Coalgebra meets Hybrid Systems

Renato Neves
joint work with Luis Barbosa, Sergey Goncharov, and José Proença

Universidade do Minho

HASLab
HIGH-ASSURANCE
SOFTWARE LABORATORY

This Talk

Some examples of how Coalgebra helps to provide

- syntax
- semantics
- and notions of equivalence

for hybrid systems

Table of Contents

Introduction to Hybrid Systems

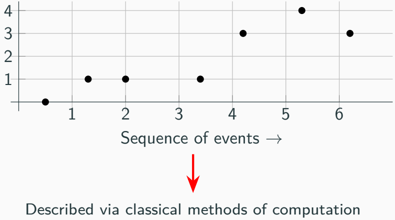
Overview

Hybrid Automata as Coalgebras

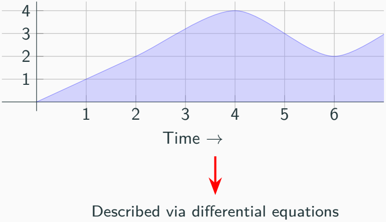
Hybrid While-Language

Conclusions and Future Work

The Essence of Hybrid Systems



+

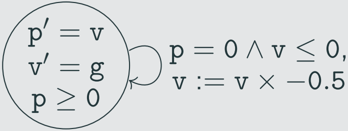


Often found in the form of

- digital devices that closely interact with physical processes
- ‘impact-based’ physical systems

Main Formalisms for Hybrid Systems

Hybrid Automata



Hybrid While-Language

$$\texttt{while}\ \mathrm{true}\ \texttt{do}\ \{p' = v, v' = g\ \texttt{until}\ p = 0 \wedge v \leq 0$$
$$v := v \times -0.5\}$$

Table of Contents

Introduction to Hybrid Systems

Overview

Hybrid Automata as Coalgebras

Hybrid While-Language

Conclusions and Future Work

Talk's Overview

We will focus on the two previous formalisms

- first hybrid automata;
- and then the hybrid while-language

Hybrid Automata and its Variants

The notion of a hybrid automaton has several variants

To be formally detailed later on

- deterministic
- non-deterministic
- probabilistic
- reactive
- weighted
- ...

Unfortunately: no uniform theory of hybrid automata

What can Coalgebra do for Hybrid Automata?

Coalgebra can help solve the aforementioned issue

It provides a uniform theory of hybrid automata, which includes a

- notion of bisimulation
- notion of observational behaviour
- and a regular-expression-like language

Semantics for a Hybrid While-Language

Suitable semantics for hybrid iteration is difficult to establish

Previous work crucially relies on nondeterminism and gives rise to problematic equations, *e.g.*

$$\texttt{while}\,\mathtt{true}\,\texttt{do}\,\{\,\mathtt{p}\,\} = 0$$

Alternative (deterministic) semantics via final coalgebra + weak bisimilarity. It revolves around two monads for hybrid computation

$$\hat{\mathrm{H}} \xrightarrow{\text{intensional to extensional}} \mathrm{H}$$

Abstracts away intermediate computational steps

Table of Contents

Introduction to Hybrid Systems

Overview

Hybrid Automata as Coalgebras

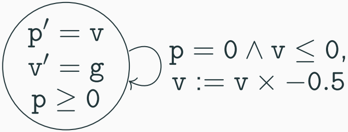
Hybrid While-Language

Conclusions and Future Work

Hybrid Automata – the Basics

They extend non-deterministic finite automata with

- differential equations (for describing continuous dynamics)
- location invariants (for restricting the latter)
- assignments (for describing discrete dynamics)
- guards (for restricting the latter)



Hybrid Automata – Formally

A hybrid automaton is a tuple $(L, E, X, dyn, inv, asg, grd)$ where

- L is a finite set of locations, E is a transition relation $E \subseteq L \times L$, and X is a finite set of real-valued variables
- dyn is a function that associates to each location a system of differential equations over X
- inv is a function that associates to each location its invariant (a predicate over the variables in X)
- asg is a function that given an edge returns an assignment command over X. The function grd associates each edge with a guard (a predicate over the variables in X)

A Surprisingly Useful Remark

Hybrid automata are nothing more than classical, non-deterministic automata with decorated states and edges, i.e.

$$L \to P_\omega(L \times \mathrm{Asg} \times \mathrm{Grd}) \times \mathrm{DifEq} \times \mathrm{Inv}$$

