EECE 571M/491M, Spring 2008
Lecture 3

Modeling Hybrid Systems

Meeko Oishi, Ph.D.

Electrical and Computer Engineering

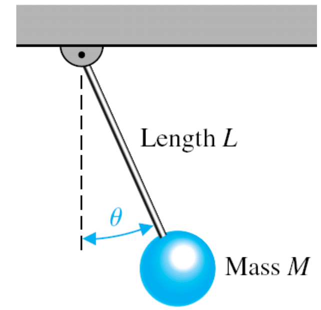
University of British Columbia, BC

http://courses.ece.ubc.ca/491m

moishi@ece.ubc.ca

Tomlin LN 2,3

Review: Continuous systems



$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} \dot{\theta} \\ -\frac{g}{L}\sin\theta \end{bmatrix}$$

- Differential equations with output
  $$\dot{x} = f(x)$$
  $$y = h(x)$$
  - State *x(t)*
  - Input *u(t)*
  - Output *y(t)*
- **Solution** (or **trajectory**) of this ODE is *x(t)*
- Solution requires knowledge of the initial condition $x(0) = x_0$

Review: continuous systems

- What can go wrong with solving ODEs?
  - *No solutions*
    - Exercise: Show that *dx/dt = -sign(x), x(0)* = 0 does NOT have a solution for t >= 0.
  - *Multiple solutions*
    - Exercise: Show that for $dx/dt = x^{1/3}$, x(0) = 0, both functions

$$x(t) = \tfrac{2}{3}t^{3/2}$$
$$x(t) = 0$$

are solutions to the differential equation.

- Theorem (Existence and uniqueness of solutions):
  If *f(x)* is **Lipschitz continuous**, then the differential equation dx/dt = f(x), *x*(0) = x_0 has a unique solution x(t) for t >0.
- This ensures **smoothness** by bounding the slope of *f*

Review: Continuous systems

Lipschitz continuity

- *f(x)* is **Lipschitz continuous** if there exists a constant *K* > 0 such that for all *x, y*

|| f(x) - f(y) || < K || x - y ||

(where || . || indicates the **vector norm**)

- *K* is the **Lipschitz constant**
- Note that
  - If *f* is Lipschitz continuous, it is also continuous.
  - A Lipschitz continuous function is not necessarily differentiable.
  - All differentiable functions with bounded derivatives are Lipschitz

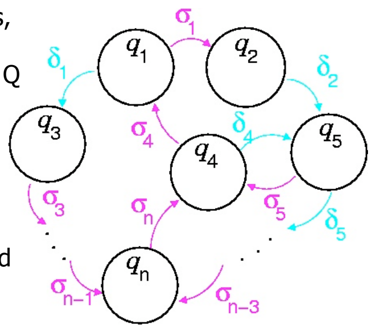
Review: Discrete systems

(Deterministic) finite automata

- Set of discrete states (or modes) **Q**
- Set of discrete inputs (or events, transitions) **Σ**
- Transition function **R**: Q x Σ -> Q
- Initial set of states $\mathbf{Q_0}$

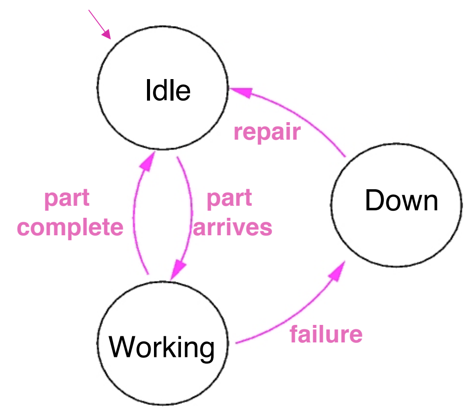
- Solutions are mode strings and event strings
- Note that R(q,σ) = {Ø} if the function is not otherwise defined for that pair
- Easily encoded in tables

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



$$q(k+1) = R(q(k), \sigma(k))$$

Review: Discrete Systems



Manufacturing machine

- Modes
  $Q = \{\text{Idle, Working, Down}\}$
- Events
  Σ = { part complete, part arrives, repair, failure }
- Transition function
  R(Idle, part arrives) = Working
  R(Working, part complete) = Idle
  R(Working, failure) = Down
  R(Down, repair) = Idle
  R(Down, failure) = {Ø}, etc.
- Initial state Q_0 = Idle

Review: Contin. Sys. & DES

Element	Continuous	Discrete
Dynamics	$\dot{x} = f(x,u)$	$R(q_k, \sigma) = q_{k+1}$
State	$x \in R^n$	$q \in Q$
Control	$u \in R^m$	$\sigma \in \Sigma$
Output	$y = h(x,u)$	$Q_M \subseteq Q$
Initial Condition	$x(0) = x_0 \in R^n$	$q[0] \in Q_0 \subseteq Q$
Solution	x(t) for u(t), x(0) known	q[k] for σ[k], q(0) known

