Hybrid Automata as Coalgebras

Renato Neves(✉) and Luis S. Barbosa

HASLab (INESC TEC) & Universidade do Minho, Braga, Portugal
rjneves@inescporto.pt, lsb@di.uminho.pt

Abstract. Able to simultaneously encode discrete transitions and continuous behaviour, hybrid automata are the *de facto* framework for the formal specification and analysis of hybrid systems. The current paper revisits hybrid automata from a coalgebraic point of view. This allows to interpret them as state-based components, and provides a uniform theory to address variability in their definition, as well as the corresponding notions of behaviour, bisimulation, and observational semantics.

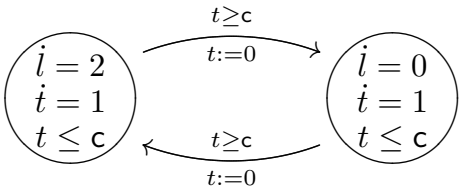
1 Introduction

1.1 Context

Consider a cruise control system. It comprises digital controllers, sensors, and actuators, that act in coordination to make the vehicle reach the intended speed. The system's behaviour, from an *external perspective*, is observed in the (continuous) evolution of a physical process (velocity). But at the same time we know that the controller, which has influence over this process, changes its *internal* state in a discrete manner.

Systems with this interaction pattern are often called *hybrid*. Their formal specification and analysis typically resorts to the theory of *hybrid automata* [Hen96], whose distinguishing feature is the ability of state variables to continuously evolve. This allows to express the evolution of physical processes, like movement, time, temperature, and pressure. In addition, there is syntactical machinery (guards, state invariants, and assignments) to facilitate the description of complex behaviour in a concise manner. For illustration purposes,

Example 1. Consider a (simplistic) system comprised of *a tank and a valve* connected to it. The valve allows water to flow in at a rate of 2 cm/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.



A. Sampaio and F. Wang (Eds.): ICTAC 2016, LNCS 9965, pp. 385–402, 2016.
DOI: 10.1007/978-3-319-46750-4_22

The variable l denotes the water level, which rises when the valve is open (differential equation $\dot{l} = 2$). Then, the differential equation $\dot{t} = 1$ defines the passage of time, which, along with invariant $t \leq \mathsf{c}$, forces the current state to be active for at most c seconds. On the other hand, the guards $t \geq \mathsf{c}$ and assignments $t := 0$ force the current state to be active at least c seconds before a switch. Finally, note that the guards $t \geq \mathsf{c}$ do not force transitions to happen, but only permit them. This means that if not for invariant $t \leq \mathsf{c}$, the valve could be open (or shut) indefinitely.

The semantics of hybrid automata is traditionally described in terms of labelled transition systems (LTS): each hybrid automaton yields an LTS whose edges encode both the discrete events and continuous evolutions (*cf.* [Hen96]). Edges in the latter are labelled by elements of $\mathbb{R}_{\geq 0}$ and reflect the difference of state variables with respect to the source and sink nodes. For example, denoting the left state (of the previous hybrid automaton) by m_1, the edge $(m_1, 1, 0.5) \xrightarrow{t} (m_1, 1 + 2t, 0.5 + t)$ exists in the underlying LTS iff $0.5 + t \leq \mathsf{c}$. This will be explained in more detail in Sect. 2.

For now, we emphasise that such a semantics 'collapses' both discrete assignments and continuous evolutions into the same relation, which makes difficult to distinguish the system's internal, thus hidden behaviour (typically its state changes), from what can be observed externally. Such a distinction, however, is at the very heart of the component-based paradigm, in which complex systems are verified through a suitable analysis of their (simpler) constituents (see *e.g.*, [Bar03, HJ11, Szy98]).

To understand hybrid automata as state-based components is an important step towards their coalgebraic characterisation in the spirit of [Bar03, HJ11]. Such an achievement would provide them several composition operators (with corresponding laws), refinement techniques, and synchronisation mechanisms.

