Semantics for Hybrid Components

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

Universidade do Minho

HASLab
HIGH-ASSURANCE SOFTWARE LABORATORY

Table of Contents

This Talk

An overview of categorical constructions for interpreting hybrid components via

- Coalgebra
- (Elgot) Monads

Iteration

Hybrid Components

Components

Standalone computational units, typically with an internal state, that interact with environment

Hybrid Components

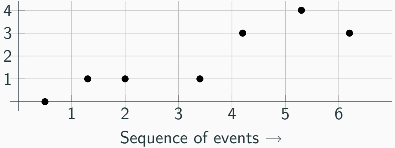
Components

Standalone computational units, typically with an internal state, that interact with environment

Hybrid Components

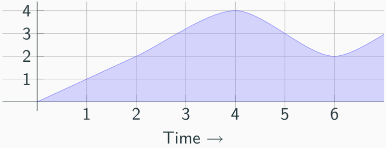
If a component's environment contains physical processes (*e.g.* velocity, time) we qualify the component as hybrid

to emphasise the discrete-continuous interaction



Described via classical methods of computation

+



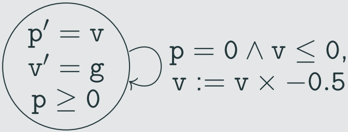
Described via differential equations

Often found in the form of

- cruise-controllers, thermostats, medical devices, ...
- impact-based physical systems

Formalisms for Hybrid Components

Hybrid Automata



Hybrid (Component-based) Programming

$$\texttt{while}\ \mathrm{true}\ \texttt{do}\ \left\{\ \texttt{if b then heatReactor()} \atop \texttt{else coolReactor()}\ \right\}$$

Talk's Overview

We will provide semantics to the two formalisms

- first for hybrid automata
- then for hybrid component-based programming

Hybrid Automata and its Variants

Hybrid automata: standard formalism for modelling hybrid systems

Underlying notion has several variants

To be formally detailed later on

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

Unfortunately: no uniform semantics for hybrid automata

Hybrid Automata and Coalgebra

Coalgebra is a uniform theory of state-based transition systems

We use it to tackle the propounded issue, and obtain uniformly

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

Hybrid Iteration

Suitable semantics for hybrid iteration is difficult to establish

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

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

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

Hybrid Iteration + Internal State

We take a monad able to handle internal states in programming

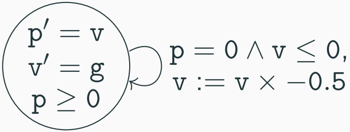
Then combine it with the extensional hybrid monad, and show that the new monad supports iteration. This yields . . .

an interpretation domain for hybrid components in programming

Table of Contents

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

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 $E \subseteq L \times L$, 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 it returns an assignment command over X
- grd is a function that given an edge it returns a guard (*i.e.* a predicate over the variables in X)

Set-based Coalgebra

Definition (Coalgebra)

A function $X \longrightarrow FX$ where $F : \mathrm{Set} \longrightarrow \mathrm{Set}$ is a functor

Set-based Coalgebra

Definition (Coalgebra)

A function $X \to FX$ where $F : \mathrm{Set} \to \mathrm{Set}$ is a functor

Different F, different transition systems

- $X \to \mathrm{Id}X$ (deterministic)
- $X \to \mathrm{P}_\omega X$ (non-deterministic)
- $X \to \mathrm{P}_\omega(A \times X)$ (labelled non-deterministic)
- $X \to D_\omega X$ (probabilistic)
- ...

The Message of Coalgebra

Coalgebra serves as a uniform theory of transition systems, whose level of abstraction is functoriality

It includes,

- notions of (bi)simulation and observational behaviour
- techniques for minimisation
- notions of regular-expression
- ...

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 notion of (bi)simulation and regular-expression

More details in [Neves and Barbosa, 2017]

A Zoo of Hybrid Automata

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

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 a semantics

We can encode these uniformly as a functor

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

Transition systems involving sols. of diff. eqs.