This immediately provides,

- a uniform notion of hybrid automata,
- a uniform regular-expression-like language

More details in [Neves and Barbosa, 2017]

A Zoo of Hybrid Automata

$$M \to F(M \times \mathrm{Asg} \times \mathrm{Grd}) \times \mathrm{DifEq} \times \mathrm{Inv}$$

Functor	Type
Id	Deterministic
P_ω	Classical
D_ω	Markov
$P_\omega D_\omega$	Probabilistic
W_ω	Weighted

Uniform Semantics for Hybrid Automata

Many variants of hybrid automata come equipped with their own semantics

We can encode these uniformly and in functorial form

$$[\![-]\!] : \mathsf{HybAt}(F) \longrightarrow \text{Category of coalgebras}$$

Let us see how ...

Uniform Semantics for Hybrid Automata – Preliminaries

We adopt the following three assumptions (the last two used merely to simplify the presentation)

Unique Solutions

The function dyn only outputs systems of differential equations with exactly one solution. This induces a function

$$\mathrm{flow} : L \times \mathbb{R}^n \times [0, \infty) \to \mathbb{R}^n$$

Urgent Transitions

As soon as an edge is enabled the current location must switch

Non-restrictive Invariants

The invariants of all locations are `true`

Uniform Semantics for Hybrid Automata – Rationale

$$L \times \mathbb{R}^n \to F(L \times \mathrm{Asg} \times \mathrm{Grd}) \times \mathrm{DifEq}$$

$$\Rightarrow L \times \mathbb{R}^n \to F(L \times \mathrm{Asg} \times \mathrm{Grd}) \times (\mathbb{R}^n)^{[0,\infty)}$$ → Space of continuous trajectories

$$\Rightarrow L \times \mathbb{R}^n \to F\left(L \times \mathrm{Asg} \times \mathrm{Grd} \times (\mathbb{R}^n)^{[0,\infty)}\right)$$

$$\Rightarrow L \times \mathbb{R}^n \to F\left(L \times \mathrm{Asg} \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times \mathbb{R}^n + (\mathbb{R}^n)^{[0,\infty)}\right)$$

$$\Rightarrow L \times \mathbb{R}^n \to F\left(L \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times \mathbb{R}^n + (\mathbb{R}^n)^{[0,\infty)}\right)$$

$$\simeq L \times \mathbb{R}^n \to F\left(L \times \mathbb{R}^n \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} + (\mathbb{R}^n)^{[0,\infty)}\right)$$

We obtain a coalgebra for $F\left(- \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} + (\mathbb{R}^n)^{[0,\infty)}\right)$

Bisimulation

Many variants of hybrid automata come equipped with their own notion of bisimulation

The notion is placed at the semantic level

The coalgebraic notion of bisimulation does not coincide with the notion of bisimulation for hybrid automata

However . . .

Coalgebraic Φ-Bisimulation

The Starting Point

Each equivalence relation $\Phi : (L \times \mathbb{R}^n) \times (L \times \mathbb{R}^n)$ induces a quotient map $q : L \times \mathbb{R}^n \twoheadrightarrow Q$

Denote $\coprod_{d \in [0,\infty)} X^{[0,d)}$ by $Tr(X)$

And then ...

$$\begin{tikzcd}
\mathrm{HybAt}(F) \arrow[d, "\text{semantics}"'] & & \\
\mathrm{CoAlg}(F(- \times Tr(\mathbb{R}^n) + (\mathbb{R}^n)^{[0,\infty)})) \arrow[rr, dotted, bend left, "\text{colour}"] & & \mathrm{CoAlg}(F(- \times Tr(\mathbb{R}^n \times L) + (\mathbb{R}^n \times L)^{[0,\infty)})) \arrow[ll, "\text{forget}"] \arrow[d, "\text{quotient}"] \\
& & \mathrm{CoAlg}(F(- \times Tr(Q) + Q^{[0,\infty)}))
\end{tikzcd}$$

Coalgebraic Φ-Bisimulation

Coalgebraic Φ-bisimilarity covers the classic notions of bisimilarity for,

1. deterministic,
2. non-deterministic,
3. and probabilistic hybrid automata.

