Hybrid systems: an introduction

Billy

Universidad de Buenos Aires

billy.mosse@gmail.com
Robé el template de internet

January 15, 2017

Overview

A little background

Before:

- Rapid progress in hardware and software $\Rightarrow$ ambitious projects $\Rightarrow$ costly ad-hoc integration and validation of systems. (Bottleneck)
- Centralized control still worked over a distributed system: each new increment was slowly made, after extensive (but not exhaustive) tests
- Ran with considerable "slack" (and yet still failed)

This centralized control, ad-hoc, slack-y approach can't meet today's standards.

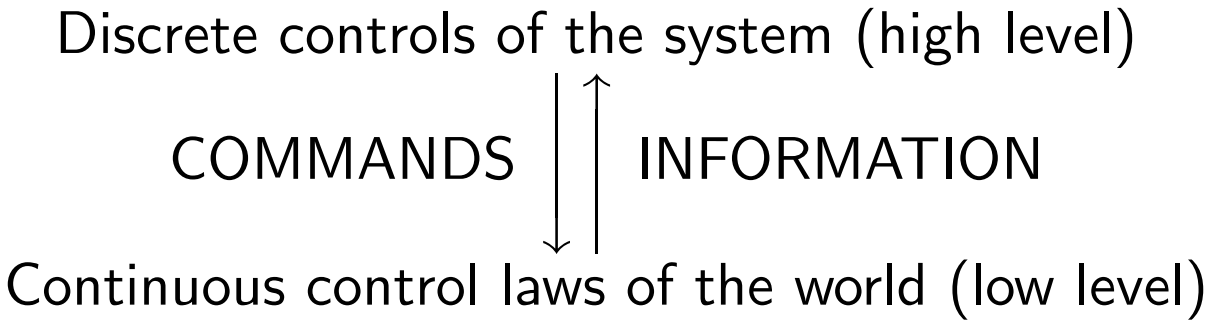
We need a change of paradigm for distributed control that

- avoids the high communication and computation costs of a central control
- but permits central authority

So...a hierarchical structure, may be?

"Studies [citation needed] indicate that, if there is no change to the structure of Air Ttraffic Control, then by the year 2015 there could be a major accident every 7 to 10 days."

Hierarchical structure



Literature on hybrid systems

One idea: extend techniques for finite state automata to include systems with simple continuous dynamics.

How? Model checking and/or deductive theorem proving.
Emphasis: computability, decidability. Is "Does the problem satisfy the specification" decidable?

Models and decidability results have been obtained for timed automata [1], linear hybrid automata [2], and hybrid input/output automata [3].

Decidability results for linear hybrid automata are fairly narrow. For all but the simplest continuous linear dynamics (two-dimensional rect- angular differential inclusions), reachability properties are semi-decidable at best, and in most cases undecidable. See [4], Section 2, for more details.

Literature on hybrid systems

Un/decidability sometimes may be proven by (bi)simulation

- In dynamical systems: with topological equivalence and homomorphism
- In hybrid systems:
  - Not easy
  - Simulation is an ad hoc definition that must include reachability: if system A simulates system B then it provides a reduction from reachability problem for A to the one for B

In the end, it's another reduction to Halt. (See [5] . Also, [6]. Interestingly, they use some tools from Topology and Geometry.)

Literature on hybrid systems

Another idea: extend the standard modeling, reachability and stability analyses, and controller design techniques in continuous state space and continuous time dynamical systems and control to capture the interaction between the continuous and discrete dynamics.

Goal: extend standard modeling, reachability* and stability analyses, and controller design techniques.

* Really hard for systems whose dynamics are nonlinear or are of order greater than one. Only recently [<2007], some attempts to directly approach this problem have been reported in the literature. TODO: check if it is still hard

Tools extended: stability theory [17], optimal control [17, 53, 67], and control of discrete event systems [49, 38]

Stability theory addresses the stability of solutions of differential equations and of trajectories of dynamical systems under small perturbations of initial conditions.

See [1] or maybe [2] for the extension to hybrid systems.

Optimal control

x $\leftarrow$ state

u $\leftarrow$ controllable parameters

minimize $J = \psi[x(T)] + \int_0^T \ell(u, x(t))dt$

One solution (for the continuous problem): 1) discretization of the problem 2) Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (See Bellman, 1957, Chap. III.3.) 3) Go backwards. Problems: costly; "curse of dimensionality" 4) http://www.mpi-magdeburg.mpg.de/94992/Optimal-Control-of-Hybrid-Systems

Another solution: gradient descent?

Another solution: perturbate (u) + differentiate, and get necessary conditions (stated in
See (Or let's do together the "a minimizing curve in the plane is a straight line)

How do we solve this in the hybrid case?