Preliminaries

Three assumptions (last two used merely to simplify 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

No Invariants

The invariants of all locations are `true`

The Semantics

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

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

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

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

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

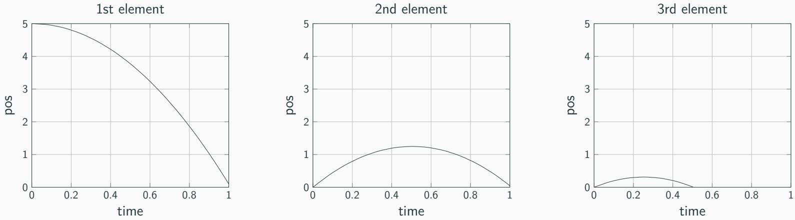
$$\Rightarrow L \times \mathbb{R}^n \to F\left(L \times \mathbb{R}^n \times \coprod_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} + (\mathbb{R}^n)^{[0,\infty)}\right)$$

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

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



Main Theorem

Our generalised semantics covers the established semantics for

- deterministic
- non-deterministic
- and probabilistic hybrid automata

Main Theorem

Our generalised semantics covers the established semantics for

- deterministic
- non-deterministic
- and probabilistic hybrid automata

We have an analogous result for (bi)simulation

Table of Contents

Syntax

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

Linear Terms

$$\mathtt{LTerm}(X) \ni \underbrace{\mathtt{r}}_{\text{real number}} \mid \mathtt{r} \cdot \mathtt{x} \mid \mathtt{t} + \mathtt{s}$$

Atomic Programs

$$\mathtt{At}(X) \ni \mathtt{x} := \mathtt{t} \mid \mathtt{x}_1' = \mathtt{t}_1, \dots, \mathtt{x}_n' = \mathtt{t}_n \ \underbrace{\mathtt{for}\ \mathtt{t}}_{\text{"run" the system of differential equations for } \mathtt{t} \text{ seconds}}$$

Hybrid Programs

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

Key Aspects of the Semantics

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

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

Key Aspects of the Semantics

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

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

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

Signature of the denotation suggests the use of (Elgot) monads

Elgot Monads in a Nutshell

A monad T on Set is called Elgot if it has an iteration operator

$$\frac{f : X \to T(Y + X)}{f^\dagger : X \to TY}$$

that satisfies a certain set of laws

Intuitively, $f^\dagger$ iterates over f until obtaining an output of type Y

Semantics without Iteration

Let $\sum_{d\in[0,\infty)}(\mathbb{R}^n)^{[0,d)}$ be the set of trajectories

It induces a monad on Set

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

The fixpoint expression says a program either produces a trajectory and resumes or terminates with a value of type X

Semantics without Iteration

Let $\sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)}$ be the set of trajectories

It induces a monad on Set

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

The fixpoint expression says a program either produces a trajectory and resumes or terminates with a value of type X

Kleisli composition amounts to concatenation of lists of trajectories

Semantics without Iteration

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 $n = 1$)

$$[\![\mathrm{x}' = 1 \ \texttt{for} \ 1]\!](0) = (\underbrace{[\lambda t \in [0,1).\ 0 + t]}_{\text{list of size 1}}, 1)$$

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

$$= (\underbrace{[\lambda t \in [0,1).\ 0 + t,\ \lambda t \in [0,1).\ 1 + t]}_{\text{list of size 2}}, 2)$$

Semantics without Iteration

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 $n = 1$)

$$[\![\mathrm{x}' = 1 \texttt{ for } 1]\!](0) = (\underbrace{[\lambda t \in [0,1).\ 0 + t]}_{\text{list of size 1}}, 1)$$

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

$$= (\underbrace{[\lambda t \in [0,1).\ 0 + t,\ \lambda t \in [0,1).\ 1 + t]}_{\text{list of size 2}}, 2)$$

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

Semantics with Iteration

Instead of using the least fixpoint we use the greatest

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

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

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

Semantics with Iteration

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

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

Example (with $n = 1$)

$$[\![\texttt{while } \textbf{true} \texttt{ do } \{\texttt{x}' = 1 \textbf{ for } 1\}]\!](0)$$

$$= \underbrace{[\lambda t \in [0,1).\ 0 + t,\ \lambda t \in [0,1).\ 1 + t, \lambda t \in [0,1).\ 2 + t, \ldots]}_{\text{infinite list of trajectories}}$$

Going Extensional

The proposed semantics is intensional *e.g.*

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

We wish to abstract away from invisible intermediate steps

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

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

A Retraction Emerges

The previous operation induces a retraction

$$\begin{tikzcd} \hat{\mathrm{H}} \arrow[r, two heads, bend left, "\rho"] & \left(X \mapsto \sum_{d \in [0,\infty)} (\mathbb{R}^n)^{[0,d)} \times X + \sum_{d \in [0,\infty]} (\mathbb{R}^n)^{[0,d)}\right) \arrow[l, hook, bend left, "\nu"] \end{tikzcd}$$

ρ resorts to concatenation of trajectories and ν is defined as

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

Denote the functor on the right-hand side by H

An Extensional Hybrid Monad

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

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 validates the aforementioned equality

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

and other expected ones, *e.g.*

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

Interpretation via H

Interpretation via H validates the aforementioned equality

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

and other expected ones, *e.g.*

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

Also it gives rise to different kinds of while-loop ...