Observable Behaviour

The generalised semantics yields coalgebras for

$$G \simeq F\left(- \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} + (\mathbb{R}^n)^{[0,\infty)}\right)$$

If F is bounded we obtain a notion of observable behaviour given by the corresponding universal map to the final coalgebra

$$\begin{tikzcd} X \arrow[r, "\mathrm{beh}_{[\![ha]\!]}"] \arrow[d, "{[\![ha]\!]}"'] & \nu\gamma.\ G\gamma \arrow[d, "\simeq"] \\ GX \arrow[r] & G(\nu\gamma.\ G\gamma) \end{tikzcd}$$

Observable Behaviour for $F := \mathrm{Id}$

The carrier of the final coalgebra is given by

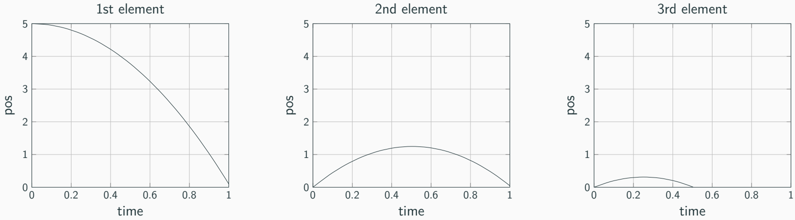
$$\nu\gamma.\left(\gamma \times \coprod_{d\in[0,\infty)} (\mathbb{R}^n)^{[0,d)} + (\mathbb{R}^n)^{[0,\infty)}\right)$$
$$\simeq \left(\coprod_{d\in[0,\infty)} (\mathbb{R}^n)^{[0,d)}\right)^* \times \underset{\substack{\downarrow \\ \text{The case in which a guard never activates}}}{(\mathbb{R}^n)^{[0,\infty)}} + \left(\coprod_{d\in[0,\infty)} (\mathbb{R}^n)^{[0,d)}\right)^\omega$$

Revisiting the Bouncing Ball

Via the semantics functor $[\![-]\!]$ we obtain the following picture

$$\mathtt{bb} = \left\{ \begin{array}{c} \mathtt{p}' = \mathtt{v} \\ \mathtt{v}' = \mathtt{g} \\ \mathtt{p} \geq 0 \end{array} \quad \begin{array}{c} \mathtt{p} = 0 \wedge \mathtt{v} \leq 0, \\ \mathtt{v} := \mathtt{v} \times -0.5 \end{array} \right\} \qquad \mathrm{beh}_{[\![\mathtt{bb}]\!]}(*, (5, 0)) = \ \ldots$$

Position and velocity



Current Work

Currently working on a coalgebraic notion of approximate bisimilarity for hybrid automata

Table of Contents

Introduction to Hybrid Systems

Overview

Hybrid Automata as Coalgebras

Hybrid While-Language

Conclusions and Future Work

Syntax

Fix a stock of variables $X = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$. Then we have,

Linear Terms

$\mathtt{LTerm}(X) \ni \mathtt{r} \mid \mathtt{r} \cdot \mathtt{x} \mid \mathtt{t} + \mathtt{s}$

real number

Atomic Programs

$\mathtt{At}(X) \ni \mathtt{x} := \mathtt{t} \mid \mathtt{x}_1' = \mathtt{t}_1, \ldots, \mathtt{x}_n' = \mathtt{t}_n \ \mathtt{for}\ \mathtt{t}$

"run" the system of differential equations for t seconds

Hybrid Programs

$\mathtt{Prog}(X) \ni \mathtt{a} \mid \mathtt{p} \,;\, \mathtt{q} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}$

Semantics – Key Aspects

How to interpret a hybrid program $\mathtt{p}$?

$$[\![\mathrm{x}' = 1 \texttt{ for } 1]\!] : \mathbb{R} \longrightarrow (\text{trajectories over } \mathbb{R})$$

i.e. functions from a time-domain into $\mathbb{R}$

How to interpret sequential composition?

The signature of the denotation suggests the use of monads

Semantics – First Approach

Recall our use of $\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)}$ to interpret hybrid automata

Then consider the left adjoint $[\mathsf{Set}, \mathsf{Set}]_\omega \to \mathsf{Mnd}_\omega(\mathsf{Set})$

We use the latter and $\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times (-)$ to obtain the monad

$$\begin{aligned} X \mapsto \; & \mu\gamma.\left(\textstyle\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times \gamma + X\right) \\ & \simeq \left(\textstyle\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)}\right)^* \times X \end{aligned}$$

Kleisli composition amounts to concatenation of lists of trajectories

Semantics – First Approach

Denotations $[\![\mathrm{p}]\!]$ become functions of the type

$$[\![\mathrm{p}]\!] : \mathbb{R}^n \longrightarrow \left(\sum_{d\in[0,\infty)}(\mathbb{R}^n)^{[0,d)}\right)^* \times \mathbb{R}^n$$

