Languages and Models for Hybrid Automata: A Coalgebraic Perspective

Renato Neves, Luís S. Barbosa

INESC TEC (HASLab) & Minho University, Portugal
nevrenato@di.uminho.pt, lsb@di.uminho.pt

Abstract

We study hybrid automata from a coalgebraic point of view. We show that such a perspective supports a generic theory of hybrid automata with a rich palette of definitions and results. This includes, among other things, notions of bisimulation and behaviour, state minimisation techniques, and regular expression languages.

Keywords: Coalgebra, hybrid automata, bisimulation, regular expression.

1. Introduction

1.1. Motivation and context

Systems whose behaviour has both discrete and continuous aspects are traditionally qualified as ***hybrid*** [1, 2, 3]. They often arise as complex networks of computational units, sensors, and actuators, suitably coordinated so that a desired outcome can be reached. A classic example is provided by a cruise control system as it is essentially a digital device that interacts with actuators and sensors which control and measure a vehicle's velocity. The same pattern also occurs in thermostats, planes, electric grids, and surgical robots (see *e.g.* [3, 2, 4]).

The formal specification and analysis of hybrid systems typically resorts to the theory of ***hybrid automata*** [5], whose distinguishing feature is the ability of state variables to evolve continuously — thus making them able to express the behaviour of physical processes, like movement, time, temperature, or pressure. In addition, they carry syntactical machinery (guards, state invariants, and assignments) to facilitate the description of complex behaviour in a concise manner.

Being perhaps the most famous answer to the rapid emergence of hybrid systems [6, 7, 8], hybrid automata form an active research area that encompasses a broad range of topics, from decidability issues [5] to extensions that cater for ***input*** mechanisms [6, 9] and ***uncertainty*** [10, 11]. To create a new extension, however, frequently entails a return to the drawing board in order to redesign or adapt whatever definitions, notions or techniques are deemed relevant for it.

The notion of ***bisimulation*** is a prime example of this, as it usually takes an apparently different form in each extension.

In a previous paper [12] we showed how to treat a number of variants of hybrid automata in a uniform manner using the theory of ***coalgebras*** [13]. In particular, we proved that the notions of bisimulation adopted by different types of hybrid automata are instances of a generic, coalgebraic definition; and we discussed briefly how to introduce new variants of hybrid automata in a systematic way, with notions of bisimulation and behaviour coming for free and tailored to the context at hand. This was illustrated by reconstructing the theory of some classes of hybrid automata (for example, ***reactive Markov*** hybrid automata), and developing new variants (for example, of what was then called ***replicating*** hybrid automata). On the whole, that paper provided, we believe, a first step towards a coalgebraic, uniform theory of hybrid automata.

1.2. Contributions

In the current paper we give a complete and formal account of the research avenue announced in [12]. More concretely,

- we show that every functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ induces a category of a specific type of hybrid automata (F shapes their 'discrete' transition type) and also a category that suitably captures their semantics. Both are categories of coalgebras and therefore several useful notions come for free.

- We prove the existence of a 'semantics' functor between these two categories, which, intuitively, associates every hybrid automaton to its corresponding model. This functor generalises the standard semantics for both ***classic hybrid automata*** [5] and ***probabilistic hybrid automata*** [11].

The idea of seeing hybrid automata as coalgebras emerged from our adoption of the ***black-box*** perspective as a strategy to handle hybrid systems (*cf.* [14]). In this view, the (discrete) state transitions of a hybrid system are internal, hidden from the environment whereas the continuous evolutions are external, making up the observable behaviour — think again about the operation of a cruise control system. One cannot directly observe the computations of the digital device; only their influence over the car's velocity which evolves over time. The black-box approach is a central concept in the theory of coalgebras and thus it naturally leads to the idea of regarding hybrid automata as coalgebras.

As discussed in [12], the coalgebraic perspective facilitates the analysis and design of hybrid automata once a suitable, coalgebraic semantics for them is set. For example, it provides a uniform, canonical notion of behaviour that faithfully reflects the black-box perspective and frames the behaviour into well known constructions (*e.g.* streams, binary trees) that mark a clear frontier between the discrete behaviour and the continuous one. Interestingly, the coalgebraic view also facilitates the understanding of hybrid automata and helps to systematise the concept along a plethora of, often elaborated, definitions in the literature. In its most basic variant, a hybrid automaton is reduced to a machine that

from a state (internally) jumps to another and (externally) produces a continuous evolution. Moreover, the coalgebraic characterisation paves the way to a hierarchy of different types of hybrid automata organised with respect to their 'expressivity', a concept which is itself understood here coalgebraically.

In order to discuss some of these benefits in detail, we devote a large portion of the paper to a specific variant of hybrid automata classified as ***reactive*** — intuitively, it combines deterministic evolutions with an input dimension. We thoroughly study the coalgebraic theory of this variant, with special focus on the notions of behaviour and bisimulation. Furthermore, we use standard coalgebraic techniques to show that reactive hybrid automata admit a Kleene-like theorem. Along the process an illuminating message emerges: hybrid automata are hybrid also in the sense that they are neither purely syntactic nor purely semantic entities.

1.3. Roadmap

In Section 2 we recall briefly the theory of hybrid automata [5] and the theory of coalgebras [13]. Then, in the same section, we start our study by showing that classic hybrid automata, and some of their variants (*e.g.* reactive hybrid automata), can be straightforwardly interpreted as coalgebras.

In Section 3 we explore and discuss the coalgebraic theory of reactive hybrid automata. In particular, we introduce the aforementioned Kleene-like theorem, the semantics functor associated with reactive hybrid automata, and some of its properties. We also study the notions of behaviour and bisimulation that coalgebras provide in the context of this variant.

In Section 4 we take the generic perspective. In particular, we establish the formal correspondence between functors $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ and variants of hybrid automata that document [12] alludes to. This leads to the (re)discovery of several variants of hybrid automata (*e.g.* probabilistic [11], ***weighted***, and ***replicating***). In the same section we revisit the generic notion of Φ-bisimulation and the hierarchy of hybrid automata introduced in [12], but now under the light of the functorial semantics that the current paper provides.

Finally, in Section 5 we conclude and discuss future work.

We assume that the reader has basic familiarity with category theory [15] and topology [16]. Throughout the paper we use an arrow with a tail $f : A \rightarrowtail B$ to stress that a map $f : A \to B$ is injective. Dually, we use a two-headed arrow $f : A \twoheadrightarrow B$ to represent a surjection. We use $\mathcal{P}$ to denote the powerspace construction and $\mathcal{D}$ to denote the distribution space construction whose distributions have finite support. Finally, for two sets A and B we denote by B^A the set of maps from A to B. If A and B are topological spaces then B^A denotes the set of continuous maps from A to B.

2. Preliminaries

2.1. Classic and probabilistic hybrid automata

Let us start by formally introducing the notion of predicate and the classic definition of hybrid automata [5].

Definition 2.1. Given a finite set X, the set of predicates φ over X, denoted by $\mathsf{P}(X)$, is generated by the grammar below in the left.

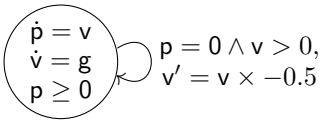
$$\varphi \ni \neg\varphi \mid \varphi \wedge \varphi \mid t < t \mid t = t, \qquad t \ni t + t \mid t \cdot t \mid x \mid r \qquad (x \in X, r \in \mathbb{R})$$

Definition 2.2. A hybrid automaton is a tuple $(M, E, X, \mathsf{inv}, \mathsf{dyn}, \mathsf{asg}, \mathsf{grd})$ such that

- M is a finite set of ***control modes*** and $E \subseteq M \times M$ is a transition relation between them.
- X is a finite set of real-valued variables $\{x_1, \ldots, x_n\}$.
- $\mathsf{inv} : M \to \mathsf{P}(X)$ is a function that associates modes to invariants, the latter being given as predicates over the variables in X.
- $\mathsf{dyn} : M \to \mathsf{P}(X \cup \dot{X})$ is a function that associates each mode to a predicate over $X \cup \dot{X}$, where the set $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ represents the first derivatives of the variables in X. This map is used to dictate which continuous evolutions may occur in a given mode.
- $\mathsf{asg} : E \to \mathsf{P}(X \cup X')$ is a function that returns a predicate over $X \cup X'$ for a given edge, where $X' = \{x'_1, \ldots, x'_n\}$ represents the variables in X immediately after a discrete jump. In other words, asg is a function that provides an assignment to each edge.
- Finally, the function $\mathsf{grd} : E \to \mathsf{P}(X)$ associates each edge with a guard.

The following examples may help to illustrate and clarify some aspects of this quite complex definition.

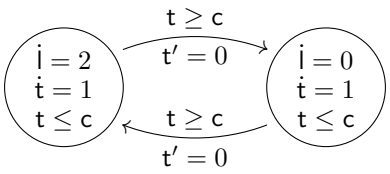
Example 2.3. Consider a bouncing ball dropped at a specific positive height p and with no initial velocity v. Due to the gravitational acceleration g, it falls into the ground and then bounces back up, losing part of its kinetic energy in the process. The following hybrid automaton sums up this behaviour.



Observe that only one mode exists; let us call it m. Furthermore there exists exactly one discrete transition: $(m, m) \in E$. Then $X = \{\mathsf{p}, \mathsf{v}\}$, and $\mathsf{inv}(m)$ is $\mathsf{p} \geq 0$. Moreover, $\mathsf{grd}(m, m)$ is $\mathsf{p} = 0 \wedge \mathsf{v} > 0$, $\mathsf{dyn}(m)$ is $\dot{\mathsf{p}} = \mathsf{v} \wedge \dot{\mathsf{v}} = \mathsf{g}$, and $\mathsf{asg}(m, m)$ is $\mathsf{v}' = \mathsf{v} \times -0.5 \wedge \mathsf{p}' = \mathsf{p}$. Note that the second conjunct does not appear in the hybrid automaton above, a common practice to avoid a notational burden.

Example 2.4. Consider now a system comprised of a tank and a valve connected to it. The valve allows water to flow in filling the tank at a rate of 2cm/s

during intervals of c seconds; between these periods the valve is shut (also) for c seconds. We can describe this behaviour via the hybrid automaton below.



The variable l denotes the water level, which rises when the valve is open (differential equation $\dot{\mathsf{l}} = 2$). The differential equation $\dot{\mathsf{t}} = 1$ defines the passage of time, which, along with invariant $\mathsf{t} \leq \mathsf{c}$, forces the current mode to be active for at most c seconds. On the other hand, the guard $\mathsf{t} \geq \mathsf{c}$ and assignment $\mathsf{t}' = 0$ present in both transitions force the current mode to be active at least c seconds before a switch. Finally, note that the guard $\mathsf{t} \geq \mathsf{c}$ does not force a transitions to happen, but only makes it possible. This means that, if not for invariant $\mathsf{t} \leq \mathsf{c}$, the valve could be open (or shut) indefinitely — note, however, that according to the assumptions below this type of behaviour cannot occur.

Contrary to document [5], we do not consider initial states nor labels in the definition of hybrid automata. This is because we wish to keep our results simple and intuitive. Moreover both mechanisms can be accommodated later on in a straightforward manner.

Assumptions 2.5. We also make the following assumptions.

1. For every mode the function dyn returns a system of differential equations with exactly ***one solution***. Even if it seems too restrictive this is actually a common assumption (*e.g.* [17, 18, 19]) since most hybrid systems described in the literature rarely involve nonlinear differential equations. The important point is that this condition allows function dyn to induce a map

$$\mathsf{sol} : M \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$$

   such that given a pair $(m, v) \in M \times \mathbb{R}^n$, the map

$$\mathsf{sol}\,(m, v, -) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$$

   is the solution to the system of differential equations associated to a specific mode and valuation $(m, v) \in M \times \mathbb{R}^n$. Its domain ($\mathbb{R}_{\geq 0}$) represents time and n is the cardinality of set X (*i.e.* the number of real-valued variables).
2. Hybrid automata cannot have ***invalid jumps***. Intuitively, they cannot jump from a valid state $(m, v) \in M \times \mathbb{R}^n$ into an invalid one, where by valid we mean that

$$(m, v) \in \{(n, u) \in M \times \mathbb{R}^n \mid u \models \mathsf{inv}(n)\}$$

and by invalid we mean the opposite of valid. Formally, assume that for every pair $(m_1, v_1, m_2, v_2) \in M \times \mathbb{R}^n \times M \times \mathbb{R}^n$ such that

$$(m_1, m_2) \in E, \quad v_1 \models \mathsf{grd}(m_1, m_2), \quad (v_1, v_2) \models \mathsf{asg}(m_1, m_2)$$

entails property $v_2 \models \mathsf{inv}(m_2)$. This is another standard assumption. Later on, when going generic, we will need to express it in terms of factorisations and to generalise the notion of valid state. In the sequel, we denote the set of valid states of a hybrid automaton X by Z_X, or simply Z if no ambiguities arise.

3. All ***assignments must be deterministic*** *i.e.* they must take the form $x' = \theta$, where θ is a term that corresponds to exactly one real value. Clearly, both the bouncing ball and the water tank system respect this assumption.
4. Finally, as soon as all edges outgoing a mode become enabled (*i.e.* the associated guards are satisfied) that mode must switch and never before. A similar condition is adopted in [20], where hybrid automata with this property are called ***time-deterministic***. In detail, assume that each pair $(m, v) \in Z$ has exactly one duration ($[0, d] \subseteq \mathbb{R}_{\geq 0}$) for its evolution $\mathsf{sol}(m, v, -) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, which intuitively corresponds to the time that the mode m takes to jump starting from (m, v). At the end of the duration all associated guards must be enabled. This happens, for example, in the hybrid automaton that describes the tank-and-valve (c seconds) and the bouncing ball system (the time that the ball takes to reach the ground from a specific height and velocity). Later on we will further discuss this condition.

The traditional semantics of hybrid automata is given in terms of labelled transition systems [5].

Let 1 be a one-point set, and $A + B$ the disjoint union of two arbitrary sets A and B.

Definition 2.6. A hybrid automaton induces a labelled transition system (Z, L, T) such that $L = 1 + \mathbb{R}_{\geq 0}$ and the transition relation $T \subseteq Z \times L \times Z$ is defined as $\big((m_1, v_1), l, (m_2, v_2)\big) \in T$ iff

1. $l \in 1$ entails $(m_1, m_2) \in E$, $v_1 \models \mathsf{grd}(m_1, m_2)$, and $(v_1, v_2) \models \mathsf{asg}(m_1, m_2)$
2. $l \in \mathbb{R}_{\geq 0}$ entails $m_1 = m_2$, $\mathsf{sol}(m_1, v_1, l) = v_2$, and

$$\mathsf{sol}(m_1, v_1, t) \models \mathsf{inv}(m_1) \quad (t \in [0, l])$$

We write a triple $(z_1, l, z_2) \in T$ as $z_1 \xrightarrow{l} z_2$.

Example 2.7. Recall the hybrid automaton that describes the bouncing ball in Example 2.3. The associated labelled transition system (Z, L, T) is defined as follows: $Z = \{m\} \times \mathbb{R}_{\geq 0} \times \mathbb{R}$ and $(m, p_1, v_1) \xrightarrow{l} (m, p_2, v_2)$ iff

1. $l \in 1$ entails $p_1 = 0 \wedge v_1 > 0$ and $v_2 = v_1 \times -0.5 \wedge p_2 = p_1$
2. $l \in \mathbb{R}_{\geq 0}$ entails $\mathsf{sol}(m, p_1, v_1, l) = (p_2, v_2)$, and

$$\mathsf{sol}(m, p_1, v_1, t) \models \mathsf{p} \geq 0 \qquad (t \in [0, l])$$

The function sol, determined by dyn, describes the continuous evolution of the ball's position and velocity between jumps.

Note that both discrete events and continuous evolutions are embedded in the relation T. Not only this makes difficult to adopt the black-box perspective mentioned before, but it also makes the verification of hybrid automata extremely challenging, as a large number of states and edges needs to be taken into consideration. The standard technique for overcoming this issue is to quotient the state space by a bisimulation equivalence, *i.e.* to collapse states that possess equivalent behaviour. The resulting states become symbolic representations of (possibly infinite) regions, and verification techniques are applied to the reduced system instead.

Note also that mixing discrete events with continuous evolutions blurs the frontier between discrete and continuous behaviour, and consequently hampers the application of techniques that require either a purely continuous universe or a purely discrete one.

Definition 2.8. Consider the underlying labelled transition system (Z, L, T) of a hybrid automaton and an equivalence relation $\Phi \subseteq Z \times Z$ over the states. A ***Φ-bisimulation*** $R \subseteq Z \times Z$ is a relation R such that $(x_1, y_1) \in R$ (or, more concisely, $x_1 \, R \, y_1$) entails the following cases:

1. $x_1 \, \Phi \, y_1$, and for every label $l \in L$
2. if $x_1 \xrightarrow{l} x_2$ then there exists a state y_2 such that $y_1 \xrightarrow{l} y_2$ and $x_2 \, R \, y_2$,
3. if $y_1 \xrightarrow{l} y_2$ then there exists a state x_2 such that $x_1 \xrightarrow{l} x_2$ and $x_2 \, R \, y_2$.

Two states $x, y \in Z$ are ***Φ-bisimilar*** (in symbols, $x \equiv^{\Phi} y$) if they are related by a Φ-bisimulation.

This is the notion of bisimulation that hybrid automata traditionally use.

Let us consider now a probabilistic variant of hybrid automata, often referred to as ***probabilistic hybrid automata*** [10, 11].