Another relevant point is the existence of several variants of hybrid automata (*e.g.*, [Hen96, Spr00, LLK+99]), motivated by the need to capture different types of behaviour (*e.g.*, nondeterministic, probabilistic, faulty). To the best of our knowledge, a uniform, formal theory for different types of hybrid automata does not yet exist.

1.2 Contributions

This paper characterises hybrid automata as coalgebras of a specific type. This promotes the black-box perspective discussed above, where the (discrete) state transitions are internal, hidden from the environment, and the continuous evolutions are external, making up the observable behaviour. To be concrete,

- 'going coalgebraic' provides a uniform, canonical *observational semantics* that faithfully reflects the black-box perspective, and frames the behaviour into well known constructions (*e.g.*, streams, infinite binary trees), marking a separation between the discrete domain and the continuous one.
- Moreover, a generic (coalgebraic) characterisation of *bisimulation*, parametrised by a transition type (technically, a functor), emerges across different

sorts of hybrid automata in a uniform manner. Indeed, it is shown that different notions of bisimilarity (associated with variants of hybrid automata) are subsumed by the corresponding coalgebraic definition.

We will also see that the coalgebraic characterisation proposed in this paper 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 becomes reduced to a machine that from a state (internally) jumps to another, and (externally) produces a continuous evolution. As expected, this implies that, even in the presence of both discrete and continuous behaviour, only the continuous part can be directly observed.

The coalgebraic characterisation paves the way to yet another contribution: a hierarchy of different types of hybrid automata organised with respect to their 'expressivity', a concept also to be here understood within the coalgebraic framework.

1.3 Roadmap

Section 2 provides a brief background on hybrid automata and coalgebras. Section 3 establishes the relation between classic hybrid automata (in a deterministic setting) and the corresponding coalgebras. In particular, it shows how to encode hybrid automata as coalgebras, explores the associated observational semantics, and reframes the classic notion of bisimulation (for hybrid automata) as a coalgebraic one.

Then, building on the coalgebraic perspective, Sect. 4 considers different types of functors in order to (re)discover several variants of hybrid automata. Two interesting cases are the ones that involve *probabilistic* [Spr00] and *replicating* behaviour, the latter being new to the best of our knowledge. Section 4 also establishes the hierarchy of hybrid automata mentioned above. Finally, Sect. 5 concludes and hints at future work directions.

We assume that the reader has some familiarity with elementary category theory and topology.

2 Background

2.1 Hybrid Automata

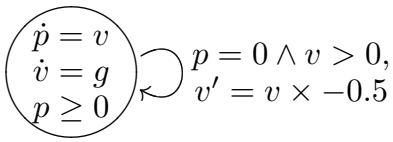
Introduced in the early nineties as an answer to the rapid emergence of hybrid systems, hybrid automata form an active research area that encompasses diverse topics. These span from decidability [Hen96], to extensions that cater for *input* mechanisms (*e.g.*, [AH97,LLK+99]), and *uncertainty* [Spr00]. Hybrid automata have also been considered as a modelling tool in life sciences [BCB+09,AMP+03]. Formally,

Definition 1 ([Hen96]). *A hybrid automaton is a tuple* $(M, E, \Sigma, X, \mathsf{init}, \mathsf{inv}, \mathsf{dyn}, \mathsf{asg}, \mathsf{grd})$ *where*

- *M is a finite set of discrete states (often called control modes, or locations), E is a transition relation $E \subseteq M \times \Sigma \times M$, and Σ a set of labels. A triple $(m_1, l, m_2) \in E$ will often be written as $m_1 \overset{l}{\rightsquigarrow} m_2$.*
- *X is a finite set of real-valued variables $\{x_1, \ldots, x_n\}$.*
- init *and* inv *are functions that associate to each mode a predicate over the variables in X. Letter Z denotes the set $\{(m, v) \in M \times \mathbb{R}^n \mid v \models (\mathsf{inv}\ m)\}$, where expression $v \models (\mathsf{inv}\ m)$ means that predicate $(\mathsf{inv}\ m)$ is satisfied by v.*
- dyn *is a function that associates to each state a predicate over the variables in $X \cup \dot{X}$, where $\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ represents the first derivatives of the variables in X. It is used to define the set of continuous evolutions that may occur at each state.*
- asg *is a function such that given an edge ($e \in E$) returns a predicate over $X \cup X'$, where $X' = \{x'_1, \ldots, x'_n\}$ represents the variables in X immediately after a discrete jump. This provides an assignment to each edge. Finally, the function* grd *associates each edge with a guard,* i.e.*, a predicate over X.*