- Get a lower bound for the minimum. See [8] (uses a discretization of Bellman's inequality to be able to do linear programming)
- Pressing a discrete switch every time we'd get an at-the-moment improvement gives us an upper bound for the minimum.

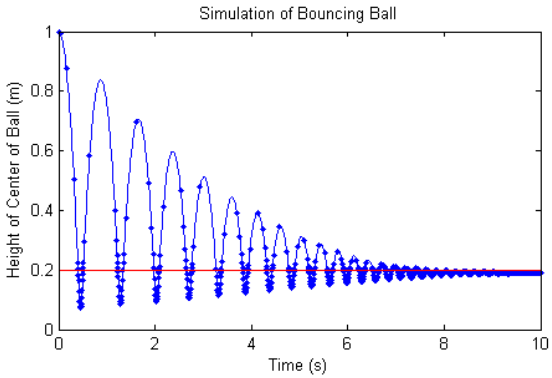
What's interesting

- Continuity with respect to initial conditions (for simulations)
  - Using topology (homotopy theory) tools (Sokorod topology for paths) - not practical (See [9])
  - [10] is a more practical (and concrete?) but still limited method
- Continuity with respect to initial condition **parameters**
- Well-posedness (existence and uniqueness of solutions)
- Zeno executions

Zeno executions

An execution is called Zeno, if it contains an infinite number of transitions in a finite amount of time (Achilles vs Tortoise [11]). An automaton is called Zeno if it accepts a Zeno execution.

Other examples: bouncing ball; 2 water tanks w/alternating faucet.



Problems:

- Semantical: how is it defined beyond the Zeno time?
- Analysis: induction and reachability proofs become suspect
- Controller synthesis: it can cheat by forcing time to converge
- Simulation: it stalls at Zeno time

Resolving the Zeno phenomenon

The only known conditions to characterize the Zeno phenomenon are fairly trivial.

Possible solution: a regularization approach (of the automata), inspired by the method used in differential equations. ([12], 7.4.3)

E.g.

- Water tank automaton: the switch takes ϵ to activate.
- Bouncing ball: each bounce takes ϵ (meanwhile, gravity still applies)

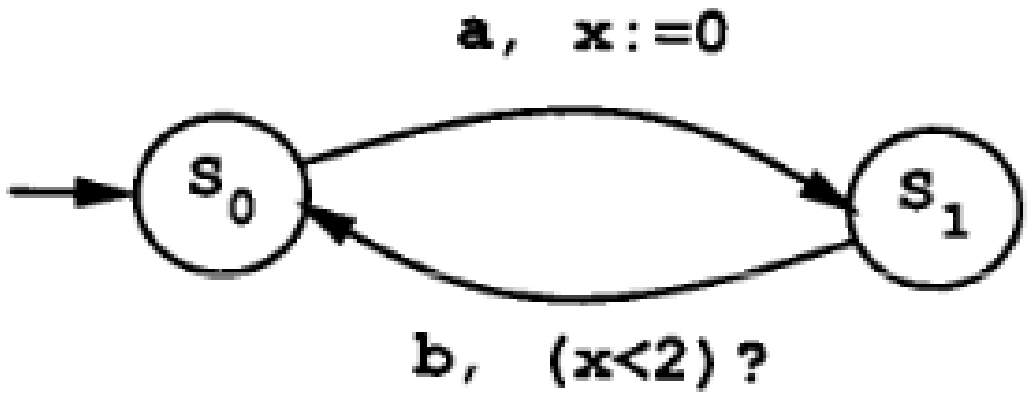
As $\varepsilon \to 0$, then the non-Zeno automata H_ε converges to the original Zeno automata (the states converge, in the Skorohod metric, i.e., wiggling space and time a bit)

See [13] for more.

Some basic descriptions: Timed automata

Timed automata:

- Nice augmentation of ω-automata (Seen in class)
- Words are timed (each letter is presented at a certain time)
- Time satisfies monotonicity and progress
- Clocks that can be reseted (though not in the language)
- You can check time
- Reachability and eventuality properties are decidable



$$\{((ab)^\omega, \tau) \mid \forall i.\ (\tau_{2i} < \tau_{2i-1} + 2)\}.$$

See [13] and [14] for more fun on timed automata.

Some basic descriptions: other automatas

- Linear hybrid automata: models $A\dot{x} \leq b$. Decidability results are fairly narrow. TODO: research how do they model discrete actions.
- Hybrid input/output automata. See Permits:
  - Easy decomposition of description and analysis
  - Showing that one automata implements another one
  - Showing that an automata doesn't contribute to produce Zeno behaviour. This property, under some compatibility conditions, is preserved by composition.