Definition 2.9. A probabilistic hybrid automaton is a tuple $(M, X, \mathsf{inv}, \mathsf{dyn}, e)$ where

- M is a finite set of control modes.
- X is a finite set of real-valued variables $\{x_1, \ldots, x_n\}$.
- $\mathsf{inv} : M \to \mathsf{P}(X)$ is a function that associates each mode to a predicate over the variables in X.
- $\mathsf{dyn} : M \to \mathsf{P}(X \cup \dot{X})$ is a function that associates each mode to a predicate over the variables in $X \cup \dot{X}$.

- $e : M \to \mathcal{PD}\,(M \times \mathsf{P}(X \cup X') \times \mathsf{P}(X))$ is a function that associates each mode to a set of probability distributions over modes, assignments, and guards.

Probabilistic hybrid automata allow to dictate the likelihood of events associated with digital computations. For example, the probability of a computer malfunction if temperature gets too high or the probability of a reset after a specific time is achieved. Technically, these automata harbour such a behaviour due to their transition map, which before a jump (or state transition) presents us with set of distributions (over modes, guards, and assignments) to choose from. The chosen distribution will then be used to compute the likelihood of a given state being the next one in the execution process.

The documents [11, 10] provide several elegant examples of probabilistic hybrid automata and explain some of their intricacies. Their semantics is also discussed. It is given in terms of probabilistic transition systems and it requires some preliminary definitions which we will recall next.

Definition 2.10. Consider a probabilistic hybrid automaton. Given a pair $(m, v) \in Z$, define $N_{(m,v)} \subseteq \mathcal{D}Z$ as the set such that $\mu \in N_{(m,v)}$ iff there exists a distribution $\xi \in e(m)$ that respects the following condition.

Let $\{(m_1, \phi_1, \varphi_1), \ldots, (m_n, \phi_n, \varphi_n)\}$ denote the support set of ξ and $\{v_1, \ldots, v_n\}$ denote the set of valuations such that

$$(v, v_i) \models \phi_i \qquad (1 \leq i \leq n)$$

Then for every pair $(m', v') \in Z$ we have

$$\mu(m', v') = \sum_{i \in I} \xi(m_i, \phi_i, \varphi_i), \qquad I = \{1 \leq i \leq n \mid m' = m_i, v' = v_i, v \models \varphi_i\}$$

Intuitively, the summation above is used to add the probabilities of triples in $M \times \mathsf{P}(X \cup X') \times \mathsf{P}(X)$ that lead to the same result. For example, let the current value of a variable x be 10. Then, neglecting modes and guards, give probability 0.5 to the assignment $x = x{+}1$ and the same probability to assignment $x = 1{+}x$. Clearly, the probability of x to become 11 is 1.

Remark 2.11. The previous definition is slightly simpler than the one given in [11, 10]. This is a consequence of Assumptions 2.5 (3). Note also that the definition of probabilistic hybrid automata presented here (Definition 2.9) is slightly more general than the usual version, as we allow guards to be probabilistic as well. Finally, probabilistic hybrid automata traditionally come equipped with a colouring map $M \to C$. Such a feature can be accommodated in Definition 2.9 in a straightforward manner: recall the embedding

$$M^C \rightarrowtail \mathcal{P}(M \times C)$$

and observe that a probabilistic hybrid automaton with a colouring map $M \to C$ can be seen as a probabilistic hybrid automaton whose set of modes is a subset of the cartesian product $M \times C$.

Definition 2.12. A probabilistic hybrid automaton induces a probabilistic transition system

$$(Z, t : Z \times L \to \mathcal{P}\mathcal{D}Z)$$

such that $L = 1 + \mathbb{R}_{\geq 0}$ and the map $t : Z \times L \to \mathcal{P}\mathcal{D}Z$ is defined as

$$t(m, v, *) = N_{(m,v)}$$
$$t(m, v, r) = \begin{cases} \{\delta\} & \text{if } \mathsf{sol}(m, v, a) \models \mathsf{inv}(m) \quad (a \in [0, r]) \\ \emptyset & \text{otherwise} \end{cases}$$

where δ is the Dirac distribution of $\mathsf{sol}(m, v, r)$.

Finally, let us recall the standard notion of bisimulation for probabilistic hybrid automata.

Definition 2.13. Consider two arbitrary sets X, Y, a relation $R \subseteq X \times Y$, and two distributions $\mu_1 \in \mathcal{D}X$, $\mu_2 \in \mathcal{D}Y$. Define the relation $\asymp_R \subseteq \mathcal{D}X \times \mathcal{D}Y$ as $\mu_1 \asymp_R \mu_2$ iff there is a distribution $\nu \in \mathcal{D}(X \times Y)$ that respects the following condition: for all elements $x \in X$, $y \in Y$,

$$\nu(x, y) > 0 \text{ entails } x \, R \, y, \qquad \mu_1(x) = \nu(\{x\} \times Y), \qquad \mu_2(y) = \nu(X \times \{y\})$$

where for every $S \subseteq X \times Y$ the expression $\nu(S)$ denotes $\sum_{s \in S} \nu(s)$.

Definition 2.14. Consider a probabilistic hybrid automaton and its underlying probabilistic transition system

$$(Z, t : Z \times L \to \mathcal{P}\mathcal{D}Z)$$

Let $\Phi \subseteq Z \times Z$ be an equivalence relation. Then a relation $R \subseteq Z \times Z$ is a probabilistic Φ-bisimulation iff $z_1 \, R \, z_2$ entails:

1. $z_1 \, \Phi \, z_2$,
2. if $\mu_1 \in t(z_1, l)$ then there exists a distribution $\mu_2 \in t(z_2, l)$ such that $\mu_1 \asymp_R \mu_2$,
3. if $\mu_2 \in t(z_2, l)$ then there exists a distribution $\mu_1 \in t(z_1, l)$ such that $\mu_1 \asymp_R \mu_2$.

Two states $x, y \in Z$ are Φ-bisimilar (in symbols, $x \equiv^{\Phi} y$) if they are related by a Φ-bisimulation.

2.2. Coalgebras

The theory of coalgebras [13, 21] provides an abstract, categorial framework for state-based transition systems that allows to derive notions and results parametric on their transition type. The idea is that a functor $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ over a category C, typically the category Set (of sets and functions), corresponds to a specific transition type and the arrows typed as $X \to \mathsf{F}X$ (F-***coalgebras***, or simply coalgebras) comprise the corresponding family of state-based transition systems. To be concrete,

Definition 2.15. A functor $\mathsf{F} : \mathsf{C} \to \mathsf{C}$ gives rise to the category $\mathsf{CoAlg}(\mathsf{F})$ whose objects are F-coalgebras and morphisms between two coalgebras $(X, c : X \to \mathsf{F}X)$, $(Y, d : Y \to \mathsf{F}Y)$ are C-arrows $f : X \to Y$ that make the diagram below commute.

$$\begin{array}{ccc} X & \xrightarrow{f} & Y \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle d} \\ \mathsf{F}X & \xrightarrow[\mathsf{F}f]{} & \mathsf{F}Y \end{array}$$

Examples 2.16. Here are some classic examples of coalgebras.

1. Coalgebras for the powerset functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$, *i.e.* $\mathcal{P}$-coalgebras, are Kripke frames and *vice-versa*.
2. Let L denote a set of symbols and 2 the two-point set $\{0, 1\}$. Deterministic automata are $(\_ \times 2)^L$-coalgebras.
3. Let $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ be the functor of distributions, which given a set X and a map $f : X \to Y$ returns

$$\mathcal{D}X = \left\{ \mu \in [0,1]^X \mid \sum_{x \in X} \mu(x) = 1,\ \mathsf{supp}(\mu) \text{ finite} \right\}$$
$$\mathcal{D}f(\mu)(y) = \sum_{x \in f^{-1}(y)} \mu(x)$$

   $\mathcal{D}$-coalgebras are ***discrete Markov chains*** (*cf.* [22]).

A category of coalgebras brings for free a number of generic, useful constructions (*e.g.* bisimulation and behaviour). The following lines briefly review some of those considered in the paper. For simplicty's sake this revision adopts Set as the working category, but note that most notions presented below can be considered in other categories as well.

Definition 2.17. Let $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ be a functor over Set and consider an F-coalgebra (X, c). We say that an F-coalgebra (S, d) is a ***subcoalgebra*** of (X, c) if there is a morphism $(S, d) \rightarrowtail (X, c)$ in $\mathsf{CoAlg}(\mathsf{F})$ whose underlying function is an inclusion.

Theorem 2.18. *Consider a functor* $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ *that preserves intersections (see [13]). The set of subcoalgebras of an* F*-coalgebra* (X, c) *is closed under intersections, the order being set inclusion.*

Thus, assuming that a functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ preserves intersections, for every F-coalgebra (X, c) and state $x \in X$ there exists a subcoalgebra $(\overline{x}, c)$ of (X, c) whose carrier contains the state x and is the smallest one among such all subcoalgebras of (X, c). We invite the interested reader to read more about subcoalgebras in *e.g.* [13, 21].

Let us now recall the notions of bisimulation and behaviour.

Definition 2.19. Consider two F-coalgebras (X, c), (Y, d) and a relation $R \subseteq X \times Y$. Then R is an F-***bisimulation*** (or simply, bisimulation) if there exists a third coalgebra (R, r) that makes the following diagram commute.

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Y \\ {\scriptstyle c}\downarrow & & {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle d} \\ \mathsf{F}X & \xleftarrow[\mathsf{F}\pi_1]{} & \mathsf{F}R & \xrightarrow[\mathsf{F}\pi_2]{} & \mathsf{F}Y \end{array}$$

We say that two states $x \in X$ and $y \in Y$ are ***coalgebraically bisimilar***, in symbols $x \sim y$, if they are related by some F-bisimulation. If $R \subseteq X \times Y$ is an equivalence relation and an F-bisimulation then we call it an F-***bisimulation equivalence***.

Before giving examples of coalgebraic bisimulation let us fix some notation: the so-called ***Mealy*** functors

$$(\mathsf{F}(\_) \times O)^I : \mathsf{Set} \to \mathsf{Set}$$

play an important role in this paper. We denote them simply by $\mathsf{F}_O^I : \mathsf{Set} \to \mathsf{Set}$ due to their frequent presence. Moreover, we use the expression

$$\overline{f} : X \to Z^Y$$

to denote the curried form of a map $f : X \times Y \to Z$.

Remark 2.20. In many cases, currying allows to shift between the algebraic and the coalgebraic perspectives. In particular, it allows to see an algebra $X \times I \to X$ as a coalgebra $X \to X^I$ and *vice-versa*.

Consider now an Id_O^I-coalgebra $(X, \overline{\langle \mathrm{nxt}, \mathrm{out} \rangle})$ and a state $x \in X$. Whenever no ambiguities occur, denote the expression $\mathrm{nxt}(x, i)$ by x_i and the expression $\mathrm{out}(x, i)$ by $x[i]$.

Example 2.21. Consider two Id_O^I-coalgebras (X, c), (Y, d) and a relation $R \subseteq X \times Y$. Then R is a bisimulation iff for all states $x \in X$, $y \in Y$ the condition $x \; R \; y$ entails

$$x[i] = y[i], \qquad x_i \; R \; y_i \qquad (i \in I)$$

Example 2.22. Bisimulation for $\mathcal{P}$-coalgebras corresponds exactly to bisimulation for Kripke frames [13, Example 2.1].

Example 2.23. Consider two $\mathcal{D}$-coalgebras (X, c), (Y, d) and a relation $R \subseteq X \times Y$. Then R is a bisimulation iff for all states $x \in X, y \in Y$ the condition $x \, R \, y$ entails $c(x) \asymp_R d(y)$.

Example 2.24. Consider two $\mathcal{PD}$-coalgebras (X, c), (Y, d) and a relation $R \subseteq X \times Y$. Then R is a bisimulation iff for all states $x \in X, y \in Y$ the condition $x \, R \, y$ entails the following conditions.

- If $\mu_1 \in c(x)$ then there exists a distribution $\mu_2 \in d(y)$ such that $\mu_1 \asymp_R \mu_2$,
- if $\mu_2 \in d(y)$ then there exists a distribution $\mu_1 \in c(x)$ such that $\mu_1 \asymp_R \mu_2$.

Note the similarity between the last example and probabilistic Φ-bisimulation (Definition 2.14).

The following two propositions are proved in [13].

Proposition 2.25. *Let $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ be a functor that preserves weak pullbacks and consider a morphism $f : (X, c) \to (Y, d)$ in $\mathsf{CoAlg}(\mathsf{F})$ between two F-coalgebras. The graph $R \subseteq X \times Y$ of the map $f : X \to Y$ is a bisimulation for (X, c) and (Y, d).*

Proposition 2.26. *Let $R \subseteq X \times X$ be a bisimulation equivalence for a coalgebra (X, c). There exists a coalgebra*

$$X/R \to \mathsf{F}(X/R)$$

that makes the diagram below commute with $q : X \twoheadrightarrow X/R$ as the quotient map induced by R.

$$\begin{array}{ccc} X & \xrightarrow{\quad q \quad} & X/R \\ {\scriptstyle c}\downarrow & & \downarrow \\ \mathsf{F}X & \xrightarrow[\mathsf{F}q]{} & \mathsf{F}(X/R) \end{array}$$

Let $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ be a functor over Set. Under mild conditions the category $\mathsf{CoAlg}(\mathsf{F})$ has a final object, often called the ***final coalgebra***. Unfolding the definition of final object and assuming its existence in $\mathsf{CoAlg}(\mathsf{F})$, every F-coalgebra has a unique morphism in $\mathsf{CoAlg}(\mathsf{F})$ to the final F-coalgebra. Intuitively, the final F-coalgebra collects the behaviours of all F-coalgebras and the universal maps associate a state of an F-coalgebra to its behaviour.

Example 2.27. Take the functor $(\_ \times O) : \mathsf{Set} \to \mathsf{Set}$ where O is an arbitrary set. The category $\mathsf{CoAlg}(\_ \times O)$ has a final coalgebra

$$\langle \mathrm{tl}, \mathrm{hd} \rangle : O^\omega \to O^\omega \times O$$

where O^ω is the set of infinite lists (***streams***) whose values are in O, and, for a given list, the map $\mathrm{hd} : O^\omega \to O$ selects the first element of the list and the map $\mathrm{tl} : O^\omega \to O^\omega$ discards the first element of the list.

This means that for a coalgebra $X \to X \times O$ the behaviour of each state $x \in X$ is an infinite sequence of values in O.

Example 2.28. Take the functor $\mathrm{Id}_O^I : \mathsf{Set} \to \mathsf{Set}$ where I and O are arbitrary sets. The associated category of coalgebras has a final coalgebra

$$\overline{\langle \mathrm{sng}, \mathrm{apd} \rangle} : O^{I^+} \to \left(O \times O^{I^+} \right)^I$$

where I^+ is the set of non-empty lists whose values are in I and the maps $\mathrm{sng} : O^{I^+} \times I \to O$, $\mathrm{apd} : O^{I^+} \times I \to O^{I^+}$ are defined by the equations below.

$$\mathrm{sng}(f, i) = f[i], \qquad \mathrm{apd}(f, i)(is) = f\ (i : is)$$

Note that the elements of O^{I^+} can be intuitively seen as trees whose nodes are labelled by elements of O and whose edges are labelled by elements of I.

Example 2.29. Recall that deterministic automata are Id_2^L-coalgebras. Given an Id_2^L-coalgebra (X, c), the behaviour of a state $x \in X$ corresponds precisely to a set of words of the alphabet L.

In the category Set a functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ admits a final coalgebra whenever it is ***bounded*** (*cf.* [13]). This is not a too strong condition. Indeed, it holds for all polynomial functors, the finitary version of the powerset functor ($\mathcal{P}_\omega$), the distribution functor with finite support ($\mathcal{D}$), and all composites made up of these cases. The reader will find in *e.g.* [13, 23, 24] a complete characterisation of the condition of boundedness and related properties.

2.3. Hybrid automata as coalgebras

We will now show that classic hybrid automata are in fact instances of a particular type of coalgebra. We start with a simple but useful remark.

Remark 2.30. For every finite set X denote the cartesian product $\mathsf{P}(X \cup X') \times \mathsf{P}(X)$ by Tr. Then note that each hybrid automaton induces the composite below.

$$M \times M \xrightarrow{E?} E + 1 \xrightarrow{\langle \mathsf{asg}, \mathsf{grd} \rangle + \mathrm{id}} \mathsf{Tr} + 1$$

which associates every pair in E with an element of Tr (an assignment and a guard) and every other pair with 'fail'. We denote this composite by

$$\mathrm{nxt} : M \times M \to \mathsf{Tr} + 1$$

Observe that it is an equivalent representation of the transition relation $E \subseteq M \times M$ together with the map $\mathsf{asg} : E \to \mathsf{P}(X' \cup X)$ and the map $\mathsf{grd} : E \to \mathsf{P}(X)$. Moreover, we can embed it into the set of maps $\mathcal{P}(M \times \mathsf{Tr})^M$ as shown by the following isomorphisms

$$(\mathsf{Tr} + 1)^{M \times M} \rightarrowtail (\mathcal{P}\,\mathsf{Tr})^{M \times M} \cong ((\mathcal{P}\,\mathsf{Tr})^M)^M \cong \mathcal{P}(M \times \mathsf{Tr})^M$$

Therefore every hybrid automaton is equivalently represented by a coalgebra typed as

$$\langle \mathrm{nxt}, \mathrm{out} \rangle : M \to \mathcal{P}(M \times \mathsf{Tr}) \times \mathsf{Ev}$$

where $\mathrm{out} = \langle \mathsf{dyn}, \mathsf{inv} \rangle : M \to \mathsf{Ev}$, $\mathsf{Ev} = \mathsf{P}(X \cup \dot{X}) \times \mathsf{P}(X)$.