A classic example may help to illustrate this quite complex definition.

Example 2. Consider a bouncing ball dropped at some positive height p and with no initial velocity v. Due to the gravitational acceleration g, it falls into the ground but then bounces back up, losing part of its kinetic energy in the process. The following hybrid automaton sums up this behaviour.



Note that only one mode exists; let us call it m. Also, there is exactly one discrete transition: $m \rightsquigarrow m \in E$, omitting its label for simplicity. Actually, in this example there is no need for labels. Then $X = \{p, v\}$, and $(\mathsf{inv}\ m)$ is $p \geq 0$ – which entails $Z = \{m\} \times \mathbb{R}_{\geq 0} \times \mathbb{R}$, where the second ($\mathbb{R}_{\geq 0}$) and third components ($\mathbb{R}$) denote, respectively, position and velocity.

Finally, $\mathsf{grd}(m \rightsquigarrow m)$ is $p = 0 \wedge v > 0$, $(\mathsf{dyn}\ m)$ is $\{\dot{p} = v, \dot{v} = g\}$, and $\mathsf{asg}(m \rightsquigarrow m)$ is $v' = v \times -0.5 \wedge p' = p$. Note that the right-hand side of the last predicate does not appear in the hybrid automaton above, a common practice to avoid a burdened notation.

In order to keep results simple and intuitive, we do not consider labels or initial states, as they can be accommodated later on in a straightforward manner.

Frequently it is assumed that, given any mode, function dyn returns a system of differential equations with exactly one solution (*e.g.*, [Jac00, ACH+95]). We adopt this approach as well. Such an assumption may seem too restrictive but, in fact, such is not the case for most hybrid systems described in the literature, as they rarely involve nonlinear differential equations. The important point is that this condition allows function dyn to induce a function,

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

such that given a pair $(m, v) \in (M \times \mathbb{R}^n)$, $\mathsf{flow}\,((m, v), -) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is a continuous function, which represents the solution to the system of differential equations; note that its domain ($\mathbb{R}_{\geq 0}$) represents time.

Assume also that an hybrid automaton cannot jump from a valid state $(m, v) \in (M \times \mathbb{R}^n)$ into an invalid one, where by valid we mean that $(m, v) \in Z$. In symbols, assume that for any pair $\big((m_1, v_1), (m_2, v_2)\big) \in (M \times \mathbb{R}^n)^2$ such that $m_1 \rightsquigarrow m_2$, $v_1 \models \mathsf{grd}(m_1 \rightsquigarrow m_2)$, and $(v_1, v_2) \models \mathsf{asg}(m_1 \rightsquigarrow m_2)$ we have $v_2 \models (\mathsf{inv}\ m_2)$.

As mentioned in Sect. 1, the semantics of hybrid automata is traditionally described in terms of LTSs.

Definition 2 ([Hen96]). *Consider a hybrid automaton. Its underlying* LTS *is a tuple* (Z, L, T) *such that* $L = 1 + \mathbb{R}_{\geq 0}$ (1 *is a singleton set*), *and* $T \subseteq Z \times L \times Z$ *is defined as* $\big((m_1, v_1), l, (m_2, v_2)\big) \in T$ *iff*

1. *if* $l \in 1$ *then* $m_1 \rightsquigarrow m_2$, $v_1 \models \mathsf{grd}\,\big(m_1 \rightsquigarrow m_2\big)$, $(v_1, v_2) \models \mathsf{asg}\,\big(m_1 \rightsquigarrow m_2\big)$,
2. *if* $l \in \mathbb{R}_{\geq 0}$ *then* $m_1 = m_2$, $\mathsf{flow}\,((m_1, v_1), l) = v_2$, *and for all* $t \in [0, l]$ $\mathsf{flow}\,((m_1, v_1), t) \models (\mathsf{inv}\ m_1)$.