Today's lecture

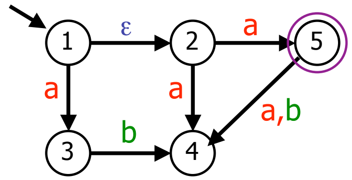
- Discrete event systems
  - Review
  - Discussion and examples of nondeterministic DES
- Hybrid systems
  - Differential or difference equations within each mode
  - New elements: domains, guards, resets, etc.
  - Common forms
  - Modeling variations
- Examples
  - Bouncing ball
  - Water tank

Discrete systems

Nondeterministic finite automata

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

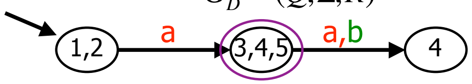
- Set of discrete states (modes) **Q**
- Set of discrete inputs (events, transitions) **Σ**
- Transition relation
  R: Q x Σ -> 2^Q
- Initial set of states $\mathbf{Q_0}$
- Marked, final, or accept states $\mathbf{Q_M}$



$q[k+1] \in R(q[k], \sigma)$

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

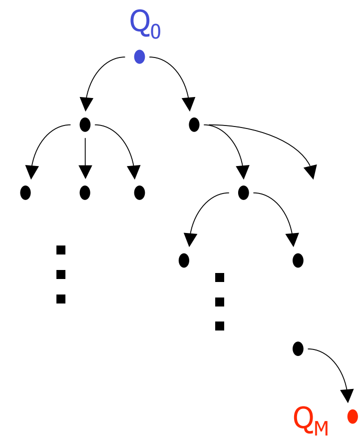
$q[k+1] = R(q[k], \sigma)$



Discrete systems

Nondeterministic finite automata

- Often event strings which lead to Q_M ("marked", "final", or "accept" states) are important (safety, liveness, etc.)
- They are distinguished from strings which do not reach Q_M
- Interpretation: Keep track of multiple "paths" in the automata simultaneously to determine which discrete inputs (event strings) produce the marked states.

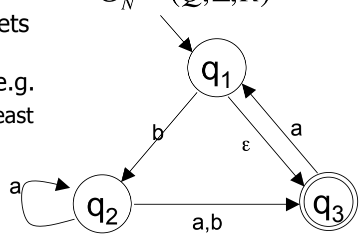


Discrete systems

Nondeterministic finite automata

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

- Solutions are strings of mode sets and strings of events that are "accepted" by the automaton, e.g.
  - $R(Q_i, s) = q_k$ if there exists at least one mode in Q_i (a subset of Q) that can transition into q_k
- Every nondeterministic finite automata G_N has an equivalent deterministic finite automata G_D (with modes in 2^Q) which accepts the same language $L(G_N)=L(G_D)$
- G_N may have **empty events** ε which transition the mode silently



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

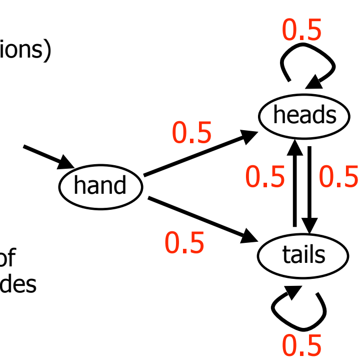
?

Discrete systems

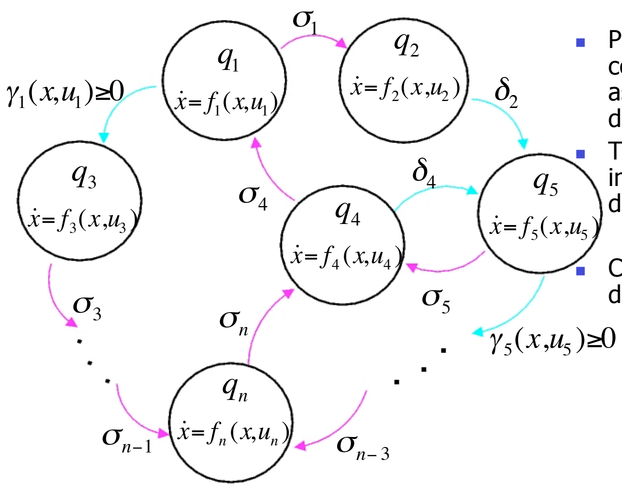
Probabilistic finite automata

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

- Set of discrete states (or **modes**) Q
- Set of **events** (or transitions) Σ which have associated probabilities
- Transition function R
- Initial set of states Q_0
- Solutions are **random processes,** in the form of strings of events and modes
- Markov chains



Hybrid Systems



- Potentially different continuous dynamics associated with each discrete mode
- Transitions can be input-driven or state-driven
- Continuous and discrete inputs