Even if quite simple this remark has several useful consequences. For example, it tells that classic hybrid automata can be organised in a category of coalgebras, and therefore part of their theory will come for free. Moreover, it provides the basis for a systematic, coalgebraic description of hybrid automata and associated variants, in many cases the coalgebraic descriptions being simpler than their standard counterparts. For example, probabilistic hybrid automata are simply finite coalgebras

$$M \to \mathcal{PD}(M \times \mathsf{Tr}) \times \mathsf{Ev}$$

The perspective 'hybrid automata as coalgebras' also allows to instantiate previous results on ***coalgebraic regular expressions*** [25] to derive for free a Kleene-like theorem for different variants of hybrid automata.

3. Reactive Hybrid Automata

This section illustrates the coalgebraic approach by focusing on the notion of reactive hybrid automata, described in [12, 9]. Formally,

Definition 3.1. A ***reactive hybrid automaton*** is a finite $\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}}$-coalgebra

$$M \to (M \times \mathsf{Tr} \times \mathsf{Ev})^I$$

with I a finite set.

We start by pursuing a Kleene-like theorem for reactive hybrid automata, using document [25] as basis.

3.1. A coalgebraic language

The following definitions and results on semilattices will be useful for this paper.

Definition 3.2. Recall that a ***join-semilattice*** $(X, \vee)$ is a set equipped with an idempotent, commutative, and associative operation $\vee : X \times X \to X$. It is ***bounded*** if there exist two points $\{\bot, \top\} \subseteq X$ such that for any $x \in X$ the equations below hold.

$$\bot \vee x = x, \qquad \top \vee x = \top$$

Definition 3.3. The product $(X \times Y, \vee_{X\times Y})$ of join-semilattices $(X, \vee_X)$, $(Y, \vee_Y)$ is the cartesian product $X \times Y$ equipped with the map $\vee_{X\times Y}$, given by the commuting diagram below.

$$\begin{array}{ccc} (X \times Y) \times (X \times Y) & \xrightarrow{\langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle} & (X \times X) \times (Y \times Y) \\ & \searrow^{\vee_{X\times Y}} & \downarrow{\scriptstyle \vee_X \times \vee_Y} \\ & & X \times Y \end{array}$$

Proposition 3.4. *The product of bounded join-semilattices is also bounded.*

Remark 3.5. Every set X induces a bounded join-semilattice $\mathcal{T}X = (X + 2, \vee)$ such that for any two elements $x_1, x_2 \in X$

$$x_1 \vee x_2 = \begin{cases} x_1 & \text{if } x_1 = x_2 \\ \top & \text{otherwise} \end{cases}$$

called the ***trivial join-semilattice*** of X.

We are ready to introduce a grammar of regular expressions for reactive hybrid automata. In order to be more familiar to the hybrid systems' community, this grammar (and associated notions) will be slightly different than the one introduced in [25, Chapter 4]. The changes, however, do not require substantial modifications to the theory nor to the underlying proofs of most results in the original reference.

Definition 3.6. Let I be a finite set of inputs, and X a finite set of variables. Regular expressions for reactive hybrid automata are given by the following grammar:

$$\begin{aligned} &\epsilon \ni \underline{\emptyset} \mid x \mid i\ (\epsilon \mid a) \mid i \downarrow b \mid \epsilon \oplus \epsilon \mid \mu x.\, \gamma \\ &\gamma \ni \underline{\emptyset} \mid i\ (\epsilon \mid a) \mid i \downarrow b \mid \gamma \oplus \gamma \mid \mu x.\, \gamma \qquad (a \in \mathcal{T}\,\mathsf{Tr}, b \in \mathcal{T}\,\mathsf{Ev}, i \in I, x \in X) \end{aligned}$$

We say that an expression ϵ is ***closed*** if every variable x in ϵ is under the scope of the binder μx. The set of closed expressions is denoted by Exp. Expressions γ that occur right after a binder μx are called ***guarded***. The set of guarded expressions is denoted by Exp_g.

Intuitively, for an input i the expression $i(\epsilon \mid a)$ denotes a transition to a state specified by ϵ. The letter a records the assignments and guards associated with the transition. The expression $i \downarrow b$ specifies the continuous behaviour associated with an input $i \in I$. Finally, the construct $\mu x.$ introduces recursion and the construct $\oplus$ works as a conjunction.

Remark 3.7. Assume that the set of inputs I is the singleton set. The grammar above can then be simplified into the following one.

$$\begin{aligned} &\epsilon \ni \underline{\emptyset} \mid x \mid (\epsilon \mid a) \mid b \mid \epsilon \oplus \epsilon \mid \mu x.\, \gamma \\ &\gamma \ni \underline{\emptyset} \mid (\epsilon \mid a) \mid b \mid \gamma \oplus \gamma \mid \mu x.\, \gamma \qquad (a \in \mathcal{T}\,\mathsf{Tr}, b \in \mathcal{T}\,\mathsf{Ev}, x \in X) \end{aligned}$$

Example 3.8. Recall the bouncing ball introduced in the previous section. Its behaviour is specified by the expression

$$\mu x.\, (x \mid \mathsf{a}) \ \oplus\ (\dot{\mathsf{p}} = \mathsf{v} \wedge \dot{\mathsf{v}} = \mathsf{g},\ \mathsf{p} \geq 0)$$

where a is the pair $(\mathsf{v}' = \mathsf{v} \times -0.5,\ \mathsf{p} = 0 \wedge \mathsf{v} > 0)$.

In the following lines we will show that the set of closed expressions Exp is the carrier of a coalgebra

$$\mathsf{Exp} \to (\mathsf{Exp} \times \mathsf{Cmd})^I$$

where Cmd is the product of join-semilattices $\mathcal{T}\,\mathsf{Tr}$ and $\mathcal{T}\,\mathsf{Ev}$. This provides a natural semantics for expressions and allows to relate the latter with modes of reactive hybrid automata in regard to coalgebraic bisimilarity.

Consider the composite of maps

$$X \times X \xrightarrow{=} 2 \xrightarrow{[\underline{a},\underline{b}]} Y$$

where $\underline{a}, \underline{b} : 1 \to Y$ are points in Y. Denote this composite by $a \lhd\ =\ \rhd b$ and, whenever no ambiguities arise, denote the expression $(a \lhd\ =\ \rhd b)\,(x, y)$ simply by $a \lhd (x = y) \rhd b$. It reads: 'if $x = y$ return a; otherwise return b'.

Definition 3.9. Define the coalgebra $\delta : \mathsf{Exp} \to (\mathsf{Exp} \times \mathsf{Cmd})^I$ as $\delta(\epsilon)(i) = (\epsilon[i], \epsilon_i)$ where

$$\begin{aligned}
\underline{\emptyset}[i] &= \bot & \underline{\emptyset}_i &= \underline{\emptyset} \\
j\ (\epsilon \mid a)[i] &= (a, \bot) \lhd (i = j) \rhd \bot & (j\ (\epsilon \mid a))_i &= \epsilon \lhd (i = j) \rhd \underline{\emptyset} \\
(j \downarrow b)[i] &= (\bot, b) \lhd (i = j) \rhd \bot & (j \downarrow b)_i &= \underline{\emptyset} \\
(\epsilon_1 \oplus \epsilon_2)[i] &= \epsilon_1[i] \vee \epsilon_2[i] & (\epsilon_1 \oplus \epsilon_2)_i &= (\epsilon_1)_i \oplus (\epsilon_2)_i \\
(\mu x.\gamma)[i] &= (\gamma[\mu x.\gamma/x])[i] & (\mu x.\gamma)_i &= (\gamma[\mu x.\gamma/x])_i
\end{aligned}$$

The expression $\gamma[\mu x.\gamma/x]$ denotes syntactic substitution. It reads: 'in the expression γ replace any free occurrence of x by $\mu x.\gamma$'.

Proposition 3.10. *The coalgebra $\delta : \mathsf{Exp} \to (\mathsf{Exp} \times \mathsf{Cmd})^I$ is well-defined. More concretely, the equations*

$$(\mu x.\gamma)[i] = (\gamma[\mu x.\gamma/x])[i], \qquad (\mu x.\gamma)_i = (\gamma[\mu x.\gamma/x])_i$$

do not entail an infinite number of computations.

Proof. The proof is analogous to the one in [25, page 55]. Start by considering the function

$$\mathsf{dpth} : \mathsf{Exp}_g \to \mathbb{N}$$

that measures the 'depth' of guarded expressions:

$$\begin{aligned}
\mathsf{dpth}(\underline{\emptyset}) = \mathsf{dpth}(i(\epsilon \mid a)) = \mathsf{dpth}(i \downarrow b) &= 0 \\
\mathsf{dpth}(\gamma_1 \oplus \gamma_2) &= 1 + \max(\mathsf{dpth}(\gamma_1), \mathsf{dpth}(\gamma_2)) \\
\mathsf{dpth}(\mu x.\gamma) &= 1 + \mathsf{dpth}(\gamma)
\end{aligned}$$

Clearly every expression $\gamma \in \mathsf{Exp}_g$ has finite depth. So to conclude the proof we just need to show that the following inequalities hold.

$$\mathsf{dpth}(\gamma_1 \oplus \gamma_2) > \mathsf{dpth}(\gamma_1), \qquad \mathsf{dpth}(\gamma_1 \oplus \gamma_2) > \mathsf{dpth}(\gamma_2), \qquad \mathsf{dpth}(\mu x.\gamma) > \mathsf{dpth}(\gamma[\mu x.\gamma/x])$$

The first two cases are obvious. The third case is a direct consequence of the equation

$$\mathsf{dpth}(\gamma) = \mathsf{dpth}(\gamma[\epsilon/y])$$

where $\epsilon \in \mathsf{Exp}$ is a closed expression and $y \in X$ a variable. This can be easily shown by induction on the structure of guarded expressions. □

Recall that the functor $\mathrm{Id}^I_{\mathsf{Cmd}} : \mathsf{Set} \to \mathsf{Set}$ admits a final coalgebra

$$\overline{\langle \mathrm{sng}, \mathrm{apd} \rangle} : \mathsf{Cmd}^{I^+} \to \left(\mathsf{Cmd} \times \mathsf{Cmd}^{I^+}\right)^I$$

Moreover, note that if I is the singleton set we have a bijection $\mathsf{Cmd}^{I^+} \cong \mathsf{Cmd}^\omega$. Hence, for a finite set I, every expression $\epsilon \in \mathsf{Exp}$ can be mapped into its behaviour $[\![\epsilon]\!] \in \mathsf{Cmd}^{I+}$, and, if $I = 1$, we have $[\![\epsilon]\!] \in \mathsf{Cmd}^{I^+} \cong \mathsf{Cmd}^\omega$.

Consider the composite of maps,

$$\mathrm{beh} : \ \mathsf{Exp} \xrightarrow{[\![\_]\!]} \mathsf{Cmd}^{1^+} \xrightarrow{\ \cong\ } \mathsf{Cmd}^\omega$$

We will use it to compute the stream associated with the expression from Example 3.8.

Example 3.11. Recall the bouncing ball described in Example 2.3 and the associated expression (Example 3.8)

$$\mu x.\ (x \mid \mathsf{a}) \ \oplus \ (\dot{\mathsf{p}} = \mathsf{v} \wedge \dot{\mathsf{v}} = \mathsf{g},\ \mathsf{p} \geq 0)$$

The associated stream is simply $((\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{b}), \dots)$ where b is the pair $(\dot{\mathsf{p}} = \mathsf{v} \wedge \dot{\mathsf{v}} = \mathsf{g},\ \mathsf{p} \geq 0)$. To see why, abbreviate the expression above into $\mu x.\psi$. The associated stream is then obtained by unfolding,

$$\begin{aligned}
\mathrm{beh}(\mu x.\ \psi) &= ((\mathsf{a}, \bot) \vee (\bot, \mathsf{b})) : \mathrm{beh}(\mu x.\ \psi \ \oplus \ \underline{\emptyset}) \\
&= (\mathsf{a}, \mathsf{b}) : \mathrm{beh}(\mu x.\ \psi \ \oplus \ \underline{\emptyset}) \\
&= ((\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{b}) \vee \bot) : \mathrm{beh}(\mu x.\ \psi \ \oplus \ \underline{\emptyset} \ \oplus \ \underline{\emptyset}) \\
&= ((\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{b})) : \mathrm{beh}(\mu x.\ \psi \ \oplus \ \underline{\emptyset} \ \oplus \ \underline{\emptyset}) \\
&= \dots \\
&= ((\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{b}) \dots)
\end{aligned}$$

Given the set of expressions Exp, the next natural step is to provide a correspondence between modes of reactive hybrid automata and expressions. For this, observe that every reactive hybrid automaton can be interpreted, without loss of information, as a finite $\mathrm{Id}^I_{\mathsf{Cmd}}$-coalgebra.

Now consider a finite $\mathrm{Id}^I_{\mathsf{Cmd}}$-coalgebra $M \to (M \times \mathsf{Cmd})^I$ with $M = \{m_1, \dots, m_n\}$. Then for every mode $m_l \in M$ define the expression

$$A^0_l = \mu m_l. \bigoplus_{i \in I} i((m_l)_i \mid a_l) \ \oplus \ i \downarrow b_l$$

where $(a_l, b_l) = m_l[i]$. Moreover, define $A_l^{k+1} = A_l^k\{A_{k+1}^k/x_{k+1}\}$ with $k \in \{0, \ldots, n-1\}$ and where $\{A_{k+1}^k/x_{k+1}\}$ denotes substitution without renaming the free variables in A_{k+1}^k that become bound due to the substitution. Finally, define $\epsilon_l = A_l^n$. Intuitively, this construction eliminates free variables at each iteration, starting with m_1 and ending with m_n.

Proposition 3.12. *Consider an expression A_l^k with $k \in \{1, \ldots, n\}$. All variables $m_1 \leq m \leq m_k$ are closed in A_l^k.*

Proof. The proof proceeds by induction. First, it is easy to see that the variable m_1 is closed in A_l^1, due to the equation $A_l^1 = A_l^0\{A_1^0/m_1\}$. So now consider an expression A_l^k. By assumption, all variables $m_1 \leq m \leq m_{k-1}$ are closed in the expressions A_k^{k-1} and A_l^{k-1}, and m_k is closed in A_k^{k-1} by definition. Since the equation

$$A_l^k = A_l^{k-1}\{A_k^{k-1}/m_k\}$$

holds, all variables $m_1 \leq m \leq m_k$ are closed in A_l^k. □

Example 3.13. Recall Example 2.4, which describes the behaviour of a water tank system. We will compute the expression relative to the left mode of the associated hybrid automaton. For this, let a denote the tuple $(\mathsf{t}' = 0, \mathsf{t} \geq \mathsf{c})$, and $\mathsf{b_1}, \mathsf{b_2}$ denote, respectively, the tuples

$$\left(\dot{\mathsf{l}} = 2 \wedge \dot{\mathsf{t}} = 1, \mathsf{t} \leq \mathsf{c}\right), \qquad \left(\dot{\mathsf{l}} = 0 \wedge \dot{\mathsf{t}} = 1, \mathsf{t} \leq \mathsf{c}\right)$$

We then compute,

$$\begin{aligned} A_1^0 &= \mu m_1.(m_2 \mid \mathsf{a}) \oplus \mathsf{b_1} & A_2^1 &= A_2^0\{A_1^0/m_1\} \\ A_2^0 &= \mu m_2.(m_1 \mid \mathsf{a}) \oplus \mathsf{b_2} & A_1^2 &= A_1^0\{A_2^1/m_2\} \\ A_1^1 &= A_1^0\{A_1^0/m_1\} = A_1^0 & A_2^2 &= A_2^1\{A_2^1/m_2\} = A_2^1 \end{aligned}$$

Hence, the left mode of the hybrid automaton (*i.e.* A_1^2) corresponds to the expression below.

$$\mu m_1.\ (\epsilon \mid \mathsf{a}) \oplus \mathsf{b_1}, \qquad \epsilon = \mu m_2.\ (\mu m_1.\ (m_2 \mid \mathsf{a}) \oplus \mathsf{b_1} \mid \mathsf{a}) \oplus \mathsf{b_2}$$

We showed how to generate expressions from modes of reactive hybrid automata. A further result to pursue is to show that a given mode and the corresponding expression are coalgebraically bisimilar.

Remark 3.14. First note that for every expression $\epsilon \in \mathsf{Exp}$ we have

$$\epsilon \sim \epsilon \oplus \underline{\emptyset}$$

One can then generate a bisimulation equivalence R from the set of pairs $\{(\epsilon, \epsilon \oplus \underline{\emptyset}) \mid \epsilon \in \mathsf{Exp}\}$. As discussed in Proposition 2.26, this induces an obvious quotient map $[\_] : \mathsf{Exp} \twoheadrightarrow \mathsf{Exp}/R$ and corresponding coalgebra

$$\mathsf{Exp}/R \to (\mathsf{Exp}/R \times \mathsf{Cmd})^I$$

that make the diagram below commute.

$$\begin{array}{ccc} \mathsf{Exp} & \xrightarrow{\quad [\_] \quad} & \mathsf{Exp}/R \\ {\scriptstyle\delta}\downarrow & & \downarrow \\ (\mathsf{Exp}\times\mathsf{Cmd})^I & \xrightarrow[([\_]\times\mathrm{id})^I]{} & (\mathsf{Exp}/R\times\mathsf{Cmd})^I \end{array}$$

In order to avoid a burdened notation, we will use the coalgebra of expressions $\delta : \mathsf{Exp} \to (\mathsf{Exp}\times\mathsf{Cmd})^I$ as if it were the 'quotient' coalgebra obtained by this process.