The language

The modelling language must be

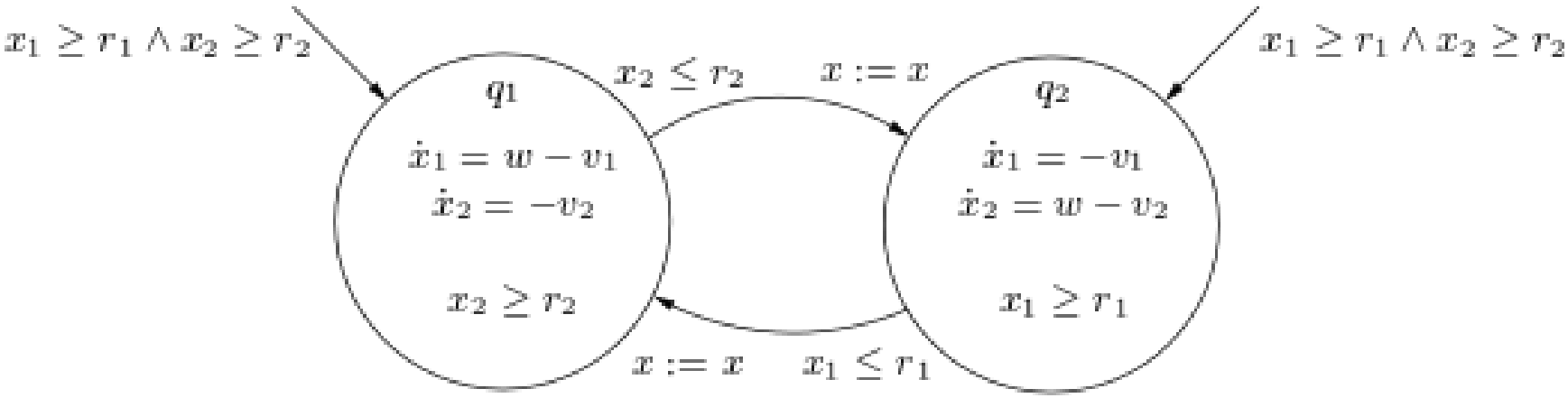
- descriptive: to model how discrete evolution affects and is affected by continuous evolution, to allow non-deterministic models to capture uncertainty
- composable
- abstractable, to refine problems down and compose results up

One of the languages: Hybrid Automata

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

- Q (countable) is a set of discrete variables
- X is a set of continuous variables
- $Init \subset Q \times X$ is a set of initial states
- $f : Q \times X \to TX$ is a vector field: it usually describes the derivative of the continuous variable
- $Inv/Dom : Q \to P(X)$ assigns to each $q \in Q$ an invariant set
- $E \subset Q \times Q$ is a collection of discrete transitions
- $G : E \to P(X)$ assigns to each $e = (q, q') \in E$ a guard.
- $R : E \times X \to P(X)$ assigns to each $e = (q, q') \in E$ and $x \in X$ a reset relation

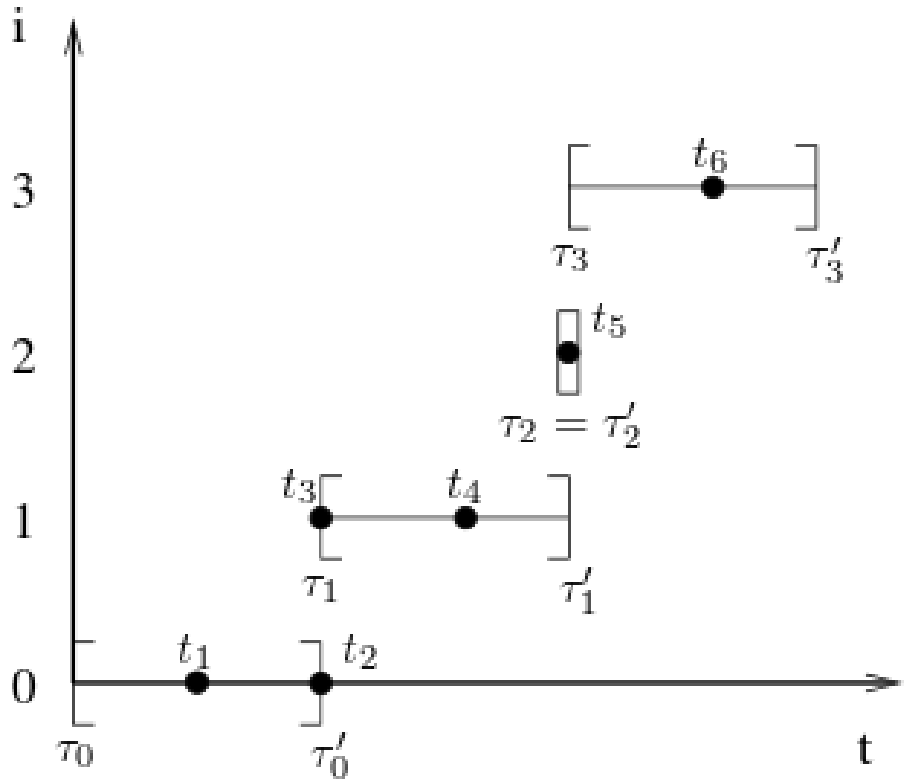
Example: Water Tank System. Two tanks are leaking are at constant rates (v_1, v_2) respectively. Water is added constantly though a hose controlled by an instantaneous switch.
¿Can the water of both tanks (x_1, x_2) be kept above (r_1, r_2)?