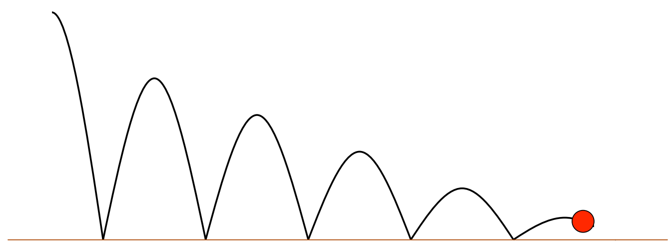
Hybrid systems

The hybrid system H is a collection with the following entities:

- Discrete state $q \in Q$
- Continuous state $x \in R^n$
- Discrete inputs $\sigma \in \Sigma$
- Continuous inputs $u \in U \subseteq R^m$
- Continuous dynamics $\dot{x} = f(q, x, u)$
- Discrete dynamics $R: Q \times R^n \times \Sigma \times R^m \rightarrow 2^{Q \times X}$
- Initial state $\text{Init} \subseteq Q \times R^n$
- Domain (combinations of states and inputs for which continuous evolution is allowed) $\text{Dom} \subseteq Q \times R^n \times \Sigma \times U$

- Interpretation: R **enables** transitions, Dom **forces** them

Example: Bouncing ball

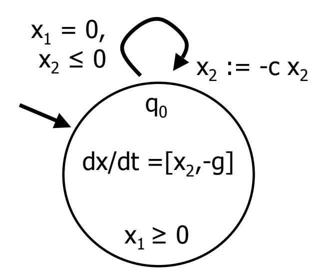


- While the ball is in the air
  m d^2x/dt^2 = -mg
- Upon impact with the ground
  $v_{new} = - c\ v_{old}, \quad 0 < c < 1$

Example: Bouncing ball

The hybrid system H_{ball} is a collection with the following entities:

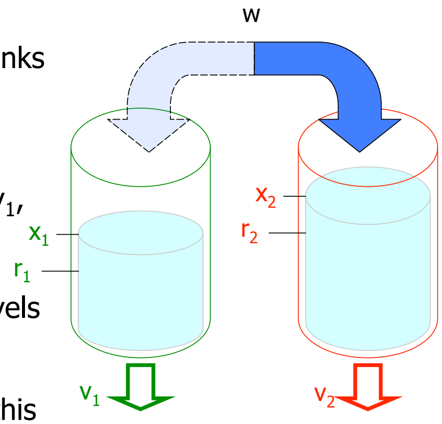
- Discrete states **Q**
- Continuous state **x**
- Continuous dynamics **dx/dt = f(q,x)**
- Discrete dynamics **R(q,x)**
- Initial state **Init**
- Domain **Dom**



Guard condition **Guard(q,q')**
Reset map **Reset(q,x)**

Example: Water tanks

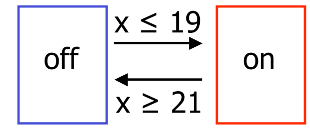
- One hose fills two tanks at constant rate *w*
- Hose switches instantaneously
- Tanks leak at rates v_1, v_2, respectively
- Goal: Keep water levels x_1, x_2 above r_1, r_2, respectively
- Question: How can this be modeled as a hybrid system?



dx/dt = w-v

Example: Thermostat

- Heater in a room is either on or off
- Temperature rises or falls depending on the heater's status
- Inaccuracies in temperature measurement
- Goal: Keep temperature near 20 C while avoiding **chattering**
- Question: How can this be modeled as a hybrid system?



dx/dt = -ax   dx/dt= a(30-x)

$x \geq 18$ $x \leq 22$

Hybrid systems

- Classified according to the form of f(x,u) and the shapes of the domains, guards, and control bounds

- Timed automata
- Rectangular hybrid automata
- Linear hybrid automata
- Piecewise affine automata
- Multi-affine hybrid automata
- Nonlinear hybrid automata
- Switched systems

Hybrid systems

Common model variations

- Terminology
  - Domain, invariant, guard, reset map, etc.
  - Description of events (controlled, disturbance, state-based)
- Autonomous hybrid systems
  - No discrete inputs and no continuous inputs (e.g., bouncing ball)
- Disturbance inputs
  - Discrete disturbances (unexpected failures)
  - Continuous disturbances (wind)
- Switching schemes
  - No discrete automaton specified (e.g., arbitrary switching)

Summary

- Review of discrete event systems:
  - Deterministic
  - Nondeterministic
  - Probabilistic

- Hybrid systems defined by states, inputs, functions/relations, initial conditions, domains
- Modifications to standard continuous and discrete elements
- Graphical representations of hybrid systems
- Example: Bouncing ball
- Example: Water tank