Theorem 3.15. *(Kleene theorem I) Let $m_l \in M$ be the mode of a reactive hybrid automaton (M, c) and ϵ_l the associated expression. These states are coalgebraically bisimilar,* i.e. *$m_l \sim \epsilon_l$.*

Proof. We will show that the map $f : M \to \mathsf{Exp}$ that associates each mode with its expression makes the diagram below commute.

$$\begin{array}{ccc} M & \xrightarrow{\quad f \quad} & \mathsf{Exp} \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle\delta} \\ (M\times\mathsf{Cmd})^I & \xrightarrow[(f\times\mathrm{id})^I]{} & (\mathsf{Exp}\times\mathsf{Cmd})^I \end{array}$$

After this we simply apply Proposition 2.25 to finish the proof.

Clearly for every input $i \in I$, the equation $m_l[i] = \epsilon_l[i]$ holds. Therefore, it remains to show that equation $(m_l)_i = (\epsilon_l)_i$ also holds. So we reason,

$$\begin{aligned} & (A_l^n)_i \\ = \ & ((\mu m_l.\psi_l)\{A_1^0/m_1\}\ldots\{A_n^{n-1}/m_n\})_i \\ & \{\text{Definition of } (\_)_i \text{ and definition of } \psi_l\} \\ = \ & (\psi_l\{A_1^0/m_1\}\ldots\{A_{l-1}^{l-2}/m_{l-1}\}\{A_{l+1}^{l}/m_{l+1}\}\ldots\{A_n^{n-1}/m_n\})_i[A_l^n/m_l] \\ & \{\text{Definition of } \psi_l \text{ and } A_l^n \text{ has no free variables}\} \\ = \ & (\psi_l\{A_1^0/m_1\}\ldots\{A_{l-1}^{l-2}/m_{l-1}\}[A_l^n/m_l]\{A_{l+1}^{l}/m_{l+1}\}\ldots\{A_n^{n-1}/m_n\})_i \\ & \{A_l^n \text{ has no free variables}\} \\ = \ & (\psi_l\{A_1^0/m_1\}\ldots\{A_{l-1}^{l-2}/m_{l-1}\}\{A_l^n/m_l\}\{A_{l+1}^{l}/m_{l+1}\}\ldots\{A_n^{n-1}/m_n\})_i \\ & \{(*)\} \\ = \ & (\psi_l\{A_1^0/m_1\}\ldots\{A_{l-1}^{l-2}/m_{l-1}\}\{A_l^{l-1}/m_l\}\{A_{l+1}^{l}/m_{l+1}\}\ldots\{A_n^{n-1}/m_n\})_i \\ & \{\text{Definition of } (\_)_i \text{ and definition of } \psi_l\} \end{aligned}$$

$$= A_k^{k-1}\{A_{k+1}^k/x_{k+1}\}\dots\{A_n^{n-1}/x_n\}$$
$$= A_k^n$$

The step $(*)$ relies on the equality

$$\varphi\{B\{C_1/y_1\}\dots\{C_n/y_n\}/x\}\{C_1/y_1\}\dots\{C_n/y_n\} = \varphi\{B/x\}\{C_1/y_1\}\dots\{C_n/y_n\}$$

which always holds if φ does not bind the variables $y_1, \dots, y_n$. Actually, note that A_k^{k-1} can only bind the variables $m_1 \leq m \leq m_k$. □

Let us now see how to generate a reactive hybrid automaton from an expression $\epsilon \in \mathsf{Exp}$ (the former being here interpreted as an $\mathrm{Id}^I_{\mathsf{Cmd}}$-coalgebra and *vice-versa*). Actually, we already know that there exists a smallest subcoalgebra $(\overline{\epsilon}, \delta)$ of (Exp, δ) whose carrier contains the expression ϵ. Unfortunately, as discussed in [25], this coalgebra is not always finite. To overcome this we need to further quotient the coalgebra

$$\delta : \mathsf{Exp} \to (\mathsf{Exp} \times \mathsf{Cmd})^I$$

with the following equivalences.

$$(\epsilon_1 \oplus \epsilon_2) \oplus \epsilon_3 \equiv \epsilon_1 \oplus (\epsilon_2 \oplus \epsilon_3)$$
$$\epsilon_1 \oplus \epsilon_2 \equiv \epsilon_2 \oplus \epsilon_1$$
$$\epsilon \oplus \epsilon \equiv \epsilon$$

Again, we have a quotient map $[\_] : \mathsf{Exp} \twoheadrightarrow \mathsf{Exp}/R$ and a coalgebra $\delta : \mathsf{Exp}/R \to (\mathsf{Exp}/R \times \mathsf{Cmd})^I$ that make the diagram below commute, where, as before, the symbol δ is overloaded in a slight abuse of notation.

$$\begin{array}{ccc} \mathsf{Exp} & \xrightarrow{\quad [\_] \quad} & \mathsf{Exp}/R \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\delta} \\ (\mathsf{Exp} \times \mathsf{Cmd})^I & \xrightarrow[([\_]\times \mathrm{id})^I]{} & (\mathsf{Exp}/R \times \mathsf{Cmd})^I \end{array}$$

We need to prove that for every expression $\epsilon \in \mathsf{Exp}$ the subcoalgebra $(\overline{[\epsilon]}, \delta)$ of $(\mathsf{Exp}/R, \delta)$ is finite. For this, we will show that there exists a finite set $\mathrm{C}(\epsilon)$ that contains $[\epsilon]$ and that forms a subcoalgebra of $(\mathsf{Exp}/R, \delta)$. By the definition of smallest subcoalgebra, the subcoalgebra $(\overline{[\epsilon]}, \delta)$ must thus be finite.

Definition 3.16. Consider an expression $\epsilon \in \mathsf{Exp}$. Let $\mathrm{cl}(\epsilon)$ be the smallest set generated by the following equations, with a slight abuse of notation in the last one

$$\begin{aligned} \mathrm{cl}(x) &= \{x\} & \mathrm{cl}(i \downarrow b) &= \{i \downarrow b\} \\ \mathrm{cl}(\underline{\emptyset}) &= \{\underline{\emptyset}\} & \mathrm{cl}(\epsilon_1 \oplus \epsilon_2) &= \mathrm{cl}(\epsilon_1) \cup \mathrm{cl}(\epsilon_2) \cup \{\epsilon_1 \oplus \epsilon_2\} \\ \mathrm{cl}(i(\epsilon \mid a)) &= \mathrm{cl}(\epsilon) \cup \{i(\epsilon \mid a)\} & \mathrm{cl}(\mu x.\gamma) &= \mathrm{cl}(\gamma)[\mu x.\gamma/x] \cup \{\mu x.\gamma\} \end{aligned}$$

Define

$$\mathrm{C}(\epsilon) = \{\, [\epsilon_1 \oplus \cdots \oplus \epsilon_k] \mid \epsilon_1, \ldots, \epsilon_k \in \mathrm{cl}(\epsilon), \text{ and } \epsilon_1, \ldots, \epsilon_k \text{ all distinct} \,\}$$

with $[\underline{\emptyset}]$ as the empty sum.

Theorem 3.17. *For every $\epsilon \in \mathsf{Exp}$ the associated set $\mathrm{C}(\epsilon)$ is finite and forms a subcoalgebra of $(\mathsf{Exp}/R, \delta)$.*

Proof. Since the set $\mathrm{cl}(\epsilon)$ is finite the set $\mathrm{C}(\epsilon)$ must be finite as well. In order to show that the set $\mathrm{C}(\epsilon)$ forms a subcoalgebra of $(\mathsf{Exp}/R, \delta)$ we will prove that

$$\text{if } [\kappa] \in \mathrm{C}(\epsilon) \text{ then } [\kappa_i] \in \mathrm{C}(\epsilon)\,.$$

This result follows by induction on the structure of closed expressions. In particular,

- $[\underline{\emptyset}_i] = [(a \downarrow b)_i] = [\underline{\emptyset}] \in \mathrm{C}(\epsilon)$, by definition of $\mathrm{C}(\epsilon)$.
- If $[i(\psi \mid a)] \in \mathrm{C}(\epsilon)$ then $i(\psi \mid a) \in \mathrm{cl}(\epsilon)$ and therefore $[\psi] \in \mathrm{C}(\epsilon)$;
- If $[\kappa_1 \oplus \kappa_2] \in \mathrm{C}(\epsilon)$ then by the induction hypothesis $[\kappa_{1i}], [\kappa_{2i}] \in \mathrm{C}(\epsilon)$. We may assume that $\kappa_{1i} = [\psi_1 \oplus \cdots \oplus \psi_k]$, $\kappa_{2i} = [\psi'_1 \oplus \cdots \oplus \psi'_l]$ and see that $[\psi_1 \oplus \cdots \oplus \psi_k \oplus \psi'_1 \oplus \cdots \oplus \psi'_l] \in \mathrm{C}(\epsilon)$.
- If $[\mu x.\gamma] \in \mathrm{C}(\epsilon)$ then, by definition of $\mathrm{cl}(\epsilon)$, $[\gamma[\mu x.\gamma/x]] \in \mathrm{C}(\epsilon)$. Using the induction hypothesis $[(\gamma[\mu x.\gamma/x])_i] \in \mathrm{C}(\epsilon)$.

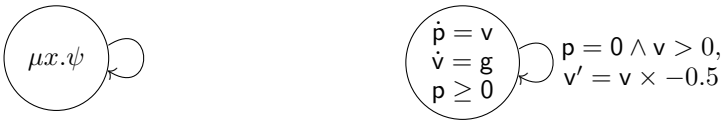
□

Theorem 3.18. *(Kleene Theorem II) Let $\epsilon \in \mathsf{Exp}$ be an expression. Then there is a finite $\mathrm{Id}^I_{\mathsf{Cmd}}$-coalgebra $(\overline{[\epsilon]}, \delta)$ with $[\epsilon] \sim \epsilon$.*

Example 3.19. Recall Example 2.3, which describes the behaviour of a bouncing ball. Recall also the associated expression

$$\mu x.\, (x \mid \mathsf{c}) \;\oplus\; (\dot{\mathsf{p}} = \mathsf{v} \wedge \dot{\mathsf{v}} = \mathsf{g},\ \mathsf{p} \geq 0)$$

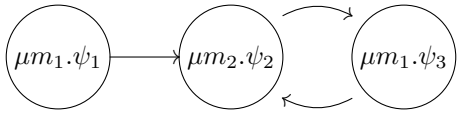
Abbreviating this expression to $\mu x.\psi$, the subcoalgebra $(\overline{[\mu x.\psi]}, \delta)$ is the hybrid automaton below on the left, or more explicitly, the reactive hybrid automaton below on the right.



Example 3.20. Let us now recall Example 2.4 which describes the behaviour of a water tank system using a hybrid automaton. In Example 3.13 we saw that the expression below specifies the left mode of the automaton.

$$\mu m_1.\ (\epsilon \mid \mathsf{a}) \oplus \mathsf{b}_1, \qquad \epsilon = \mu m_2.\ (\mu m_1.\ (m_2 \mid \mathsf{a}) \oplus \mathsf{b}_1 \mid \mathsf{a}) \oplus \mathsf{b}_2$$

To keep the notation simple, abbreviate expression $\mu m_1.\ (\epsilon \mid \mathsf{a}) \oplus \mathsf{b}_1$ to $\mu m_1.\ \psi_1$, expression ϵ to $\mu m_2.\ \psi_2$, and expression $\mu m_1.\ (m_2 \mid \mathsf{a}) \oplus \mathsf{b}_1$ to $\mu m_1.\psi_3$. The subcoalgebra $(\overline{[\mu m_1.\psi_1]}, \delta)$ is the hybrid automaton below.



The attentive reader may notice that in Example 2.4 we specified the water tank system using a hybrid automaton with two modes, but somehow we ended up with three here. This is because the subcoalgebra $(\overline{[\mu m_1.\psi_1]}, \delta)$ is not necessarily minimal. There exist however, several works on minimisation of coalgebras (*e.g.* [26, 24, 27, 28]) that can be used to obtain a better result. For example, it is well-known that every $\mathrm{Id}^I_{\mathsf{Cmd}}$-coalgebra (X, c) has a minimal coalgebra (Y, d) which results from the factorisation of the universal map $X \to \mathsf{Cmd}^{I^+}$, as illustrated in the diagram below (see *e.g.* [24]).

$$\begin{array}{ccccc} X & \twoheadrightarrow & Y & \rightarrowtail & \mathsf{Cmd}^{I^+} \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle \overline{\langle \mathrm{sng}, \mathrm{apd} \rangle}} \\ (X \times \mathsf{Cmd})^I & \longrightarrow & (Y \times \mathsf{Cmd})^I & \longrightarrow & (\mathsf{Cmd}^{I^+} \times \mathsf{Cmd})^I \end{array}$$

The subcoalgebra $(\overline{[\mu m_1.\psi_1]}, \delta)$ can therefore be turned into a minimal coalgebra. For this particular case, one just needs to show that

$$\mu m_1.\ \psi_1 \sim \mu m_1.\ \psi_3$$

and collapse both modes by identifying them (recall Proposition 2.26).

3.2. A coalgebraic semantics

Recall that a reactive hybrid automaton is a finite $\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}}$-coalgebra (M, c). This entails that for each mode $m \in M$ there is an associated behaviour $[\![m]\!] \in (\mathsf{Tr} \times \mathsf{Ev})^{I^+}$. However, calling an element of the set $(\mathsf{Tr} \times \mathsf{Ev})^{I^+}$ 'behaviour' may be somewhat misleading, as the elements of $\mathsf{Tr} \times \mathsf{Ev}$ are basically syntactic

constructs — for this reason we say that $[\![(m)]\!]$ is the ***syntactic behaviour*** of m.

Actually, the observable behaviour of hybrid systems traditionally comprises continuous evolutions intercalated with variable resets. So, building on our previous work [12], the current subsection shows how to systematically compute such a behaviour from reactive hybrid automata. More generally, assuming the four conditions in Assumptions 2.5, we prove the existence of a functor between a category of reactive hybrid automata and a category that suitably captures their semantics in terms of continuous evolutions and discrete jumps. We then study some properties of this functor, compare it with the traditional semantics of hybrid automata, and use it in the analysis of hybrid systems.

Definition 3.21. [12, 14] Let O be a topological space and $U : \mathsf{Top} \to \mathsf{Set}$ the forgetful functor from Top to Set. Then define $\mathcal{H}O$ as the set

$$U\left(\coprod_{d\in\mathbb{R}_{\geq 0}} O^{[0,d]}\right)$$

where $[0, d]$ is equipped with the subspace topology induced by the Euclidean one. Intuitively, the set $\mathcal{H}O$ is the sum of all continuous evolutions with a finite duration.

Note that for every topological space O there is a map $\mathsf{lst} : \mathcal{H}O \to UO$ that returns the last point of a given evolution.

Remark 3.22. Since reactive hybrid automata accommodate a notion of input, the conditions in Assumptions 2.5 need to be slightly generalised so that this dimension can be taken into account. In most cases the generalisation is quite straightforward, except perhaps for case of condition (2) in Assumptions 2.5, which we will present below.

Consider a reactive hybrid automaton with a finite set of real-valued variables $X = \{x_1, \ldots, x_n\}$ and denote the corresponding set of valuations $\mathbb{R}^n (\cong \mathbb{R}^X)$ by V. As discussed in Assumptions 2.5 (1), we may assume the existence of a map

$$\mathsf{sol} : M \times V \times I \times \mathbb{R}_{\geq 0} \to V$$

The assumption of time-determinism ensures that for each pair $(m, v, i) \in M \times V \times I$ there exists a continuous map

$$\mathsf{sol}(m, v, i, -) : \mathbb{R}_{\geq 0} \to V$$

whose domain can be restricted to a specific interval $[0, d] \subseteq \mathbb{R}_{\geq 0}$. This canonically induces a map

$$\overline{\mathsf{sol}} : M \times V \times I \to \mathcal{H}V$$

Whenever no ambiguities arise we drop the overline in $\overline{\mathsf{sol}}$. Moreover, the assumption on deterministic assignments induces a map

$$\mathsf{jmp} : M \times V \times I \to M \times V$$

defined as

$$\mathsf{jmp}(m,v,i) = (E(m,i), \mathsf{asg}(m,i)(u))\,, \qquad u = \mathsf{lst} \cdot \mathsf{sol}\,(m,v,i)$$

Note that deterministic assignments can be seen as maps $V \to V$. Therefore, with a slight abuse in notation, every pair $(m,i) \in M \times I$ corresponds to a function $\mathsf{asg}(m,i) : V \to V$.

Definition 3.23. Consider a reactive hybrid automaton (M,c). A state $(m,v) \in M \times V$ is valid if

$$v \models \mathsf{inv}(m,i) \qquad (i \in I)$$

Finally, we generalise Assumptions 2.5 (2).

Definition 3.24. A reactive hybrid automaton (M,c) has no invalid jumps if the map

$$Z \times I \rightarrowtail M \times V \times I \xrightarrow{\mathsf{jmp}} M \times V$$

can always be factorised through the obvious inclusion map $Z \rightarrowtail M \times V$. Diagrammatically,

$$\begin{array}{ccc} M \times V \times I & \xrightarrow{\mathsf{jmp}} & M \times V \\ \uparrow & & \uparrow \\ Z \times I & \dashrightarrow & Z \end{array}$$

We will refer to the generalisation above of Assumptions 2.5 (2) and the obvious generalisations of Assumptions 2.5 (1,3,4) as ***generalised Assumptions 2.5***.