$x_1 \geq r_1 \wedge x_2 \geq r_2$
q_1
$\dot{x}_1 = w - v_1$
$\dot{x}_2 = -v_2$
$x_2 \geq r_2$
$x_2 \leq r_2$
$x := x$
$x := x$
$x_1 \leq r_1$
q_2
$\dot{x}_1 = -v_1$
$\dot{x}_2 = w - v_2$
$x_1 \geq r_1$
$x_1 \geq r_1 \wedge x_2 \geq r_2$

Definition: Hybrid time set.

A hybrid time set is a set $\tau = \{I_0, \cdots, I_N\}$ (finite or infinite) of almost-disjoint ordered intervals.

- $I_i = [\tau_i, \tau_i']$ with $\tau_i' = \tau_{i+1}\ \forall\ i < N$
- If $N < \infty$ then I_N might be $[\tau_N, \tau_N')$



Definition: execution

Definition: A hybrid trajectory is a triple (τ, q, x), with the hybrid time set $\tau = \{I_i\}$, and two sequences of functions $q = \{q_i\}, x = \{x_i\}$, with $q_i(.) : I_i \to Q, x_i(.) : I_i \to \mathbb{R}^n$

Definition: an execution $\mathbb{X}$ of a hybrid automaton H is a hybrid trajectory (τ, q, x) satisfying:

- Initial condition: $q(0), x(0) \in Init$
- Discrete evolution: the discrete transitions must be valid and the guards and reset functions must be satisfied
- Continuous evolution: $\frac{dx_i}{dt} = f(q_i(t), x_i(t))$, and between them they must belong to the correct domain. q_i must be constant over I_i

Controllers: a really brief introduction

- We add input variables $v \in V$ (continuous/discrete and controls(U)/disturbances (D)). Controls are used to guide the continuous evolution through f, enable internal transitions through G, force internal transitions through Dom, and determine the state after a transition through R.
- Same with output variables (for each state of the automaton)
- $C : \mathbb{X}^* \to 2^U$ a controller that restricts the control input variables allowed at the final state
- $\mathbb{H}_C$ the set of "closed loop causal executions" (executions with inputs always approved by C)
- C satisfies $(Q \cap X, \Box F)$ if $\Box F(\chi) = True \ \forall \ \chi \in \mathbb{X}$. Problems: Zeno, blocked executions
- We say a controller is memoryless if every time two executions χ_1, χ_2 have the same endpoint, $C(\chi_1) = C(\chi_2)$

([0], Proposition 6.2)

A controller satisfying $(Q \cap X, \Box F)$ exists if and only if a memorless controller satisfying $(Q \cap X, \Box F)$ exists.

Idea of the proof: a drawing. Also, we must strongly use that valid executions and properties are "locally memoryless"
(By contradiction)
1. Assume χ_1, χ_2, reaching (x, q) with $C(\chi_1) \neq C(\chi_2)$.
2. Then, append χ' to χ_2, but with a control rule applied to $\chi_1\chi'$ so that F doesn't hold somewhere. (We can assume we can do it for some pair χ_1, χ_2, because we are supposing a memoryless strategy doesn't exist)
3. F breaks down for $\chi_2\chi'$ at some time t. $\chi_1\chi'$ is a valid execution as executions are memoryless, so the same happens at time t. Absurd! Because $\chi_1\chi'$ was also validated by the controller, by construction, and the controller satisfied $(Q \cap X, \Box F)$

Bibliografía

1. R. Alur and D. Dill. A theory of timed automata. Theoretical Computer Science, 126:183–235, 1994.

2. R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid automaton: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, Hybrid Systems, number 736 in LNCS, pages 209–229. Springer-Verlag, Berlin, 1993.

3. N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In Hybrid Systems III, number 1066 in LNCS, pages 496–510. Springer-Verlag, Berlin, 1996.