Hybrid systems: an introduction

Billy

Universidad de Buenos Aires

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

January 10, 2017

Overview

Tipos de juegos que conozco: combinatoriales

Rapid progress in hardware and software $\rightarrow$ ambitious projects $\rightarrow$ costly ad-hoc integration and validation of systems.
Slowly incremental distributed systems "sin usar todo al mango"
more people $\rightarrow$ we need new paradigm: we can't afford a central control, but it would be nice to have a central authority so we can take smart decisions with the information from all the system.
hierarchic (esto no se escribe asi) system, more or less autonomous functions, and harmony.
"Studies [citation needed] indicate that, if there is no change to the structure of ATC, then by the year 2015 there could be a major accident every 7 to 10 days."

Hierarchical structure

Discrete controls of the system (high level) INFORMATION GOES UP
COMMANDS GO DOWN
Continuous control laws of the world (low level)

Problems we could resolve

- Automated highway systems
- Air traffic control
- Groups of anmanned aerial vehicles
- Underwater autonomous vehicles
- Mobile offshore platforms

Literature on hybrid systems

1) 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. "Does the problem satisfy the specification" is decidable?
Models and decidability results have been obtained for timed automata [4], linear hybrid automata [3], and hybrid input/output automata [59].
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 https://www.irif.fr/ asarin/papers/incl.pd, Section 2

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

- Dynamical systems: topological equivalence and homomorphism
- Hybrid systems:
  - Not easy
  - 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 http://www.sciencedirect.com/science/article/pii/030439759400228B . Also, http://www.sciencedirect.com/science/article/pii/S0890540112000028) Interestingly, it uses some tools from Topology and Geometry.)

Literature on hybrid systems

2) 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.
Tools extended: stability theory [17], optimal control [17, 53, 67], and control of discrete event systems [49, 38]
ne area in which results have been hard to come by is the efficient computation of reachable sets for hybrid 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

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

Optimal control

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

One solution: 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"

Another solution: gradient descent?

Another solution: perturbate (u) + differentiate, and get 4 necessary conditions (stated in
See (Or let's do together the "a minimizing curve in the plane is a straight line)

What's interesting

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

Zeno executions

An execution is called Zeno, if it contains an infinite number of transitions in a finite amount of time. An automaton is called Zeno if it accepts a Zeno execution.
Example: bouncing ball
Problems:

- Semantical: how it is 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.
A regularization approach (of the automata), inspired by the method used in differential equations.
E.g.

- ater 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 ($\phi_\varepsilon : Q_\varepsilon \times X_\varepsilon \to Q \times X$ converges in the Skorohod metric)
See

Some basic descriptions: Timed automata

See
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 (in the automata, not...the language)
- Reachability and eventuality properties are decidable

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 there isn't Zeno behaviour under

The language

The modelling language must be

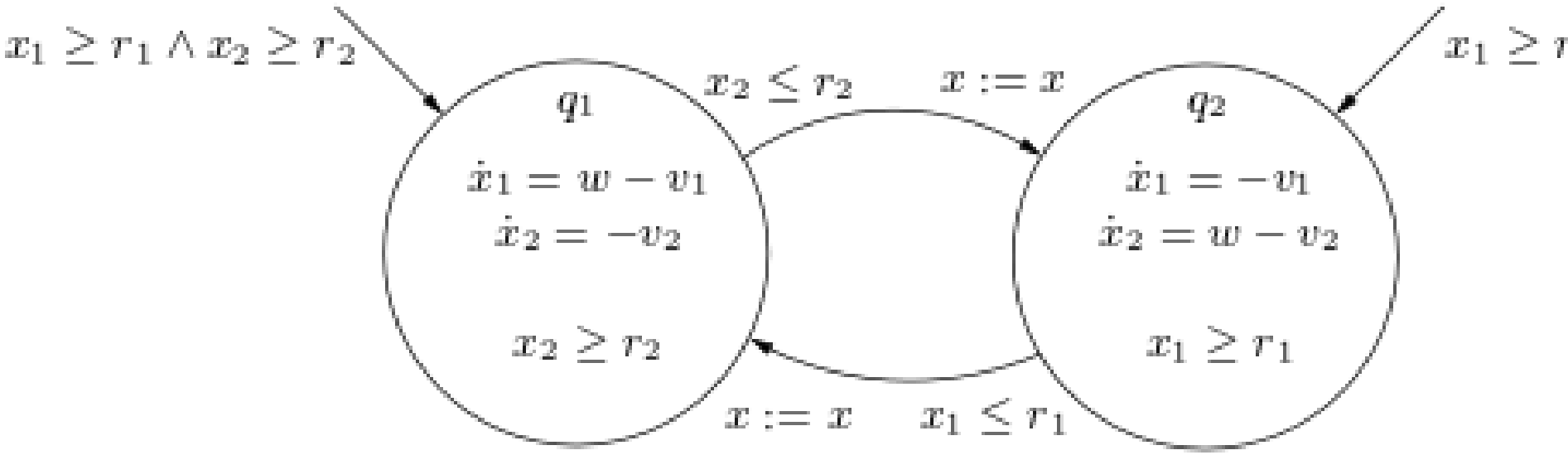
- descriptive: to model how discrete evolution affects and is affected by continuous evolution, to allow non-deterministic models to capture uncertainty
- composable
- abstractable, to refine problems down and compose results up