We are now ready to introduce a functorial semantics for reactive hybrid automata. For this, let $\mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ be the full subcategory of $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ whose objects respect the generalised Assumptions 2.5.

Proposition 3.25. *Consider two coalgebras $(M,c),(N,d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ and two modes $m \in M, n \in N$ such that $m \sim n$. Then we have*

$$(m,v) \in Z_{(M,c)} \equiv (n,v) \in Z_{(N,d)}$$

Proof.

$$\begin{aligned} & (m,v) \in Z_{(M,c)} \\ \equiv\ & v \models \mathsf{inv}(m,i) \quad (i \in I) && \text{(Definition of } Z_{(M,c)}) \\ \equiv\ & v \models \mathsf{inv}(n,i) \quad (i \in I) && (m \sim n) \\ \equiv\ & (n,v) \in Z_{(N,d)} && \text{(Definition of } Z_{(N,d)}) \end{aligned}$$

□

Theorem 3.26. *There exists a 'semantics' functor*

$$\mathcal{S} : \mathsf{SimpHat}\left(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}}\right) \to \mathsf{CoAlg}\left(\mathrm{Id}^I_{\mathcal{H}V}\right)$$

defined by the mappings

$$\mathcal{S}(M,\overline{c}) = (Z, \overline{\mathcal{S}c}), \qquad \mathcal{S}f = f \times \mathrm{id}$$

and the commuting diagram below.

$$\begin{array}{ccc} M\times V\times I & \xrightarrow{\langle \mathsf{jmp},\mathsf{sol}\rangle} & M\times V\times \mathcal{H}V \\ \uparrow & & \uparrow \\ Z\times I & \xrightarrow[\mathcal{S}c]{} & Z\times \mathcal{H}V \end{array}$$

Proof. The proof is a direct consequence of Theorem 4.7 below. □

Intuitively, every reactive hybrid automaton $(M,c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$ induces an $\mathrm{Id}^I_{\mathcal{H}V}$-coalgebra

$$\overline{\langle \mathsf{jmp},\mathsf{sol}\rangle} : Z \to (Z\times\mathcal{H}V)^I$$

that encodes the former's behaviour: each state $z \in Z$ and input $i \in I$ give rise to an observable, continuous evolution (an element of $\mathcal{H}V$) and an internal, discrete transition to the next state. Let us illustrate this idea with a few examples. To keep them simple and illustrative assume that $I = 1$.

Example 3.27. Recall the tank-and-valve system described in Example 2.4 and denote the corresponding hybrid automaton by $(M,\overline{c}) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$. We then have $\mathcal{S}c : Z \to Z\times\mathcal{H}V$ defined as

$$\mathcal{S}c\,(m_1,l,t) = ((m_2, l+2\mathsf{c}, 0), f), \quad \mathcal{S}c\,(m_2,l,t) = ((m_1,l,0),g)$$

with the functions $f,g : [0,\mathsf{c}] \to \mathbb{R}^2$ given by

$$f(r) = (l+2r,\ t+r), \quad g(r) = (l,\ t+r).$$

The maps $f,g : [0,\mathsf{c}] \to \mathbb{R}^2$ encode the evolution of the water level and time.

Example 3.28. Consider again the bouncing ball from Example 2.3. Denoting the corresponding hybrid automaton by $(M,\overline{c}) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$, we have $\mathcal{S}c : Z \to Z\times\mathcal{H}V$ given by

$$\mathcal{S}c\,(m,p,v) = \big((m,0,v'), mov(p,v,-)\big)$$

where the value v' corresponds to the (abrupt) change of velocity due to the collision with the ground, function $mov(p,v,-) : [0,d] \to \mathbb{R}$ describes the ball's movement and velocity between jumps, and d denotes the time that the ball takes to reach the ground from the state (p,v). In symbols,

$$v' = (v+\mathsf{g}d)\times -0.5, \quad mov(p,v,t) = (p+vt+\tfrac{1}{2}\mathsf{g}t^2, v+gt), \quad d = \tfrac{\sqrt{2\mathsf{g}p+v^2}+v}{\mathsf{g}}.$$

Example 3.29. Suppose also that one is able to change the dampening factor of the ball at each bounce. This gives rise to an obvious reactive hybrid automaton

$$(M, c) : M \to (M \times \mathsf{Tr} \times \mathsf{Ev})^I$$

with I (now not necessarily 1) as the set of possible dampening factors. We then have

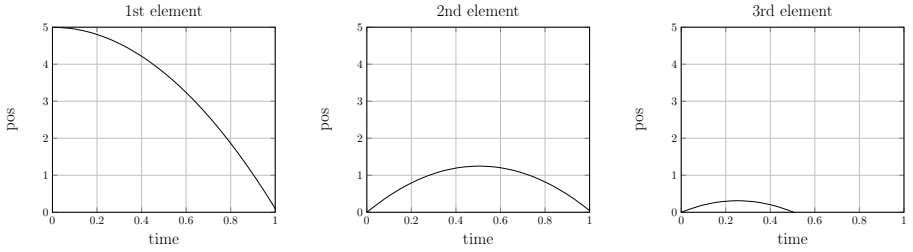
$$\mathcal{S}c\,(m, p, v, i) = \big((m, 0, v'), mov(p, v, -)\big)$$

where the value d and the map $mov(p, v, -) : [0, d] \to \mathbb{R}$ are defined as before, and the change of the ball's velocity after a collision is expressed by the equation below.

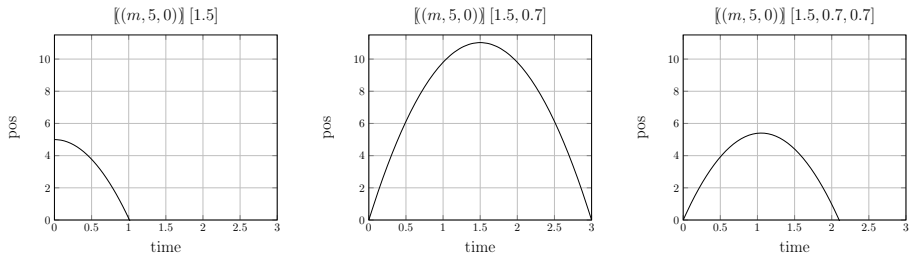
$$v' = (v + \mathsf{g}d) \times -i$$

As mentioned in the previous subsection, each $\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}}$-coalgebra (X, c) with $I = 1$ yields a map $\mathrm{beh} : Z \to (\mathcal{H}V)^{1^+} \cong (\mathcal{H}V)^\omega$ which for a given $z \in Z$ computes the stream of (observable) continuous evolutions $[\![z]\!]$, as illustrated in the following example.

Example 3.30. Consider again the bouncing ball system discussed in Examples 2.3 and 3.28. Hiding the evolutions concerning the ball's velocity, to keep our illustration simple, the first three elements of $\mathrm{beh}(m, 5, 0)$ are shown in the following plots.



Example 3.31. Recall from Example 3.29 the bouncing ball that allows to change its dampening factor at each bounce and the corresponding reactive hybrid automaton $(M, \overline{c})$. We can use the universal map $[\![\_]\!] : Z \to (\mathsf{Tr} \times \mathsf{Ev})^{I^+}$ to compute the behaviour of this system. For example, expressions $[\![(m, 5, 0)]\!]\,[1.5]$, $[\![(m, 5, 0)]\!]\,[1.5, 0.7]$, and $[\![(m, 5, 0)]\!]\,[1.5, 0.7, 0.7]$ yield the following sequence of plots.



Interestingly, and in contrast to the traditional semantics of hybrid automata (Definition 2.6), the coalgebraic semantics, illustrated in the examples above, puts a clear frontier between the discrete domain and the continuous one: the elements of $\mathcal{H}V$ live in the continuous domain whereas the structures $(\mathcal{H}V)^{\omega}$ and $(\mathcal{H}V)^{I^+}$ clearly possess a discrete nature.

Moreover, recall that the coalgebraic semantics was obtained in a completely canonical manner and, as discussed in the previous section, bears a well-known relationship with coalgebraic bisimulation. We will take advantage of this knowledge in the ensuing section.

3.3. Bisimulation

Recall that the coalgebraic definition of bisimulation (see Definition 2.19) is parametrised by a functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$. Thus, taking the functor $\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}} : \mathsf{Set} \to \mathsf{Set}$, a relation $R \subseteq X \times Y$ between two $\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}}$-coalgebras (or two reactive hybrid automata) (X, c) and (Y, d) is a bisimulation if there is a third coalgebra (R, r) that makes the diagram below commute.

$$\begin{array}{ccccc} X & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Y \\ {\scriptstyle c}\downarrow & & {\scriptstyle r}\downarrow & & \downarrow{\scriptstyle d} \\ (X \times (\mathsf{Tr} \times \mathsf{Ev}))^I & \xleftarrow[(\pi_1 \times \mathrm{id})^I]{} & (R \times (\mathsf{Tr} \times \mathsf{Ev}))^I & \xrightarrow[(\pi_2 \times \mathrm{id})^I]{} & (Y \times (\mathsf{Tr} \times \mathsf{Ev}))^I \end{array}$$

In other words, the relation $R \subseteq X \times Y$ is a bisimulation if $x \, R \, y$ entails

$$x[i] = y[i] \text{ and } x_i \, R \, y_i \qquad (i \in I)$$

In the context of hybrid automata, however, this notion of bisimulation (which we call ***syntactic bisimulation***) is too strong. For example, in the bouncing ball system (Example 2.3) to replace the differential equation $\dot{\mathsf{p}} = \mathsf{v}$ by $\dot{\mathsf{p}} = \mathsf{v} + 0$ yields a totally different hybrid automaton in the sense that the mode of the original one is not coalgebraically bisimilar to the mode of the modified version. Nevertheless, we know that at the semantic level both modes will produce exactly the same behaviour.

The reason is that the notion of bisimulation above is anchored to the syntactic level rather than to the semantic one. Fortunately, we know that the

category $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathcal{H}V})$ also carries a notion of bisimulation (see Definition 2.19) and that there exists a functor

$$\mathcal{S} : \mathsf{SimpHat}\left(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}}\right) \to \mathsf{CoAlg}\left(\mathrm{Id}^I_{\mathcal{H}V}\right)$$

that maps each reactive hybrid automaton to its corresponding model. Thus,

Definition 3.32. Consider two reactive hybrid automata $(M,c),(N,d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$. A relation $R \subseteq Z_{(M,c)} \times Z_{(N,d)}$ is a ***semantic bisimulation*** (or simply, a bisimulation if no ambiguities occur) for $\mathcal{S}(M,c)$ and $\mathcal{S}(N,d)$ iff $z_1\, R\, z_2$ entails

$$\mathsf{sol}_{(M,c)}(z_1,i) = \mathsf{sol}_{(N,d)}(z_2,i), \qquad \mathsf{jmp}_{(M,c)}(z_1,i)\; R\; \mathsf{jmp}_{(N,d)}(z_2,i) \qquad (i \in I)$$

The following properties involve the semantics functor and the two types of bisimulation for reactive hybrid automata discussed above.

Proposition 3.33. *Consider two reactive hybrid automata $(X,c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$, $(Y,d) \in \mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$ and two modes $x \in X$, $y \in Y$ that are coalgebraically bisimilar $x \sim y$. Then the mode y respects the generalised Assumptions 2.5.*

Proof. We analyse each assumption in detail.

1. We need to show that $\mathsf{dyn}(y,i)$ has exactly one solution. For this, note that $x \sim y$ and therefore $\mathsf{dyn}(x,i) = \mathsf{dyn}(y,i)$ for every $i \in I$. Hence, by assumption, $\mathsf{dyn}(y,i)$ has exactly one solution.
2. In order to see that for every $i \in I$ the assignment $\mathsf{asg}(y,i)$ is deterministic proceed as above.
3. We focus now on time-determinism. The duration associated with each pair $(x,v) \in Z_{(X,c)}$ and input $i \in I$ depends solely on the values $\mathsf{dyn}(x,i)$, $\mathsf{grd}(x,i)$, and $\mathsf{inv}(x,i)$. By assumption, these three values must be equal to $\mathsf{dyn}(y,i)$, $\mathsf{grd}(y,i)$, and $\mathsf{inv}(y,i)$, respectively. Moreover, the condition

   $$(x,v) \in Z_{(X,c)} \equiv (y,v) \in Z_{(Y,d)}$$

   holds. For every pair $(y,v) \in Z_{(Y,d)}$ the state $(x,v) \in Z_{(X,c)}$ is time-deterministic which then entails that (y,v) is time-deterministic as well.
4. Finally, a similar argument shows that there are no invalid jumps.

□

Corollary 3.34. *Consider two reactive hybrid automata $(X,c) \in \mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$ and $(Y,d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$. If there is a morphism $(X,c) \to (Y,d)$ in $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$ then $(X,c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$.*

Corollary 3.35. *Consider two reactive hybrid automata $(X,c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$, $(Y,d) \in \mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$. If there is surjective morphism $(X,c) \twoheadrightarrow (Y,d)$ in $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$ then $(Y,d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr}\times\mathsf{Ev}})$.*

Theorem 3.36. *Consider two reactive hybrid automata* $(M, c), (N, d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ *two modes* $m \in M$, $n \in N$ *that are coalgebraically bisimilar* $m \sim n$ *and a state* $(m, v) \in Z_{(M,c)}$. *The property* $(m, v) \sim (n, v)$ *holds.*

Proof. Assume that $m \sim n$. We then have a span

$$\begin{array}{ccc} & (R,r) & \\ {\scriptstyle \pi_1}\swarrow & & \searrow{\scriptstyle \pi_2} \\ (M,c) & & (N,d) \end{array}$$

in the category $\mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ by Proposition 3.25 and Corollary 3.34. The functor

$$\mathcal{S} : \mathsf{SimpHat}\left(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}}\right) \to \mathsf{CoAlg}\left(\mathrm{Id}^I_{\mathcal{H}V}\right)$$

maps it into the span

$$\begin{array}{ccc} & \mathcal{S}(R,r) & \\ {\scriptstyle \pi_1 \times \mathrm{id}}\swarrow & & \searrow{\scriptstyle \pi_2 \times \mathrm{id}} \\ \mathcal{S}(M,c) & & \mathcal{S}(N,d) \end{array}$$

in $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$. Applying Proposition 2.25, we obtain

$$(m, v) \sim ((m, n), v) \sim (n, v)$$

□

Theorem 3.37. *(Kleene Theorem III) Let* $(M, c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ *be a reactive hybrid automaton, consider a mode* $m \in M$ *and associated expression* ϵ. *Then the diagram below commutes*

$$\begin{array}{ccc} M & \xrightarrow{\quad f \quad} & \mathsf{Exp} \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle \delta} \\ (M \times \mathsf{Cmd})^I & \xrightarrow[(f \times \mathrm{id})^I]{} & (\mathsf{Exp} \times \mathsf{Cmd})^I \end{array}$$

and $f(m) = \epsilon$. *Moreover, we have the factorisation*

$$\begin{array}{ccccc} M & \xrightarrow{\quad f \quad} & f[M] & \rightarrowtail & \mathsf{Exp} \\ {\scriptstyle c}\downarrow & & \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle \delta} \\ (M \times \mathsf{Cmd})^I & \xrightarrow[(f \times \mathrm{id})^I]{} & (f[M] \times \mathsf{Cmd})^I & \longrightarrow & (\mathsf{Exp} \times \mathsf{Cmd})^I \end{array}$$

and the following conditions hold: $\epsilon \in f[M]$, $(f[M], d) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$. *Finally, for every state* $(m, v) \in Z$ *we have*

$$(m, v) \sim (\epsilon, v)$$

Proof. Direct application of Corollary 3.35 and Theorem 3.36. □

Corollary 3.38. *Let* $(M, c) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ *be a reactive hybrid automaton. For every state* $(m, v) \in Z$ *the following property holds*

$$(m, v) \sim ([\![m]\!]_{(M,c)}, v)$$

Intuitively, this last corollary states that to compute the behaviour of a specific mode one can either move directly to $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathcal{H}V})$ and compute its behaviour there, or first compute its syntactic behaviour $[\![m]\!]_{(M,c)}$ and then move to $\mathsf{CoAlg}(\mathrm{Id}^I_{\mathcal{H}V})$ to compute the behaviour of $[\![m]\!]_{(M,c)}$.

Using standard coalgebraic definitions, we introduced two notions of bisimulation for reactive hybrid automata, one at the syntactic level and other at the semantic one. However, the standard notion of bisimulation for hybrid automata differs from these two. In the next section we show how to frame coalgebraically the standard notion as well.

4. Going generic

4.1. The general picture

In the previous section we studied reactive hybrid automata from a coalgebraic perspective. We saw that their discrete (or internal) transition type reflects a deterministic setting and showed that discrete transitions can be seen as state transitions of a computational device. Digital controllers, however, produce far more complex behaviours than what the deterministic setting allows for, often combining nondeterministic, or probabilistic features, among other things. Naturally, this calls for variations in the definition of hybrid automata and, consequently, for a more general coalgebraic semantics than the one we have pursued in the paper so far.