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

Example 3. Recall the hybrid automaton from Example 2. The associated LTS (Z, L, T) is defined as follows: $Z = \{m\} \times \mathbb{R}_{\geq 0} \times \mathbb{R}$, $L = 1 + \mathbb{R}_{\geq 0}$, and $(m, p_1, v_1) \xrightarrow{\ l\ } (m, p_2, v_2)$ iff

1. if $l \in 1$ then $p_1 = 0 \wedge v_1 > 0$, and $v_2 = v_1 \times -0.5 \wedge p_1 = p_2$;
2. if $l \in \mathbb{R}_{\geq 0}$ then $\mathsf{flow}\,((m, p_1, v_1), l) = (p_2, v_2)$, and for all $t \in [0, l]$, $\mathsf{flow}\,((m, p_1, v_1), t) \geq 0$.

In this case the function flow, induced by dyn, describes the continuous evolution of 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 above, but it also turns the verification of hybrid automata into a challenging task, as an infinite number of states and edges needs to be taken into consideration. The standard technique for overcoming the latter issue is to quotient by a bisimulation equivalence, *i.e.*, to collapse states that possess equivalent behaviour. The resulting states become then symbolic representations of (possibly infinite) regions, and verification techniques are applied to the reduced system instead.

Definition 3 ([Hen96]). *Consider the underlying labelled transition system* (S, L, T) *of a hybrid automaton, and an equivalence relation* $\Phi \subseteq S \times S$ *over the states. A* Φ*-bisimulation* $R \subseteq S \times S$ *is a relation such that* $(s_1, q_1) \in R$ (*or more concisely,* $s_1\, R\, q_1$) *entails the following cases:*

1. $s_1\, \Phi\, q_1$,

2. *for each label* $l \in L$, *if* $s_1 \xrightarrow{l} s_2$ *then there is a state* q_2 *such that* $q_1 \xrightarrow{l} q_2$ *and* $s_2 \, R \, q_2$,
3. *for each label* $l \in L$, *if* $q_1 \xrightarrow{l} q_2$ *then there is a state* s_2 *such that* $s_1 \xrightarrow{l} s_2$ *and* $s_2 \, R \, q_2$.

Two states $s_1, q_1 \in S$ are *Φ-bisimilar* (in symbols, $s_1 \equiv^{\Phi} q_1$) if they are related by a Φ-bisimulation.
We will start our (coalgebraic) rendering of hybrid automata in a deterministic setting, restricting Definition 1 with the following conditions:

1. Relation E is a function ($E : M \to M$).
2. Assignments are deterministic, *i.e.*, they take the form $x := \theta$, where θ is an expression with variables of X that denotes a real value, and $x \in X$. For example, in the case of the bouncing ball above, the assignment $v' = v \times -0.5$ is changed to $v := v \times -0.5$. Note that Example 1 (tank-and-valve) also adopted this approach.
3. As soon as an edge becomes enabled (*i.e.* the associated guard is satisfied) the current state must switch (a similar condition is adopted in [Nad97], where hybrid automata with this property are called time-deterministic). More concretely, each pair $(m, v) \in Z$ has exactly one duration ($\delta \in \mathbb{R}_{\geq 0}$) for its evolution $\mathsf{flow}((m, v), -) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, which, intuitively, corresponds to the time that the current mode takes to jump starting in (m, v). This happens, for example, in the hybrid automaton that describes the tank-and-valve (c seconds) and the bouncing ball system (the time the ball takes to reach the ground from a specific height and velocity).
   Unlike the two conditions above, this condition, which we refer to as *as-soon-as*, is assumed throughout the paper.