Example (with $n = 1$)

$$[\![\mathrm{x}' = 1 \texttt{ for } 1]\!](0) = ([\lambda t \in [0,1).\ 0 + t], 1)$$

$$[\![\mathrm{x}' = 1 \texttt{ for } 1 \ ; \ \mathrm{x}' = 1 \texttt{ for } 1]\!](0)$$

$$= ([\lambda t \in [0,1).\ 0 + t,\ \lambda t \in [0,1).\ 1 + t], 2)$$

$$[\![\texttt{while } \texttt{true} \texttt{ do } \{\mathrm{x}' = 1 \texttt{ for } 1\}]\!] = ?$$

Semantics – Second Approach

Instead of using the least fixpoint we use the greatest

$$X \mapsto \nu\gamma. \left(\textstyle\sum_{d\in[0,\infty)}(\mathbb{R}^n)^{[0,d)} \times \gamma + X\right)$$
$$\simeq \left(\textstyle\sum_{d\in[0,\infty)}(\mathbb{R}^n)^{[0,d)}\right)^* \times X + \left((\textstyle\sum_{d\in[0,\infty)}(\mathbb{R}^n)^{[0,d)}\right)^\omega$$

This is an instance of a universal construction which tells that

- the functor above is also a monad (henceforth denoted by $\hat{\mathrm{H}}$)
- the monad supports a partial iteration operator

$$\frac{f : X \to \hat{\mathrm{H}}(Y + X)}{f^\# : X \to \hat{\mathrm{H}}(Y)}$$

Semantics – Second Approach

$$\frac{f : X \to \hat{\mathrm{H}}(Y + X)}{f^{\#} : X \to \hat{\mathrm{H}}(Y)}$$

$f^{\#}$ iterates over f until the latter outputs a value of type Y; and concatenates all lists of trajectories produced along the way

Example (with $n = 1$)

$$[\![\texttt{while true do } \{\mathrm{x}' = 1 \texttt{ for } 1\}]\!](0)$$
$$= [\lambda t \in [0,1).\ 0 + t,\ \lambda t \in [0,1).\ 1 + t, \lambda t \in [0,1).\ 2 + t, \dots]$$

Semantics – Third Approach

The proposed semantics is intensional *e.g.*

$$(\mathrm{x}' = 1 \ \texttt{for}\ 1) \ ; (\mathrm{x}' = 1 \ \texttt{for}\ 1) \neq (\mathrm{x}' = 1 \ \texttt{for}\ 2)$$

We should abstract away from invisible intermediate steps, similarly to the case of weak bisimulation

This amounts to 'coherently' turning a sequence of trajectories into a single trajectory

From a Sequence of Trajectories into a Single Trajectory

Concatenation of Trajectories

$$(\lambda t \in [0, d_1).\ f_1(t)) +\!\!+ (\lambda t \in [0, d_2).\ f_2(t))$$
$$= \lambda t \in [0, d_1 + d_2).\ \text{if } t < d_1 \text{ then } f_1(t) \text{ else } f_2(t - d_1)$$

Infinite Concatenation of Trajectories

$f_1 +\!\!+ f_2 +\!\!+ \cdots = \lambda t \in [0, \sum_{i \in \mathbb{N}} d_i).\ f_j(t - \sum_{i<j} d_i)$ where $j \geq 1$ is the smallest integer $s.t.\ t < \sum_{i \leq j} d_i$

A Retraction Appears

The previous operation induces a retraction

$$\hat{\mathrm{H}} \underset{\nu}{\overset{\rho}{\rightleftarrows}} \Big(X \mapsto \textstyle\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times X + \sum_{d \in [0,\infty]} (\mathbb{R}^n)^{[0,d)}\Big)$$

where ρ resorts to concatenation of trajectories and ν is defined as

$$\mathrm{inl}(f, x) \mapsto \mathrm{inl}([f], x)$$
$$\mathrm{inr}(f) \mapsto \mathrm{inr}[f_{[0,1)}, f_{[1,2)}, \ldots] \text{ if duration of } f \text{ equals } \infty$$
$$\mathrm{inr}(f) \mapsto \mathrm{inr}[f, !, !, \ldots] \text{ otherwise}$$