Therefore, in this section we consider $\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr})$-coalgebras typed as

$$\overline{\langle \mathrm{nxt}, \mathrm{out} \rangle} : M \to (\mathsf{F}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

where the functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ determines a discrete transition type and the set I denotes an input set. Technically, such arrows can be decomposed into

$$\mathrm{nxt} : M \times I \to \mathsf{F}(M \times \mathsf{Tr}), \qquad \mathrm{out} : M \times I \to \mathsf{Ev}$$

This makes clear that variations in the functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ correspond to variations in relation E and functions asg and grd (recall Definition 2.2). As we will see in the following subsection, such variations dictate how a system (discretely) jumps to a next state.

Table 1 lists functors $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ and associated variants of hybrid automata. Some of the latter are already well known (*e.g.* the nondeterministic case in row 4, if $I = 1$), and others are new (*e.g.* the replicating case in row 3). This illustrates the high level of genericity that coalgebras bring to the theory of hybrid automata: specific types of automata are captured in specific instantiations of $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ and global constructions and results are defined parametric on F once and for all.

Coalgebra	Functor F	Behaviour
$M \to (M \times \mathsf{Tr} \times \mathsf{Ev})^I$	$\mathrm{Id}\, X = X$	Deterministic [12, 9]
$M \to (\mathcal{M}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\mathcal{M}\, X = X + 1$	Faulty
$M \to (\Delta(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\Delta\, X = X \times X$	Replicating
$M \to (\mathcal{P}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\mathcal{P}\, X = \{A \subseteq X\}$	Nondeterministic [5]
$M \to (\mathcal{D}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\mathcal{D}\, X \subseteq \{\mu \in [0,1]^X \mid \mu[X] = 1\}$	Probabilistic [11]
$M \to (\mathcal{P}\mathcal{D}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\mathcal{P}\mathcal{D}$ —	Segala [11]
$M \to (\mathcal{W}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$	$\mathcal{W}\, X \subseteq S^X$. S is a semiring.	Weighted [29]

Table 1: Possible variants for F.

Faulty and replicating behaviour

The Maybe functor $\mathcal{M} : \mathsf{Set} \to \mathsf{Set}$ (second row) brings faulty behaviour into the scene by giving rise to finite coalgebras

$$M \to (\mathcal{M}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

which we call ***faulty hybrid automata***. Intuitively, these automata can terminate an execution at a discrete transition, as a response, for instance, to a program exception or loss of information.

The diagonal functor $\Delta : \mathsf{Set} \to \mathsf{Set}$ yields finite coalgebras typed as

$$M \to (\Delta(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

These behave like reactive hybrid automata (see Section 3), but a discrete transition forces a jump to two different places at the same time. The intuition is that such systems replicate themselves at each discrete transition. For example, in this context the bouncing ball would turn into two at each bounce.

From a strict computer science point of view this kind of behaviour may seem strange, but in other areas it is quite common: *e.g.* in biology, cells indeed replicate when a specific saturation point is reached. We call this variant of hybrid automata ***replicating hybrid automata***

Nondeterministic, probabilistic, and Segala behaviour

The powerset functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ leads to finite coalgebras typed as

$$M \to (\mathcal{P}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

which clearly subsume classic hybrid automata (see Section 2). Next, the functor $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ leads to ***Markov hybrid automata***, or more concretely finite coalgebras typed as

$$M \to (\mathcal{D}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

Similarly to probabilistic hybrid automata (see Section 2), they allow to consider computational devices with probabilistic features. On the same note, observe that finite coalgebras typed as

$$M \to (\mathcal{PD}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

subsume classic probabilistic hybrid automata.

Weighted behaviour

Consider a semiring S and a functor $\mathcal{W} : \mathsf{Set} \to \mathsf{Set}$ such that for every set X and map $f : X \to Y$ the following equations hold.

$$\mathcal{W}X = \left\{\mu \in S^X \mid \mathsf{supp}(\mu) \text{ finite}\right\}, \qquad \mathcal{W}f(\mu)(y) = \sum_{x \in f^{-1}(y)} \mu(x)$$

This functor yields finite coalgebras typed as

$$M \to (\mathcal{W}(M \times \mathsf{Tr}) \times \mathsf{Ev})^I$$

which we call ***weighted hybrid automata***. Such automata decorate each edge, associated guard and assignment, with a cost. This is often a relevant feature which has been discussed in the hybrid systems' domain several times (*e.g.* [29, 30]). In the case of the bouncing ball, for example, one may need to consider the cost of throwing the ball up.

4.2. A generic semantics

We will now introduce a generic semantics for hybrid automata that is parametric on the type of their discrete transitions. We start by mentioning some useful properties.

Proposition 4.1. *Every functor* $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ *has two natural transformations*

$$\tau : \mathsf{F} \times \mathrm{Id} \to \mathsf{F}(\mathrm{Id} \times \mathrm{Id}), \qquad \upsilon : \mathrm{Id} \times \mathsf{F} \to \mathsf{F}(\mathrm{Id} \times \mathrm{Id})$$

defined as

$$\tau_{X,Y}(a, b) = \mathsf{F}(\_, b)\,(a), \qquad \upsilon_{X,Y}(a, b) = \mathsf{F}(a, \_)\,(b) \qquad (X, Y \in \mathsf{Set})$$

Proposition 4.2. *Consider a functor* $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$*, a non-empty set* X*, and an injective map* $f : X \rightarrowtail Y$*. The map* $\mathsf{F}f : \mathsf{F}X \to \mathsf{F}Y$ *is injective as well.*

Proof. See for example [13]. □

We will also use the natural transformation

$$\alpha : (\mathrm{Id} \times \mathrm{Id}) \times \mathrm{Id} \to \mathrm{Id} \times (\mathrm{Id} \times \mathrm{Id})$$

with $\mathrm{Id} : \mathsf{Set} \to \mathsf{Set}$ as the identity functor over Set.

Let us focus on the conditions enumerated in Assumptions 2.5, in particular the notion of valid state and the notion of invalid jump.

Definition 4.3. Consider an F-hybrid automaton (M, c). We say that a state $(m, v) \in M \times V$ is valid if the condition below holds.

$$v \models \mathsf{inv}(m, i) \qquad (i \in I)$$

Consider an $\mathsf{F}_{\mathsf{Ev}}^{I} \cdot (\_ \times \mathsf{Tr})$-coalgebra $(M, \overline{\langle \mathrm{nxt}, \mathrm{out} \rangle})$. It induces a map

$$\mathsf{jmp} : M \times V \times I \to \mathsf{F}(M \times V)$$

given by the commuting square below.

$$\begin{array}{ccc} \mathsf{F}(M \times \mathsf{Tr}) \times V & \xrightarrow{\mathsf{F}\alpha \cdot \tau_{M,\mathsf{Tr},V}} & \mathsf{F}(M \times (\mathsf{Tr} \times V)) \\ {\scriptstyle \langle \mathrm{nxt} \cdot (\pi_1 \times \mathrm{id}), \mathsf{lst} \cdot \mathsf{sol} \rangle} \uparrow & & \downarrow {\scriptstyle \mathsf{F}(\mathrm{id} \times \mathsf{ev})} \\ (M \times V) \times I & \xrightarrow[\mathsf{jmp}]{} & \mathsf{F}(M \times V) \end{array}$$

where $\mathsf{ev} : \mathsf{Tr} \times V \to V$ is the map that given an assignment, a guard, and a value $v \in V$, it returns the result of the application of the assignment to v. Note that in the definition of jmp we are implicitly using the assumptions that impose deterministic assignments, unique solutions, and time-determinism.

Definition 4.4. An F-hybrid automaton (M, c) has no invalid jumps if the map

$$Z \times I \rightarrowtail M \times V \times I \xrightarrow{\mathsf{jmp}} M \times V$$

can always be factorised through the obvious map $\mathsf{F}Z \rightarrowtail \mathsf{F}(M \times V)$. Diagrammatically,

$$\begin{array}{ccc} M \times V \times I & \xrightarrow{\mathsf{jmp}} & \mathsf{F}(M \times V) \\ \uparrow & & \uparrow \\ Z \times I & \dashrightarrow & \mathsf{F}Z \end{array}$$

At first sight the function $\mathsf{ev} : \mathsf{Tr} \times V \to V$ may seem rather strange, because it applies the assignment given as input independently of the guard that is also given as input. This is actually a consequence of time-determinism (see Assumptions 2.5 (4)). Recall that, according to this assumption, at the end of the duration associated with a state all corresponding guards must be enabled. For example, in the case of replicating hybrid automata this notion of time-determinism tells that at the end of the duration of a state the two corresponding guards must be enabled; and similarly for reactive hybrid automata. In the case of Markov hybrid automata time-determinism tells that at the end of the duration of a state all guards in the support of the associated distribution must be enabled.

Remark 4.5. In some settings such a notion of time-determinism is too strong, a prime example being the non-deterministic case ($\mathcal{P}$). In the conclusions we will also show that one may consider instead a map

$$\mathsf{ev} : \mathsf{Tr} \times V \to \mathsf{F}V$$

for a discrete transition type $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$. This allows to relax the condition on time-determinism and deterministic assignments, but at the cost of putting assumptions on the functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ that are not always met.

Let $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ be a functor and denote by $\mathsf{SimpHat}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ the full subcategory of $\mathsf{CoAlg}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ whose objects respect the generalised Assumptions 2.5.

Remark 4.6. Take a coalgebra $(M, \overline{c}) \in \mathsf{SimpHat}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$. Note that the solution map $\mathsf{sol} : Z \to \mathcal{H}V$ can be seen as a composite of maps as shown by the diagram below

$$\begin{array}{ccccc} (M \times I) \times V & & \xrightarrow{c \times \mathrm{id}} & & (\mathsf{F}(M \times \mathsf{Tr}) \times \mathsf{Ev}) \times V \\ \uparrow{\scriptstyle \langle \pi_1 \times \mathrm{id}, \pi_2 \cdot \pi_1 \rangle} & & & & \downarrow{\scriptstyle (\mathsf{F}\pi_2 \times \mathrm{id}) \times \mathrm{id}} \\ Z \times I & \xrightarrow[\mathsf{sol}]{} & \mathcal{H}V & \xleftarrow[\mathsf{evo}]{} & \mathsf{F}\,\mathsf{Tr} \times \mathsf{Ev} \times V \end{array}$$

where $\mathsf{evo} : \mathsf{F}\,\mathsf{Tr} \times \mathsf{Ev} \times V \to \mathcal{H}V$ is the function that calculates evolutions from triples in $\mathsf{F}\,\mathsf{Tr} \times \mathsf{Ev} \times V$. Recall that the latter is given by the assumption on unique solutions and the assumption on time-determinism.

Theorem 4.7. *There exists a semantics functor*

$$\mathcal{S} : \mathsf{SimpHat}\left(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr})\right) \to \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}V}\right)$$

defined by the mappings

$$\mathcal{S}\,(M, \overline{c}) = \left(Z, \overline{\mathcal{S}c}\right), \qquad \mathcal{S}f = f \times \mathrm{id}$$

and the commuting diagram below

$$\begin{array}{ccc} M \times V \times I & \xrightarrow{\langle \mathsf{jmp}, \mathsf{sol} \rangle} & F(M \times V) \times \mathcal{H}V \\ \uparrow & & \uparrow \\ Z \times I & \xrightarrow[\mathcal{S}c]{} & FZ \times \mathcal{H}V \end{array}$$

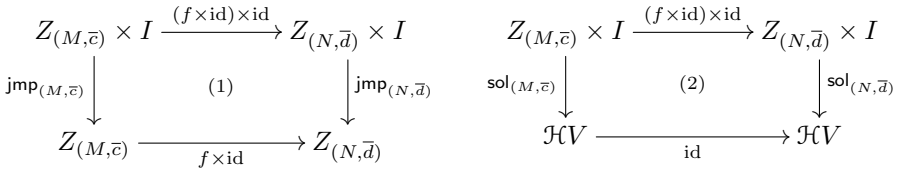
where $\mathsf{F}Z \times \mathcal{H}V \rightarrowtail \mathsf{F}(M \times V) \times \mathcal{H}V$ *is the obvious inclusion map.*

Proof. It is straightforward to show that the construction above preserves identity maps and distributes over composition. So it remains to show that it sends morphisms in $\mathsf{SimpHat}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ to morphisms in $\mathsf{CoAlg}(\mathsf{F}^I_{\mathcal{H}V})$.

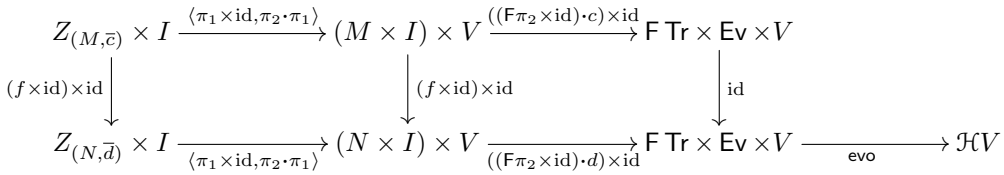
Assume that we have a morphism $f : (M, \overline{c}) \to (N, \overline{d})$ in $\mathsf{SimpHat}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$. We will show that the diagram

$$\begin{array}{ccc} Z_{(M,\overline{c})} \times I & \xrightarrow{(f \times \mathrm{id}) \times \mathrm{id}} & Z_{(N,\overline{d})} \times I \\ \downarrow{\scriptstyle \mathcal{S}c} & & \downarrow{\scriptstyle \mathcal{S}d} \\ Z_{(M,\overline{c})} \times \mathcal{H}V & \xrightarrow[(f \times \mathrm{id}) \times \mathrm{id}]{} & Z_{(N,\overline{d})} \times \mathcal{H}V \end{array}$$

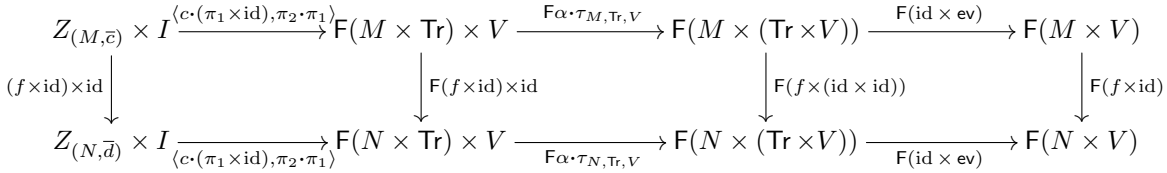
commutes by showing that the diagrams below commute.



Recall Remark 4.6. Then, note that, by assumption, the diagram below commutes. This entails the commutativity of the diagram above in the right (2).



Finally, by assumption and by naturality, the diagram below must commute



This entails the commutativity of the diagram above in the left (1). □

Consider a hybrid automaton $(M, c) \in \mathsf{SimpHat}(\mathsf{F}_{\mathsf{Ev}}^{I} \cdot (\_ \times \mathsf{Tr}))$ and a pair $((m, v), i) \in Z \times I$. We denote the map $\langle \underline{m}, \mathsf{sol}(m, v, i) \rangle : [0, d] \to Z$ simply by $\mathsf{ev}_{(m,v,i)} : [0, d] \to Z$.
The following two theorems show that the generic semantics defined above generalises the semantics of both classic and probabilistic hybrid automata.

Theorem 4.8. *Let $(M, c) \in \mathsf{SimpHat}(\mathcal{P}_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ be a classic hybrid automaton and consider two states $z_1, z_2 \in Z$. Then the following equivalences hold.*

$$z_1 \xrightarrow{l} z_2 \equiv \mathsf{ev}_{z_1}(l) = z_2 \qquad (l \in [0, d])$$
$$\mathsf{lst} \cdot \mathsf{sol}(z_1) \xrightarrow{*} z_2 \equiv z_2 \in \mathsf{jmp}(z_1)$$

Proof. Follows from Definition 2.6 and the definition of the map $\mathsf{jmp} : (M \times V) \times I \to \mathsf{F}(M \times V)$. □

Theorem 4.9. *Let $(M, c) \in \mathsf{SimpHat}(\mathcal{PD}_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ be a probabilistic hybrid automaton and consider two states $z_1, z_2 \in Z$. Then the following equivalences hold.*

$$z_1 \xrightarrow{l} \delta_{z_2} \equiv \mathsf{ev}_{z_1}(l) = z_2 \qquad (l \in [0, d])$$
$$\mathsf{lst} \cdot \mathsf{sol}(z_1) \xrightarrow{*} \mu_2 \equiv \mu_2 \in \mathsf{jmp}(z_1)$$

Proof. Follows from Definition 2.12 and the definition of the map $\mathsf{jmp} : (M \times V) \times I \to \mathsf{F}(M \times V)$. □

4.3. Φ-bisimulation coalgebraically

In this subsection we generalise the definition of Φ-bisimulation (see Section 2) so that it becomes parametric on a type of discrete transition as captured by $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$. This provides a notion of Φ-bisimulation for F-hybrid automata and thus allows to study the notion across several variants of hybrid automata in a uniform manner.

In the generalisation process we use the notions of ***colouring*** and ***up-to bisimulation***, two common concepts in the theory of coalgebras.

Definition 4.10. Let $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ be a functor. Then consider an $\mathsf{F}^I_{\mathcal{H}V}$-coalgebra $(X, \overline{\langle \mathrm{nxt}, \mathrm{out} \rangle})$ and a map $c : X \to C$, which, intuitively, associates states to colours in C. There exists an $\mathsf{F}^I_{\mathcal{H}(C\times V)}$-coalgebra $(X, \overline{\langle \mathrm{nxt}, \mathrm{out}^\dagger \rangle})$ with the map $\mathrm{out}^\dagger : X \times I \to \mathcal{H}(C \times V)$ given by the commuting diagram below.

$$\begin{array}{ccc} X \times I & \xrightarrow{\langle c \cdot \pi_1, \mathrm{out} \rangle} & C \times \mathcal{H}V \\ & \searrow^{\mathrm{out}^\dagger} & \downarrow{\scriptstyle \upsilon_{C,V}} \\ & & \mathcal{H}(C \times V) \end{array}$$

The map $\upsilon_{C,V} : C \times \mathcal{H}V \to \mathcal{H}(C \times V)$ is defined as $\tau_{C,V}\,(a, f) = \langle \underline{a}, f \rangle$ where $\underline{a}$ is the constant map over a.