The three conditions together give no possibility for a hybrid automaton to choose between possible executions, and therefore induce a function $\mathsf{nxt} : Z \to Z$, which given a pair $(m, v) \in Z$, returns the pair that results from the corresponding evolution (given by function flow and associated duration δ) and subsequent discrete transition. Formally,

$$\mathsf{nxt}(m, v) = \big(E(m),\ \mathsf{asg}(m \rightsquigarrow E(m))\ u\big)$$

where $u = \mathsf{flow}((m, v), \delta)$. By a slight abuse of notation we denote the expression $\mathsf{asg}(m \rightsquigarrow E(m))$ as a function. Note also that the value u is the last point (in the evolution of (m, v)) before the jump.

2.2 Coalgebras

The theory of coalgebras [Rut00] establishes an abstract, categorial framework that promotes a uniform study of state-based transition systems[1]. The idea is

[1] We restrict ourselves to the concepts strictly necessary to the paper. The interested reader will find in document [Rut00] a comprehensive introduction to the theory of coalgebras.

that a functor $\mathcal{F} : \mathbf{C} \to \mathbf{C}$ over some category $\mathbf{C}$ (typically, **Set**) gives 'shape' to a transition type, and arrows $S \to \mathcal{F}S$ in $\mathbf{C}$ ($\mathcal{F}$-coalgebras, or simply coalgebras) make up the family of corresponding transition systems.

Definition 4. *Consider a functor $\mathcal{F} : \mathbf{C} \to \mathbf{C}$. It gives rise to category $\mathbf{CoAlg}_{\mathcal{F}}$ whose objects are coalgebras $S \to \mathcal{F}S$, and morphisms between two coalgebras $\alpha : S \to \mathcal{F}S$, $\beta : Q \to \mathcal{F}Q$ are arrows $f : S \to Q$ in $\mathbf{C}$ such that the diagram below in the left commutes.*

$$\begin{array}{ccc} S & \xrightarrow{f} & Q \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\beta} \\ \mathcal{F}S & \xrightarrow[\mathcal{F}f]{} & \mathcal{F}Q \end{array} \qquad \begin{array}{ccc} S & \overset{[\![-]\!]}{\dashrightarrow} & \nu_{\mathcal{F}} \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\gamma} \\ \mathcal{F}S & \underset{\mathcal{F}[\![-]\!]}{\dashrightarrow} & \mathcal{F}\nu_{\mathcal{F}} \end{array}$$

Under mild conditions, a category $\mathbf{CoAlg}_{\mathcal{F}}$ has a *final* object, *i.e.*, a coalgebra $\gamma : \nu_{\mathcal{F}} \to \mathcal{F}\nu_{\mathcal{F}}$ such that for any coalgebra $\alpha : S \to \mathcal{F}S$ there is a unique morphism $[\![-]\!] : S \to \nu_{\mathcal{F}}$ that makes the diagram above in the right to commute. A prime example is the final $(- \times A)$-coalgebra $\langle tl, hd\rangle : A^\omega \to A^\omega \times A$. Briefly, A^ω is the set of infinite lists (streams) of elements in A, and $\langle tl, hd\rangle$ is defined as,

$$\langle tl, hd\rangle\ (a_0, a_1, \dots) = ((a_1, \dots), a_0).$$

Since $\langle tl, hd\rangle$ is final, each coalgebra $\alpha : S \to S \times A$ has a unique morphism $[\![-]\!]_\alpha : S \to A^\omega$, called the behaviour or *coinductive* extension of α – whenever found suitable we will drop the subscript in $[\![-]\!]_\alpha$. Intuitively, $[\![-]\!]_\alpha : S \to A^\omega$ gives the observable behaviour of each state ($s \in S$) of $\alpha : S \to S \times A$. Actually, final objects in categories of coalgebras provide the observational semantics mentioned in Sect. 1.