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

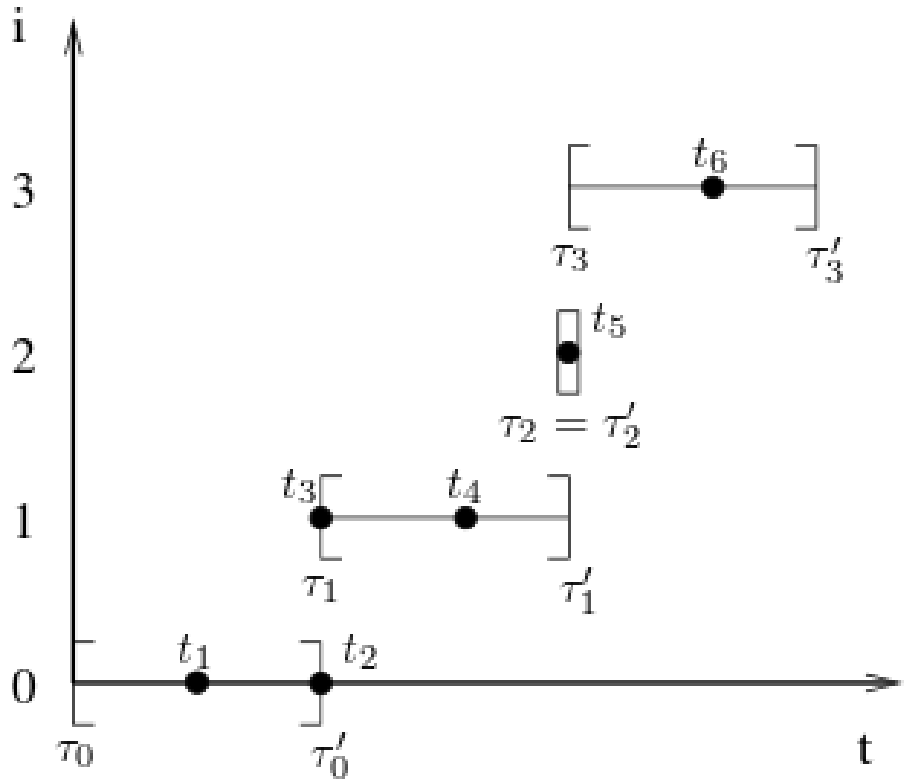
Example: Water Tank System. Two tanks are leaking are at constant rates (v_1, v_2) respectively. Water is added constantly though a hose controlled by an instantaneous switch.
¿Can the water of both tanks (x_1, x_2) be kept above (r_1, r_2)?


$x_1 \geq r_1 \wedge x_2 \geq r_2$
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
$\dot{x}_1 = -v_1$
$\dot{x}_2 = w - v_2$
$x_1 \geq r_1$
$x_1 \geq r$

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 $q_i(.), x_i(.) : I_i \to \mathbb{R}^n$

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

- Initial condition: $q(0), x(0) \in \mathit{Init}$
- Discrete evolution: the discrete transitions must be valid and the guards must be satisfied
- Continuous evolution: the continuous variables must be reseted after each transition, and between them they must belong to the 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))
- $C : \mathbb{H} \to 2^U$ 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 $\Box F(\chi) = \mathit{True}\ \forall\ \chi \in \mathbb{H}$. Problems: Zeno, blocked executions

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 $\chi_2\chi'$ so that F doesn't hold somewhere. (We can assume we can do it, because we are supposing a memoryless strategy doesn't exist) 3. $\chi_2\chi'$ breaks down at some time t. $\chi_1\chi'$ is a valid executions as executions are memoryless, and also breaks down 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

Game Theory, Ferguson