Consider an $\mathsf{F}^I_{\mathcal{H}V}$-coalgebra (X, c). We denote by $(X, c^\dagger)$ the $\mathsf{F}^I_{\mathcal{H}(C\times V)}$-coalgebra that is built from (X, c) as shown in the previous definition with $\mathrm{id} : X \to X$ as the colouring map.

Note that there exists an obvious 'projection' functor

$$\Pi : \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}(C\times V)}\right) \to \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}V}\right)$$

Remark 4.11. Denote by $\mathsf{CoAlg}(\mathsf{F}, X)$ the full subcategory of $\mathsf{CoAlg}(\mathsf{F})$ whose objects have the set X as carrier. We have a cospan of functors

$$\begin{array}{ccc} \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}V}, X\right) & & \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}(X\times V)}\right) \\ & \searrow \quad \swarrow_{\Pi} & \\ & \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}V}\right) & \end{array}$$

and the image of the projection functor contains the image of the inclusion functor. More concretely, for every coalgebra $(X, c) \in \mathsf{CoAlg}(\mathsf{F}^I_{\mathcal{H}V}, X)$ there exists the coalgebra $(X, c^\dagger) \in \mathsf{CoAlg}(\mathsf{F}^I_{\mathcal{H}(X\times V)})$ and $\Pi(X, c^\dagger) = (X, c)$.

The following theorem is well-known and helps to characterise a generic notion of Φ-bisimulation.

Theorem 4.12. *Let $\alpha : \mathsf{F} \to \mathsf{G}$ be a natural transformation between two functors $\mathsf{F}, \mathsf{G} : \mathsf{Set} \to \mathsf{Set}$. It induces a functor*

$$\alpha : \mathsf{CoAlg}(\mathsf{F}) \to \mathsf{CoAlg}(\mathsf{G})$$

defined as

$$\alpha(X, c) = (X, \alpha_X \cdot c), \qquad \alpha(f) = f$$

Moreover, the functor $\alpha : \mathsf{CoAlg}(F) \to \mathsf{CoAlg}(G)$ preserves bisimulations: i.e. *if R is a bisimulation for two F-coalgebras (X, c) and (Y, d) then R is a bisimulation for G-coalgebras $\alpha(X, c)$ and $\alpha(Y, d)$.*

Proposition 4.13. *Let $\Phi \subseteq S \times S$ be an equivalence relation over a set S and $q : S \twoheadrightarrow S/_\Phi$ the corresponding quotient map. There exists a natural transformation*

$$\zeta_\Phi : \mathsf{F}^I_{\mathcal{H}S} \to \mathsf{F}^I_{\mathcal{H}(S/_\Phi)}$$

defined as $\zeta_{\Phi X} = (\mathrm{id} \times (q\cdot))^I$ for every every set X.

Finally,

Definition 4.14. Consider an F-hybrid automaton $(M, c) \in \mathsf{SimpHat}(\mathsf{F}^I_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ and an equivalence relation $\Phi \subseteq Z \times Z$. A relation $R \subseteq Z \times Z$ is a ***coalgebraic Φ-bisimulation*** if it is a bisimulation for the coalgebra $\zeta_\Phi(\mathcal{S}(M, c)^\dagger)$.

We say that two states $z_1, z_2 \in Z$ are ***coalgebraically Φ-bisimilar*** if they are related by a coalgebraic Φ-bisimulation.

As discussed in Section 2, classic hybrid automata and probabilistic hybrid automata already carry a notion of Φ-bisimulation, detailed in Definition 2.8 and Definition 2.14. In the following lines we relate both definitions with the one above.

We start with non-deterministic hybrid automata.

Lemma 4.15. *Consider a hybrid automaton $(M, c) \in \mathsf{SimpHat}(\mathcal{P}_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ and a Φ-bisimulation $R \subseteq Z \times Z$. The relation $R \subseteq Z \times Z$ is a coalgebraic Φ-bisimulation as well.*

Proof. Assume that $x \, R \, y$, with $x = (x_1, x_2), y = (y_1, y_2)$. We will show that the following cases hold.

- The condition $\mathsf{ev}_x \ \Phi \ \mathsf{ev}_y$ holds.

  $x \, R \, y$

  $\Rightarrow \ \mathsf{ev}_x \ R \ \mathsf{ev}_y$ (Defn. of Φ-bisimulation)

  $\Rightarrow \ \mathsf{ev}_x \ \Phi \ \mathsf{ev}_y$ (Defn. of Φ-bisimulation)

- For every $x' \in \mathsf{jmp}(x)$ there is a $y' \in \mathsf{jmp}(y)$ such that $x' \, R \, y'$.

  $x \, R \, y, \;\; x' \in \mathsf{jmp}(x)$

  $\Rightarrow \; (x_1, \mathsf{lst} \cdot \mathsf{sol}(x)) \, R \, (y_1, \mathsf{lst} \cdot \mathsf{sol}(y))$ (Defn. of Φ-bisimulation)

  $\Rightarrow \; (x_1, \mathsf{lst} \cdot \mathsf{sol}(x)) \xrightarrow{*} x', \;\; (y_1, \mathsf{lst} \cdot \mathsf{sol}(y)) \xrightarrow{*} y'$

  and $x' \, R \, y'$ (Defn. of Φ-bisimulation and jmp)

  $\Rightarrow \; y' \in \mathsf{jmp}(y), \;\; x' \, R \, y'$ (Defn. of jmp)

- For every pair $y' \in \mathsf{jmp}(y)$ there is a pair $x' \in \mathsf{jmp}(x)$ such that $x' \, R \, y'$.

  Analogous to the previous case.

□

Interestingly, to assume that $R \subseteq Z \times Z$ is a coalgebraic Φ-bisimulation does not entail that R is a Φ-bisimulation. However, we can construct a relation $\overline{R} \subseteq Z \times Z$ such that $R \subseteq \overline{R}$ and $\overline{R}$ is Φ-bisimulation. This relation is defined as the smallest relation such that $R \subseteq \overline{R}$ and if $x \, R \, y$ then $\mathsf{ev}_x \;\overline{R}\; \mathsf{ev}_y$.

Lemma 4.16. *Consider a hybrid automaton $(M, c) \in \mathsf{SimpHat}(\mathcal{P}_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ and a coalgebraic Φ-bisimulation $R \subseteq Z \times Z$. Then the relation $\overline{R} \subseteq Z \times Z$ is a Φ-bisimulation.*

Proof. Assume that $x \, \overline{R} \, y$. We will show that the following cases hold.

- The condition $x \, \Phi \, y$ holds.

  $x \, \overline{R} \, y$

  $\Rightarrow \; \exists a, b \in Z, l \in [0, d]$
  $a \, R \, b \wedge \mathsf{ev}_a(l) = x \wedge \mathsf{ev}_b(l) = y$ (Defn. of $\overline{R}$)

  $\Rightarrow \; x \, \Phi \, y$ (Defn. of coalgebraic Φ-bisimulation)

- if $x \xrightarrow{l} x'$ then there exists a state y' such that $y \xrightarrow{l} y'$ and $x' \, \overline{R} \, y'$.

  In the case of an evolution step,

  $x \, \overline{R} \, y, \;\; x \xrightarrow{r} \mathsf{ev}_x(r)$

  $\Rightarrow \; \mathsf{ev}_x \;\overline{R}\; \mathsf{ev}_y, \;\; x \xrightarrow{r} \mathsf{ev}_x(r)$ (Defn. of $\overline{R}$)

  $\Rightarrow \; \mathsf{ev}_x \;\overline{R}\; \mathsf{ev}_y, \;\; y \xrightarrow{r} \mathsf{ev}_y(r)$ (Defn. of coalgebraic Φ-bisimulation)

  $\Rightarrow \; y \xrightarrow{r} \mathsf{ev}_y(r), \;\; \mathsf{ev}_x(r) \;\overline{R}\; \mathsf{ev}_y(r)$

In the case of a discrete transition,

$$
\begin{aligned}
& x \,\overline{R}\, y, \;\; x \xrightarrow{*} x' \\
\Rightarrow\; & \exists a, b \in Z, l \in [0, d] \\
& a \,R\, b \wedge \mathsf{ev}_a(l) = x \wedge \mathsf{ev}_b(l) = y && \text{(Defn. of } \overline{R}\text{)} \\
\Rightarrow\; & x' \in \mathsf{jmp}(a) && (\mathsf{jmp}(a) = \mathsf{jmp}(x)) \\
\Rightarrow\; & y' \in \mathsf{jmp}(b), \;\; x' \,R\, y' && \text{(Defn. of coalgebraic } \Phi\text{-bisimulation)} \\
\Rightarrow\; & y \xrightarrow{*} y', \;\; x' \,\overline{R}\, y' && (\mathsf{jmp}(b) = \mathsf{jmp}(y), \;\; R \subseteq \overline{R})
\end{aligned}
$$

- if $y \xrightarrow{l} y'$ then there exists a state x' such that $x \xrightarrow{l} x'$ and $x' \,\overline{R}\, y'$. Analogous to the previous case.

□

Then, using the two previous lemmas one obtains the following result.

Theorem 4.17. *Consider a hybrid automaton* $(M, c) \in \mathsf{SimpHat}(\mathcal{P}_{\mathsf{Ev}} \bullet (\_ \times \mathsf{Tr}))$ *and two states* $x, y \in Z$*. The equivalence below holds.*

$$x \sim^{\Phi} y \text{ iff } x \equiv^{\Phi} y$$

Proof. A direct consequence of Lemma 4.15 and Lemma 4.16. □

Let us now focus on probabilistic hybrid automata. Again in this context, to assume that $R \subseteq Z \times Z$ is a coalgebraic Φ-bisimulation does not entail that R is a Φ-bisimulation. Hence, like in the non-deterministic case, we will need to consider the construction $\overline{R}$.

Lemma 4.18. *Consider a hybrid automaton* $(M, c) \in \mathsf{SimpHat}(\mathcal{PD}_{\mathsf{Ev}} \bullet (\_ \times \mathsf{Tr}))$ *and a* Φ*-bisimulation* $R \subseteq Z \times Z$*. The relation* $R \subseteq Z \times Z$ *is a coalgebraic* Φ*-bisimulation as well.*

Proof. Assume that $x \,R\, y$, with $x = (x_1, x_2), y = (y_1, y_2)$. We will show that the following cases hold.

- The condition $\mathsf{ev}_x \;\Phi\; \mathsf{ev}_y$ holds.

$$
\begin{aligned}
& x \,R\, y \\
\Rightarrow\; & \mathsf{ev}_x \;R\; \mathsf{ev}_y && \text{(Defn. of } \Phi\text{-bisimulation)} \\
\Rightarrow\; & \mathsf{ev}_x \;\Phi\; \mathsf{ev}_y && \text{(Defn. of } \Phi\text{-bisimulation)}
\end{aligned}
$$

- $\mu_1 \in \mathsf{jmp}(x)$ entails that there is some $\mu_2 \in \mathsf{jmp}(y)$ such that $\mu_1 \asymp_R \mu_2$.

  $x \mathrel{R} y, \quad \mu_1 \in \mathsf{jmp}(x)$

  $\Rightarrow\ (x_1, \mathsf{lst} \cdot \mathsf{sol}(x)) \mathrel{R} (y_1, \mathsf{lst} \cdot \mathsf{sol}(y))$ (Defn. of Φ-bisimulation)

  $\Rightarrow\ (x_1, \mathsf{lst} \cdot \mathsf{sol}(x)) \xrightarrow{*} \mu_1, (y_1, \mathsf{lst} \cdot \mathsf{sol}(y)) \xrightarrow{*} \mu_2$

  and $\mu_1 \asymp_R \mu_2$ (Defn. of Φ-bisimulation and jmp)

  $\Rightarrow\ \mu_2 \in \mathsf{jmp}(y), \quad \mu_1 \asymp_R \mu_2$ (Defn. of jmp)

- $\mu_2 \in \mathsf{jmp}(y)$ entails that there is some $\mu_1 \in \mathsf{jmp}(x)$ such that $\mu_1 \asymp_R \mu_2$. Analogous to the previous case.

□

Lemma 4.19. *Consider a hybrid automaton $(M, c) \in \mathsf{SimpHat}(\mathcal{PD}_{\mathsf{Ev}} \cdot (\_ \times \mathsf{Tr}))$ and a coalgebraic Φ-bisimulation $R \subseteq Z \times Z$. Then the relation $\overline{R} \subseteq Z \times Z$ is a Φ-bisimulation.*

Proof. Assume that $x \mathrel{\overline{R}} y$. We will show that the following cases hold.

- The condition $x \mathrel{\Phi} y$ holds.

  $x \mathrel{\overline{R}} y$

  $\Rightarrow\ \exists a, b \in Z, l \in [0, d]$

  $a \mathrel{R} b \wedge \mathsf{ev}_a(l) = x \wedge \mathsf{ev}_b(l) = y$ (Defn. of $\overline{R}$)

  $\Rightarrow\ x \mathrel{\Phi} y$ (Defn. of coalgebraic Φ-bisimulation)

- If $x \xrightarrow{l} \mu_1$ then there exists a μ_2 such that $y \xrightarrow{l} \mu_2$ and $\mu_1 \asymp_{\overline{R}} \mu_2$.

  First consider an evolution step.

  Assume that $x \mathrel{\overline{R}} y$ and $x \xrightarrow{l} \delta_{\mathsf{ev}_x(l)}$. By the definition of $\overline{R}$ and coalgebraic Φ-bisimulation we have $y \xrightarrow{l} \delta_{\mathsf{ev}_y(l)}$. Hence it remains to show that condition $\mu_1 \asymp_{\overline{R}} \mu_2$ holds. For this consider the Dirac distribution δ of $(\mathsf{ev}_x(l), \mathsf{ev}_y(l))$. We will show that the three conditions below hold.

  1. $\delta(a, b) > 0$ entails $a \mathrel{\overline{R}} b$.

     $x \mathrel{\overline{R}} y, \quad \delta(a, b) > 0$

     $\Rightarrow\ \mathsf{ev}_x \mathrel{\overline{R}} \mathsf{ev}_y, \quad \delta(a, b) > 0$ (Defn. of $\overline{R}$)

     $\Rightarrow\ \mathsf{ev}_x \mathrel{\overline{R}} \mathsf{ev}_y, \quad a = \mathsf{ev}_x(l), \quad b = \mathsf{ev}_y(l)$ (Defn. of δ)

     $\Rightarrow\ a \mathrel{\overline{R}} b$

2. $\delta_{\mathsf{ev}_x(l)}(a) = \delta(\{a\} \times Z)$.
   If $a = \mathsf{ev}_x(l)$ then the equation below holds.

$$\delta_{\mathsf{ev}_x(l)}(a) = 1 = \delta(\{a\} \times Z)$$

   Otherwise, the equation below must be true.

$$\delta_{\mathsf{ev}_x(l)}(a) = 0 = \delta(\{a\} \times Z)$$

3. $\delta_{\mathsf{ev}_y(l)}(b) = \delta(Z \times \{b\})$.
   Analogous to the previous case.

Now assume that $x \,\overline{R}\, y$ and $x \xrightarrow{*} \mu_1$.

$$\begin{aligned}
& x \,\overline{R}\, y, \;\; x \xrightarrow{*} \mu_1 \\
\Rightarrow\; & \exists a, b \in Z, l \in [0, d] \\
& a \,R\, b \wedge \mathsf{ev}_a(l) = x \wedge \mathsf{ev}_b(l) = y && \text{(Defn. of } \overline{R}) \\
\Rightarrow\; & \mu_1 \in \mathsf{jmp}(a) && (\mathsf{jmp}(a) = \mathsf{jmp}(x)) \\
\Rightarrow\; & \mu_2 \in \mathsf{jmp}(b), \;\; \mu_1 \asymp_R \mu_2 && \text{(Defn. of coalgebraic } \Phi\text{-bisimulation)} \\
\Rightarrow\; & y \xrightarrow{*} \mu_2, \;\; \mu_1 \asymp_{\overline{R}} \mu_2 && (\mathsf{jmp}(b) = \mathsf{jmp}(y), \;\; R \subseteq \overline{R})
\end{aligned}$$

- If $y \xrightarrow{l} \mu_2$ then there exists a μ_1 such that $x \xrightarrow{l} \mu_1$ and $\mu_1 \asymp_{\overline{R}} \mu_2$. Analogous to the previous case.

□

Theorem 4.20. *Consider a probabilistic hybrid automaton* $(M, c) \in \mathsf{SimpHat}(\mathcal{PD}_{\mathsf{Ev}^\bullet}(\_ \times \mathsf{Tr}))$ *and two states* $x, y \in Z$*. The equivalence below holds.*

$$x \sim^{\Phi} y \text{ iff } x \equiv^{\Phi} y$$

Proof. Direct consequence of Lemma 4.18 and Lemma 4.19. □

Another interesting aspect to mention concerns reactive hybrid automata, studied in detail in the previous section, and the apparent absence of a suitable notion of Φ-bisimulation for them. However, instantiating Definition 4.14 with $\mathsf{F} = \mathrm{Id}$, we obtain a suitable notion of Φ-bisimulation for such automata as follows.