Bisimulation is another key concept in coalgebra theory.

Definition 5. *Consider two $\mathcal{F}$–coalgebras $\alpha : S \to \mathcal{F}S$, $\beta : Q \to \mathcal{F}Q$ in* **Set**, *and a relation $R \subseteq S \times Q$. Then R is an $\mathcal{F}$-bisimulation (or simply bisimulation) if there is a third coalgebra $\gamma : R \to \mathcal{F}R$ that makes the following diagram to commute.*

$$\begin{array}{ccccc} S & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Q \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\gamma} & & \downarrow{\scriptstyle\beta} \\ \mathcal{F}S & \xleftarrow[\mathcal{F}\pi_1]{} & \mathcal{F}R & \xrightarrow[\mathcal{F}\pi_2]{} & \mathcal{F}Q \end{array}$$

We say that states $s \in S$, and $q \in Q$ are *coalgebraically bisimilar* (in symbols, $s \sim q$) if they are related by some $\mathcal{F}$-bisimulation.

3 Deterministic Hybrid Automata as Coalgebras

3.1 The Model

In order to encode hybrid automata as coalgebras, recall the state-based, black-box perspective described in the introductory section: discrete transitions occur internally, hidden from the environment, whereas the observable behaviour (or output) corresponds to continuous evolutions. As explained before, for any given suitable pair $(m, v) \in (M \times \mathbb{R}^n)$, a hybrid automaton outputs a continuous evolution over $\mathbb{R}^n$, with a specific duration $\delta \in \mathbb{R}_{\geq 0}$. Formally, a continuous function $[0, \delta] \to \mathbb{R}^n$ where $[0, \delta]$ has the subspace topology induced by the Euclidean one, and $\mathbb{R}^n$ has the Euclidean topology – this requires a brief use of topological notions in the following construction.

Definition 6. *Generalising the output type from $\mathbb{R}^n$ to an arbitrary topological space (O, τ), the output of an hybrid automaton is defined as the sum of all* continuous *evolutions over (O, τ). In symbols,*

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

*where $[0, \delta]$ is equipped with the subspace topology induced by the Euclidean one, and U : **Top** $\to$ **Set** is the forgetful functor between the category of topological spaces and continuous functions* (**Top**) *and* **Set**. *We will denote the construction above by $\mathcal{H}(O, \tau)$, or simply $\mathcal{H}O$.*

For what follows, let us denote the curried version of a function $f : A \times B \to C$, by $\lambda f : A \to C^B$. Then, consider a hybrid automaton and recall that each pair $(m, v) \in Z$ defines $\mathsf{flow}((m, v), -) : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ whose domain can be restricted to duration $[0, \delta]$. This leads to a function $\lambda\,\mathsf{flow} : Z \to \mathcal{H}(\mathbb{R}^n)$, which by a slight abuse of notation, and for the sake of generality we type as

$$\mathsf{out} : Z \to \mathcal{H}O.$$

Finally, note that function $\mathsf{out} : Z \to \mathcal{H}O$, together with function $\mathsf{nxt} : Z \to Z$ (see Sect. 2), forms a $(- \times \mathcal{H}O)$-coalgebra

$$\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to Z \times \mathcal{H}O,$$

which (fully) characterises the behaviour of the hybrid automaton.
The intuition is that each state $(m, v) \in Z$ gives rise to an observable, continuous evolution ($e \in \mathcal{H}O$), and an internal, discrete transition to the next state ($z \in Z$). Let us illustrate this concept with a few examples.