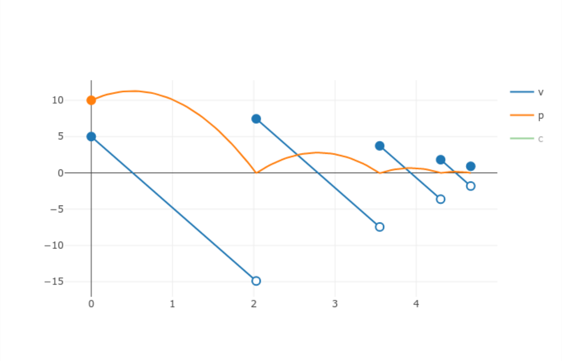
A Taxonomy of While Loops

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

The tool Lince

An implementation of the semantics available at

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



More details in [Goncharov et al., 2020]

Table of Contents

Hybrid Components

We wish to combine the notion of internal state with that of hybrid behaviour

Useful for studying (the orchestration of) computational units that interact with physical processes

$$\begin{aligned}&\mathtt{while}\ \mathtt{true}\ \mathtt{do}\ \{\ \mathtt{if}\ \mathtt{f}\big(\mathtt{readSens}_1(), \mathtt{readSens}_2()\big)\\ &\qquad\qquad\quad\ \mathtt{then}\ \mathtt{heatReactor}()\\ &\qquad\qquad\quad\ \mathtt{else}\ \mathtt{coolReactor}()\ \}\end{aligned}$$

The State Monad

Let S be a set of states

Functorial part defined by $X \mapsto (S \times X)^S$

Kleisli composition amounts to carrying the current state from one computation to another

Does not support iteration

State Monad + Extensional Hybrid Monad

General categorical results allow us to combine both monads

The functorial part of the combined monad is given by

$$X \mapsto (\mathrm{H}(S \times X))^{S}$$

Kleisli composition is a combination of the previous compositions

State Monad + Extensional Hybrid Monad

General categorical results allow us to combine both monads

The functorial part of the combined monad is given by

$$X \mapsto (\mathrm{H}(S \times X))^S$$

Kleisli composition is a combination of the previous compositions

A hybrid component $\mathrm{c} : \mathbb{A} \to \mathbb{B}$ is interpreted as a map

$$[\![\mathrm{c}]\!] : [\![\mathbb{A}]\!] \to (\mathrm{H}(S \times [\![\mathbb{B}]\!]))^S$$

State + Hybrid Iteration in Programming

The combined monad inherits iteration from the hybrid one

$$
\begin{aligned}
& f : X \to (\mathrm{H}(S \times (Y + X)))^S \\
\Rightarrow\ & f : X \times S \to \mathrm{H}(S \times (Y + X)) \\
\Rightarrow\ & f : X \times S \to \mathrm{H}(S \times Y + S \times X) \\
\Rightarrow\ & f^\dagger : X \times S \to \mathrm{H}(S \times Y) \\
\Rightarrow\ & f^\dagger : X \to (\mathrm{H}(S \times Y))^S
\end{aligned}
$$

Theorem

The combined monad is Elgot

Table of Contents

Conclusions and Future Work

We saw how to interpret hybrid behaviour in different settings

- hybrid automata
- hybrid while-language
- while-language + hybrid components

Currently working on the extension of the previous results to a quantitative setting

- quantitative bisimulation
- probabilities
- stability

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

Goncharov, S., Neves, R., and Proença, J. (2020).
Implementing hybrid semantics: From functional to imperative.
In Pun, V. K. I., Stolz, V., and Simão, A., editors, *Theoretical Aspects of Computing - ICTAC 2020 - 17th International Colloquium, Macau, China, November 30 - December 4, 2020, Proceedings*, volume 12545 of *Lecture Notes in Computer Science*, pages 262–282. Springer.

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