Definition 4.21. Consider a reactive hybrid automaton $(M, \overline{c}) \in \mathsf{SimpHat}(\mathrm{Id}^I_{\mathsf{Tr} \times \mathsf{Ev}})$ and an equivalence relation $\Phi \subseteq Z \times Z$ over its states. A relation $R \subseteq Z \times Z$ is a coalgebraic bisimulation if $z_1 \,R\, z_2$ entails

$$\mathsf{ev}_{(z_1,i)} \,\Phi\, \mathsf{ev}_{(z_2,i)}, \qquad \mathsf{jmp}(z_1, i) \;R\; \mathsf{jmp}(z_2, i) \quad (i \in I)$$

4.4. A hierarchy of hybrid automata

Natural transformations are an interesting mechanism to transform a coalgebra into another of a different transition type since naturality entails preservation of bisimilarity (see *e.g.* [22] and Theorem 4.12). Reflection of bisimilarity is also ensured if the natural transformation involved is injective (*i.e.* all of its components are injective) and the underlying functor in its domain preserves weak pullbacks (*cf.* [22]).

The latter condition (preservation of weak pullbacks) is quite mild. In fact, it holds for polynomial functors, the powerset functor, and the distribution functor respect it (*cf.* [22]). Therefore, in many cases checking for reflection reduces to checking for injectivity. Observe also that a natural transformation $\tau : \mathsf{F} \to \mathsf{G}$ induces another natural transformation

$$(\tau \times \mathrm{id})^I : \mathsf{F}^I_{\mathcal{H}V} \to \mathsf{G}^I_{\mathcal{H}V}$$

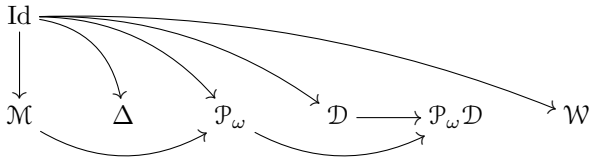
This means that an injective natural transformation $\tau : \mathsf{F} \to \mathsf{G}$ between two discrete transition types induces a functor

$$(\tau \times \mathrm{id})^I : \mathsf{CoAlg}\left(\mathsf{F}^I_{\mathcal{H}V}\right) \to \mathsf{CoAlg}\left(\mathsf{G}^I_{\mathcal{H}V}\right)$$

that bridges the semantics of F-hybrid automata to the semantics of G-hybrid automata and moreover the functor preserves and reflects bisimilarity given that the functor F preserves weak pullbacks. Therefore, such natural transformations can be used to establish formal relationships between different variants of hybrid automata, in particular they generate a hierarchy of hybrid automata in terms of their expressive power.

'To be more expressive' here means that looking at the model of an F-hybrid automaton as a $\mathsf{G}^I_{\mathcal{H}V}$-coalgebra (through the natural transformation $\tau : \mathsf{F} \to \mathsf{G}$) does not entail loss of observable information. In different words, if two states of an $\mathsf{F}^I_{\mathcal{H}V}$-coalgebra are bisimilar when looking at the latter as a $\mathsf{G}^I_{\mathcal{H}V}$-coalgebra, then they were already bisimilar before the application of τ (*i.e.* coalgebraic bisimilarity is reflected).

A possible hierarchy of hybrid automata is expressed in the diagram of injective natural transformations below.



All of them are quite well-known or can be easily derived, so we will just make a few remarks about absent transformations that the reader may be wondering about. Note that there is no injective natural transformation $\Delta \to \mathcal{P}_\omega$ as order is not preserved. Moreover observe that the obvious mapping $\mathcal{P}_\omega \to \mathcal{D}$ (which maps any finite set to the corresponding uniform distribution) does not respect naturality.

5. Conclusions and future work

Despite being the standard formalism for hybrid systems, the notion of hybrid automata frequently needs to be modified so that different types of computational behaviour can be taken into account. To tackle this issue, the current paper proposes a coalgebraic theory of hybrid automata where definitions and results are parametric on a discrete transition type $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$. We showed that within this theory suitable notions of bisimulation, behaviour, and state minimisation are easily derived. These can be studied across several variants of hybrid automata in a uniform manner and, consequently, results can be stated at a generic level, independently of whatever intricacies in each particular variant of hybrid automata one may encounter. As mentioned before, such an abstraction level is possible because our work is based on the theory of coalgebras.

Recall also that the latter promotes a black-box perspective in which discrete actions are hidden from the environment whereas continuous evolutions make up the observable behaviour. Interestingly, a somewhat dual perspective appears in document [17], where an object-oriented approach for hybrid systems is introduced: hybrid systems are viewed as coalgebras equipped with a monoid action (to represent time) that acts over the state space, forcing continuous evolutions to be hidden from the environment. This allows to consider physical processes that (continuously) evolve internally and that are possible to interact with at specific instants of time.

Adopting the same perspective than [17], the authors of document [31] study different notions of behavioural equivalence for timed systems using a coalgebraic approach and the notion of lax functor. They also consider different variants of these systems by assuming that their transition type is given by a monad that follows certain conditions.

We believe that our paper opens several research avenues. We detail some of them next along with related work.

Relaxing the assumptions

A natural research line pertains (the generalised) Assumptions 2.5, in particular the ones associated with deterministic assignments and time-determinism. The former can be dropped by assuming the existence of an 'evaluation' map (recall Remark 4.5)

$$\mathsf{ev} : \mathsf{Tr} \times V \to \mathsf{F}V$$

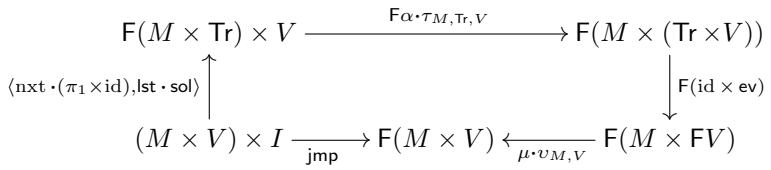
and by assuming that the functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ comes equipped with a natural transformation $\mu : \mathsf{FF} \to \mathsf{F}$.

Remark 5.1. All functors $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ that can be equipped with a monadic structure (*e.g.* $\mathcal{M}, \mathcal{P}, \mathcal{D}, \mathcal{W}$) have such a natural transformation.

Then proceed like before, defining the map

$$\mathsf{jmp} : M \times V \times I \to \mathsf{F}(M \times V)$$

as the composite in the diagram below.



Finally, similarly to Theorem 4.7, one can show that there exists a semantics functor

$$\mathcal{S} : \mathsf{SimpHat}\left(\mathsf{F}_{\mathsf{Ev}}^{I} \cdot (\_ \times \mathsf{Tr})\right) \to \mathsf{CoAlg}\left(\mathsf{F}_{\mathcal{H}V}^{I}\right)$$

but now without the assumption of deterministic assignments.

The case of time-determinism is much more complex. A simple approach would be to regard as the semantics of F-hybrid automata coalgebras typed as

$$S \to \mathsf{F}(S \times \mathcal{H}V)^{I}$$

and extend the theory developed in the paper accordingly. In many cases, however, this may be problematic, as the transition type would need to have a continuous nature. In other words, the functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ would need to behave well in the presence of infinite or uncountable possibilities in a jump of a hybrid automaton, which usually arise when the condition of time-determinism is dropped. Such is indeed not the case for the powerset functor and the distribution functor and it suggests a shift from the category Set to other categories where this kind of behaviour can be better handled. For example the category of topological spaces Top [32] and the category of Polish spaces Pol [33].

Bisimulation

Haghverdi *et al.* [34] resort to the notion of ***open map***, which is closely related to coalgebras, to provide an abstract notion of bisimulation for dynamical, control, and hybrid systems (the latter being understood as hybrid automata). However, variants of hybrid automata were not taken into consideration.

In our case, we showed that the coalgebraic perspective allows to systematically define and study notions of bisimulation for different variants of hybrid automata. Recall that we introduced 'syntactic', and 'semantic' bisimulation for reactive hybrid automata, and Φ-bisimulation parametrised by a discrete transition type $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$. An interesting next step is to study (and generalise) other notions of bisimulation, one prime example being ***approximate bisimulation*** for hybrid automata [35].

Languages for hybrid automata

We also used standard coalgebraic results [25] to derive a language for reactive hybrid automata and a corresponding Kleene-like theorem. Using again document [25] this result can be straightforwardly extended to a much more

general setting by allowing the discrete type functor $\mathsf{F} : \mathsf{Set} \to \mathsf{Set}$ to be, for example, polynomial, the powerset functor, the distribution functor, the functor of weights, or certain composites of these.

Moreover, the *op. cit.* can be used to produce axiomatisations for the derived languages. In many cases, however, the axioms may not be rich enough as they do not reflect the fact that differential equations themselves are part of a language and, therefore, that syntactic equality is a too fine grain equivalence notion for them. Actually, this problem is completely analogous to that which syntactic bisimulation carries and this suggests that we may follow a similar strategy to the one we employed for the latter case — this will be part of our main research for the near future.

Finally, note that there already exist a number of interesting works on regular expressions for subclasses of classic hybrid automata. For example, the document [36] introduces the so-called ***hybrid regular expressions*** that allow to describe possible executions of ***linear*** hybrid automata in terms of sequences of modes visited and respective durations of the visits. Such expressions, however, are not sufficiently rich to capture the behaviour of every linear hybrid automata and therefore a Kleene-like theorem cannot be established. On the other hand, document [37] establishes a Kleene-like theorem for ***timed automata***, which form an exiguous subclass of classic hybrid automata, but does not provide an axiomatisation.

Compositional operators

In document [14], we showed that a slightly different version of the continuous evolution space ($\mathcal{H}$) is a monad. This allows us to extend the theory developed in Section 3 with different kinds of compositional mechanisms in a straightforward manner. Recall that every monad generates a so-called ***Kleisli category***, the latter being usually a rich, powerful universe where different types of composition operators for a family of systems can be worked out. In the same note, we showed that $\mathcal{H}$'s Kleisli category carries a calculus with several forms of composition operators (*e.g.*, parallel, pipelining, sum), refinement techniques, and wiring mechanisms, as well as the corresponding algebraic laws. The systems in this Kleisli category cover reactive hybrid automata and therefore this calculus comes to them for free. Such a result opens up a number of possibilities. For example, we are currently studying the regular expressions detailed in Section 3 (Definition 3.6) combined with the composition operators provided by $\mathcal{H}$'s Kleisli category.

Acknowledgements

This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project POCI-01-0145-FEDER-016692 and by the PT-FLAD Chair on Smart

Cities & Smart Governance at Universidade do Minho. The first author is also sponsored by FCT grant SFRH/BD/52234/2013.

References

[1] P. Tabuada, Verification and Control of Hybrid Systems - A Symbolic Approach, Springer, 2009.

[2] R. Alur, Principles of Cyber-Physical Systems, MIT Press, 2015.

[3] A. Platzer, Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics, Springer, Heidelberg, 2010.

[4] R. Rajkumar, I. Lee, L. Sha, J. A. Stankovic, Cyber-physical systems: the next computing revolution, in: S. S. Sapatnekar (Ed.), Proceedings of the 47th Design Automation Conference, DAC 2010, Anaheim, California, USA, July 13-18, 2010, ACM, 2010, pp. 731–736.

[5] T. A. Henzinger, The theory of hybrid automata, in: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society, 1996, pp. 278–292.

[6] R. Alur, T. A. Henzinger, Modularity for timed and hybrid systems, in: A. W. Mazurkiewicz, J. Winkowski (Eds.), CONCUR '97, Vol. 1243 of Lecture Notes in Computer Science, Springer, 1997, pp. 74–88.

[7] O. Maler, Amir pnueli and the dawn of hybrid systems, in: K. H. Johansson, W. Yi (Eds.), Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010, ACM, 2010, pp. 293–295.

[8] K. Kim, P. R. Kumar, Cyber-physical systems: A perspective at the centennial, Proceedings of the IEEE 100 (Centennial-Issue) (2012) 1287–1308.

[9] J. Liu, X. Liu, T.-K. J. Koo, B. Sinopoli, S. Sastry, E. A. Lee, A hierarchical hybrid system model and its simulation, in: 38th IEEE Decision and Control, Vol. 4, IEEE, 1999, pp. 3508–3513.

[10] J. Sproston, Decidable model checking of probabilistic hybrid automata, in: M. Joseph (Ed.), Formal Techniques in Real-Time and Fault-Tolerant Systems, Vol. 1926 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2000, pp. 31–45.

[11] J. Sproston, Model checking of probabilistic timed and hybrid systems, Ph.D. thesis, School of Computer Science, University of Birmingham (2000).

[12] R. Neves, L. S. Barbosa, Hybrid automata as coalgebras, in: A. Sampaio, F. Wang (Eds.), Theoretical Aspects of Computing - ICTAC 2016 - 13th International Colloquium, Taipei, Taiwan, ROC, October 24-31, 2016, Proceedings, Vol. 9965 of Lecture Notes in Computer Science, 2016, pp. 385–402.

[13] J. Rutten, Universal coalgebra: a theory of systems, Theoretical Computer Science 249 (1) (2000) 3 – 80.

[14] R. Neves, L. S. Barbosa, D. Hofmann, M. A. Martins, Continuity as a computational effect, J. Log. Algebr. Meth. Program. 85 (5) (2016) 1057–1085.

[15] J. Adámek, H. Herrlich, G. E. Strecker, Abstract and Concrete Categories - The Joy of Cats, Dover Publications, 2009.

[16] J. Goubault-Larrecq, Non-Hausdorff Topology and Domain Theory—Selected Topics in Point-Set Topology, Vol. 22 of New Mathematical Monographs, Cambridge University Press, 2013.

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

[18] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems, Theoretical Computer Science 138 (1) (1995) 3–34.

[19] J. M. Davoren, On hybrid systems and the modal μ-calculus, in: P. J. Antsaklis, W. Kohn, M. D. Lemmon, A. Nerode, S. Sastry (Eds.), Hybrid Systems V, Vol. 1567 of Lecture Notes in Computer Science, Springer, 1997, pp. 38–69.

[20] S. Nadjm-Tehrani, Time-deterministic hybrid transition systems, in: P. J. Antsaklis, W. Kohn, M. D. Lemmon, A. Nerode, S. Sastry (Eds.), Hybrid Systems V, Vol. 1567 of Lecture Notes in Computer Science, Springer, 1997, pp. 238–250.

[21] J. Adamek, Introduction to coalgebra, Theory and Applications of Categories 14 (8) (2005) 157–199.

[22] A. Sokolova, Coalgebraic analysis of probabilistic systems, Ph.D. thesis, Technische Universiteit Eindhoven (2005).

[23] H. P. Gumm, T. Schröder, Coalgebras of bounded type, Mathematical Structures in Computer Science 12 (5) (2002) 565–578.

[24] B. Jacobs, Introduction to Coalgebra: Towards Mathematics of States and Observation, Vol. 59 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2016.

[25] A. Silva, Kleene coalgebra, Ph.D. thesis, Radboud Universiteit Nijmegen (2010).

[26] F. Bonchi, M. M. Bonsangue, H. H. Hansen, P. Panangaden, J. J. M. M. Rutten, A. Silva, Algebra-coalgebra duality in brzozowski's minimization algorithm, ACM Trans. Comput. Logic 15 (1) (2014) 3:1–3:29.

[27] N. Bezhanishvili, C. Kupke, P. Panangaden, Minimization via duality, in: C. L. Ong, R. J. G. B. de Queiroz (Eds.), Logic, Language, Information and Computation - 19th International Workshop, WoLLIC 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings, Vol. 7456 of Lecture Notes in Computer Science, Springer, 2012, pp. 191–205.

[28] J. Adámek, F. Bonchi, M. Hülsbusch, B. König, S. Milius, A. Silva, A Coalgebraic Perspective on Minimization and Determinization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 58–73.

[29] R. Alur, S. L. Torre, G. J. Pappas, Optimal paths in weighted timed automata, Theoretical Computer Science 318 (3) (2004) 297 – 322.

[30] P. Bouyer, Weighted timed automata: Model-checking and games, Electr. Notes Theor. Comput. Sci. 158 (2006) 3–17.

[31] T. Brengos, M. Peressotti, A uniform framework for timed automata, in: J. Desharnais, R. Jagadeesan (Eds.), 27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada, Vol. 59 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 26:1–26:15.

[32] D. Hofmann, R. Neves, P. Nora, Limits in categories of vietoris coalgebras, CoRR abs/1612.03318.

[33] E. Doberkat, Stochastic Coalgebraic Logic, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2009.

[34] E. Haghverdi, P. Tabuada, G. J. Pappas, Bisimulation relations for dynamical, control, and hybrid systems, Theoretical Computer Science 342 (2-3) (2005) 229–261.

[35] A. Girard, G. J. Pappas, Approximate bisimulation: A bridge between computer science and control theory, European Journal of Control 17 (5 - 6) (2011) 568 – 578.

[36] X. Li, T. Zheng, J. Hou, J. Zhao, G. Zheng, Hybrid regular expressions, in: T. A. Henzinger, S. Sastry (Eds.), Hybrid Systems: Computation and Control, First International Workshop, HSCC'98, Berkeley, California, USA, April 13-15, 1998, Proceedings, Vol. 1386 of Lecture Notes in Computer Science, Springer, 1998, pp. 384–399.

[37] E. Asarin, P. Caspi, O. Maler, A kleene theorem for timed automata, in: Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997, IEEE Computer Society, 1997, pp. 160–171.