb{R}$ and f_k, f_l again MMPS expressions.
- Example: $5x_1 - 3x_2 + 7 + \max(\min(2x_1, -8x_2), x_2 - 3x_3)$
- MMPS systems:

$$x(k+1) = \mathscr{M}_x(x(k), u(k), d(k))$$
$$y(k) = \mathscr{M}_y(x(k), u(k), d(k))$$
$$\mathscr{M}_c(x(k), u(k), d(k)) \leqslant c,$$

  with $\mathscr{M}_x$, $\mathscr{M}_y$, $\mathscr{M}_c$ MMPS expressions
- $d(k)$: real-valued auxiliary variables

5.4 Max-Min-Plus-Scaling (MMPS) systems (continued)

- Applications:
  - discrete-event systems (also max-plus)
  - traffic-signal controlled intersection
  - railway networks
  - manufacturing systems
  - systems with soft & hard synchronization constraints
  - logistic systems

Example of MMPS system

- Integrator with upper saturation:

$$x(k+1) = \begin{cases} x(k) + u(k) & \text{if } x(k) + u(k) \leqslant 1 \\ 1 & \text{if } x(k) + u(k) \geqslant 1 \end{cases}$$
$$y(k) = x(k)$$

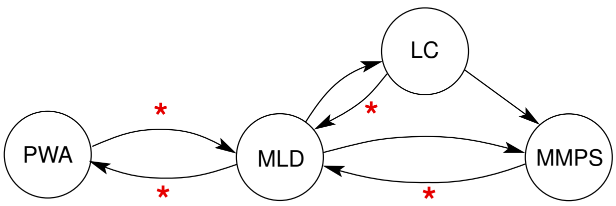
  can be recast as

$$x(k+1) = \min(x(k) + u(k), 1)$$
$$y(k) = x(k)$$

5.5 Equivalence of MLD, LC, PWA and MMPS systems

Equivalence between model classes $\mathscr{A}$ and $\mathscr{B}$:
for each model $\in \mathscr{A}$ there exists model $\in \mathscr{B}$ with same input/output behavior (+ vice versa)

MLD, LC, PWA and MMPS systems are equivalent:



Equivalence of MLD, LC, PWA and MMPS systems (cont.)

- Each subclass has own advantages:
  - stability criteria for PWA
  - control and verification techniques for MLD
  - control techniques for MMPS
  - conditions of existence and uniqueness of solutions for LC

  $\rightarrow$ transfer techniques from one class to other
- It depends on the application which class is best suited
- Reference: W.P.M.H. Heemels, B. De Schutter, and A. Bemporad, "Equivalence of hybrid dynamical models," *Automatica*, vol. 37, no. 7, pp. 1085–1091, July 2001.

6. Timed automata

- Timed automata involve simple continuous dynamics:
  - all differential equations of form $\dot{x} = 1$,
  - all invariants, guards, etc. involve comparison of real-valued states with constants (e.g., $x = 1$, $x < 2$, $x \geqslant 0$, etc.)
- Timed automata are limited for modeling physical systems.
- However, very well suited for encoding timing constraints such as “event A must take place at least 2 seconds after event B and not more than 5 seconds before event C”
- Applications: multimedia, Internet, audio protocol verification

6.1 Rectangular sets

- Subset of $\mathbb{R}^n$ set is called rectangular if can be written as finite boolean combination of constraints of form

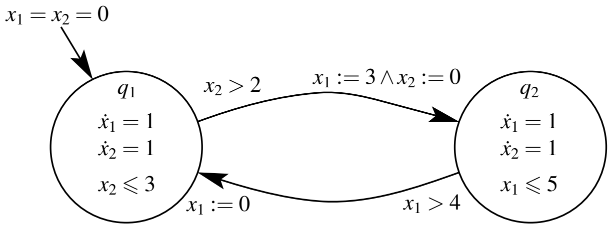
$$x_i \leqslant a, \quad x_i < b, \quad x_i = c, \quad x_i \geqslant d, \quad x_i > e$$