Let us denote the functor on the right-hand side by H

An Extensional Hybrid Monad Appears

$$\begin{tikzcd} \hat{\mathrm{H}} \arrow[r, two heads, bend left, "\rho \text{ (to extensional)}"] & \mathrm{H} \arrow[l, hook, bend left, "\nu"] \end{tikzcd}$$

H inherits from the monad $\hat{\mathrm{H}}$ (through ν and ρ)

- Kleisli composition
- an iteration operator

$$\frac{f : X \to \mathrm{H}(Y + X)}{f^{\dagger} : X \to \mathrm{H}(Y)}$$

Interpretation via H

Interpretation via H provides the desired aforementioned equality

$$(\mathrm{x}' = 1 \texttt{ for } 1) \mathbin{;} (\mathrm{x}' = 1 \texttt{ for } 1) = (\mathrm{x}' = 1 \texttt{ for } 2)$$

and also other expected ones, such as

$$\texttt{while true do } \{\mathrm{x}' = 1 \texttt{ for } 1\} = \texttt{while true do } \{\mathrm{x}' = 1 \texttt{ for } 2\}$$

Thoughts about this Interpretation of While-Loops

- We did not use domain theory
- Instead we used the concept of final coalgebra to guide us
- Extensional
- Contrasts with previous works in the sense that
  - it is deterministic
  - does not collapse infinite while-loops into a single point of divergence, i.e. we do not necessarily obtain

$$\texttt{while}\,\texttt{true}\,\texttt{do}\,\{\,\mathrm{p}\,\} = 0$$

  - In fact we get a continuum of divergence points

A Taxonomy of While Loops

	Non-progressive	Progressive	Zeno
Divergent	$\texttt{while}\,(\texttt{true})\,\{$ $\;\texttt{x} := \texttt{x} + 1\,\}$	$\texttt{while}\,(\texttt{true})\,\{$ $\;\texttt{x} := \texttt{x} + 1\,;(\texttt{wait}\ \epsilon)\,\}$	$\epsilon := 1$ $\texttt{while}\,(\texttt{true})\,\{$ $\;\texttt{x} := \texttt{x} + 1\ ;\,(\texttt{wait}\ \epsilon)$ $\;\epsilon := \frac{\epsilon}{2}\,\}$
Convergent	$\texttt{x} := 0$ $\texttt{while}\,(\texttt{x} \leq 10)\,\{$ $\;\texttt{x} := \texttt{x} + 1\ \}$	$\texttt{x} := 0$ $\texttt{while}\,(\texttt{x} \leq 10)\,\{$ $\;\texttt{x} := \texttt{x} + 1\,;(\texttt{wait}\ \epsilon)\,\}$	N.A.

Demo of the Semantics in Operational Form

http://arcatools.org/assets/lince.html#fulllince

Table of Contents

Introduction to Hybrid Systems

Overview

Hybrid Automata as Coalgebras

Hybrid While-Language

Conclusions and Future Work

Conclusions and Future Work

- Hybrid systems in an object-oriented setting [Jacobs, 2000]
  - Seen as coalgebras $U \to A \times U^B \times U^{\mathbb{R}_{\geq 0}}$
- Approximate bisimulation coalgebraically
  e.g. [Sprunger et al., 2018, König and Mika-Michalski, 2018]
  - Seems particularly well-suited for systems with continuous state-spaces (such as those used in the semantics of hybrid automata)

Funding

This talk was financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 under the Portugal 2020 Partnership Agreement and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within project IBEX, with reference PTDC/CCI-COM/4280/2021.

REPÚBLICA PORTUGUESA

FCT Fundação para a Ciência e a Tecnologia

References i

Jacobs, B. (2000).
Object-oriented hybrid systems of coalgebras plus monoid actions.
Theoretical Computer Science, 239(1):41 – 95.

König, B. and Mika-Michalski, C. (2018).
(metric) bisimulation games and real-valued modal logics for coalgebras.
In *29th International Conference on Concurrency Theory (CONCUR 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

References ii

Neves, R. and Barbosa, L. S. (2017).
Languages and models for hybrid automata: A coalgebraic perspective.
Theoretical Computer Science.

Sprunger, D., Katsumata, S.-y., Dubut, J., and Hasuo, I. (2018).
Fibrational Bisimulations and Quantitative Reasoning.
In Cîrstea, C., editor, *14th International Workshop on Coalgebraic Methods in Computer Science (CMCS)*, volume LNCS-11202 of *Coalgebraic Methods in Computer Science*, pages 190–213, Thessaloniki, Greece. Springer International Publishing.