- Rectangular sets are “rectangles” or “boxes” in $\mathbb{R}^n$ whose sides are aligned with the axes, or unions of such rectangles/boxes
- Examples:
  - $\{(x_1, x_2) \mid (x_1 \geqslant 0) \wedge (x_1 \leqslant 2) \wedge (x_2 \geqslant 1) \wedge (x_2 \leqslant 2)\}$
  - $\{(x_1, x_2) \mid ((x_1 \geqslant 0) \wedge (x_2 = 0)) \vee ((x_1 = 0) \wedge (x_2 \geqslant 0))\}$
  - empty set (e.g., $\varnothing = \{(x_1, x_2) \mid (x_1 > 1) \wedge (x_1 \leqslant 0)\}$)
- However, set $\{(x_1, x_2) \mid x_1 = 2x_2\}$ is not rectangular

6.2 Timed automaton

- Timed automaton is hybrid automaton with following characteristics:
  - automaton involves differential equations of form $\dot{x}_i = 1$; continuous variables governed by this differential equation are called “clocks” or “timers”
  - sets involved in definition of initial states, guards, and invariants are rectangular sets
  - reset maps involve either rectangular set, or may leave certain states unchanged

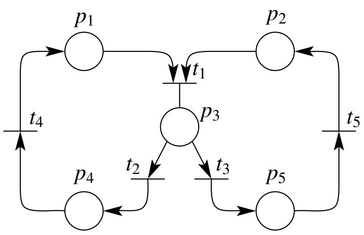
6.3 Example of timed automaton



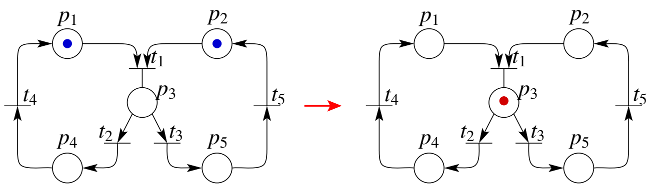
7. Timed Petri nets

7.1 Petri nets

- Graphical representation: bipartite directed multigraph
  - places (circles) → activities
  - transitions (bars) → events, actions



- marking → tokens are assigned to places
- execution of Petri net:
  - transition enabled if all input places ($^\bullet t$) contain at least 1 token
  - enabled transition can fire:
    - ∗ one token is removed from each input place ($^\bullet t$)
    - ∗ one token is deposited in each output place ($t^\bullet$)



- synchronization & choice

7.2 Timed Petri nets

- Untimed Petri net describes order in which events can occur, but no timing
- Timed Petri → timing, transition should be executed within certain time interval after it becomes enabled
  - discrete state variables (markings, $m_\theta(p)$)
  - continuous state variables (arrival times, $M_\theta(p)$)
- $M_\theta(p) := \{\theta_1, \ldots, \theta_{m_\theta(p)}\}$ with arrival times $\theta_1 \leqslant \theta_2 \leqslant \ldots \leqslant \theta_{m_\theta(p)}$ of $m_\theta(p)$ tokens in place p
- For each transition t we define interval $[L(t), U(t)]$

7.2 Timed Petri nets (continued)

- Transition t becomes enabled at

$$\max_{p \in {}^\bullet t} \min M_\theta(p)$$

- Then transition t may fire at some time

$$\theta \in [\max_{p \in {}^\bullet t} \min M_\theta(p) + L(t), \max_{p \in {}^\bullet t} \min M_\theta(p) + U(t)]$$

  provided t is enabled during whole interval
- If enabling condition is still valid at final time of firing interval, then transition is forced to fire
- Many techniques for untimed Petri nets can be extended to timed Petri nets
- However, many problems are undecidable or NP-hard

8. Summary

- Hybrid: combination of discrete-event and continuous dynamics
- Many modeling frameworks
  - trade-off: modeling power vs. decision power
  - application specific
- In the next lectures: properties, analysis, control, identification, fault diagnosis, and applications

Selected references

- P.J. Antsaklis and A. Nerode, eds., "Special issue on hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, Apr. 1998.
- A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.
- M.S. Branicky, *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1995.
- R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005.
- A.S. Morse, C.C. Pantelides, S. Sastry, and J.M. Schumacher, eds., "Special issue on hybrid systems," *Automatica*, vol. 35, no. 3, Mar. 1999.
- A.J. van der Schaft and J.M. Schumacher, *An Introduction to Hybrid Dynamical Systems*, vol. 251 of *Lecture Notes in Control and Information Sciences*. London: Springer-Verlag, 2000.