Example 4. Recall the tank-and-valve system described in Sect. 1. The corresponding coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to Z \times \mathcal{H}O$ is defined as

$$\langle \mathsf{nxt}, \mathsf{out} \rangle (m_1, l, t) = ((m_2, l + 2\,\mathsf{c}, 0), f), \quad \langle \mathsf{nxt}, \mathsf{out} \rangle (m_2, l, t) = ((m_1, l, 0), g)$$

where the functions $f, g : [0, \mathsf{c}] \to \mathbb{R}^2$ are defined as

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

Example 5. Consider again the bouncing ball system and, for illustration purposes, take only its movement as the observable behaviour. The corresponding coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to Z \times \mathcal{H}O$ is given by

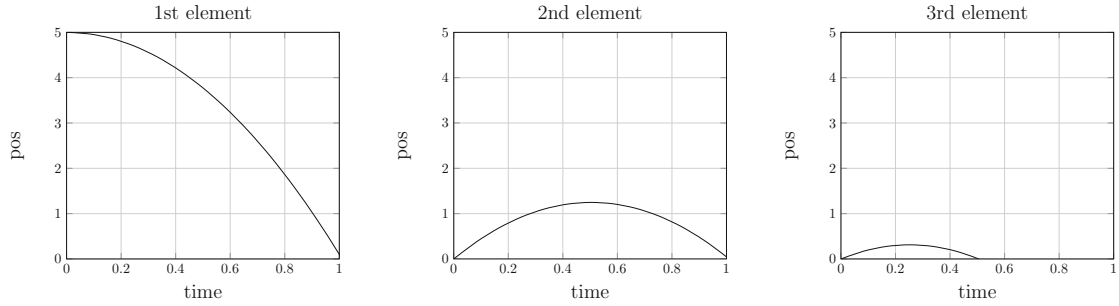
$$\langle \mathsf{nxt}, \mathsf{out} \rangle\ (m, p, v) = \big((m, 0, v'), \mathsf{mov}(p, v, -)\big)$$

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

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

As mentioned in the previous section, each coalgebra $S \to S \times \mathcal{H}O$ yields a function $[\![-]\!] : S \to (\mathcal{H}O)^\omega$ which computes, for a given $s \in S$, a stream of (observable) continuous evolutions $[\![s]\!]$, which correspond to the (internal) states that are visited starting in s. For example,

Example 6. Consider again the bouncing ball system; the first three elements of $[\![(0, 5)]\!]$ are represented in the following plots.



3.2 Bisimulation in the Deterministic Case

Recall from the previous section that bisimulation for hybrid automata (Definition 3) is parametrised by an equivalence relation over the state space. Let us see how to capture this coalgebraically.

Consider a coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to Z \times \mathcal{H}O$ (modelling a hybrid automaton) and an equivalence relation over its states $\Phi \subseteq Z \times Z$. We define a coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi : Z \to Z \times \mathcal{H}(Z/_\Phi)$ such that

$$\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi z = (\mathsf{nxt}\, z, q \cdot (\mathsf{ev}\, z))$$

where $\mathsf{ev} : Z \to \mathcal{H}Z$ is defined as $(\mathsf{ev}(m, v))\ t = (m, (\mathsf{out}(m, v))\ t)$, and $q : Z \to Z/_\Phi$ is the quotient map induced by Φ.

Technically, $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi$ is a $\mathcal{F}_{Z/_\Phi}$-coalgebra where $\mathcal{F}_{Z/_\Phi} X = X \times \mathcal{H}(Z/_\Phi)$. Intuitively, coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi$ behaves like $\langle \mathsf{nxt}, \mathsf{out} \rangle$ but allows its internal states and continuous evolutions to be 'partially' observed; 'how much' one can observe, is dictated by equivalence relation Φ. Denoting $Z/_\Phi$ by Q,

Definition 7. *Consider a coalgebra* $\langle \mathsf{nxt}, \mathsf{out} \rangle^{\Phi} : Z \to Z \times \mathcal{H}Q$ *induced by a hybrid automaton and an equivalence relation* $\Phi \subseteq Z \times Z$. *A relation* $R \subseteq Z \times Z$ *is a* coalgebraic Φ-bisimulation *iff there is a* $\mathcal{F}_{Z/\Phi}$*-coalgebra* $\gamma : R \to R \times \mathcal{H}Q$ *that makes the following diagram to commute.*

$$\begin{array}{ccccc} Z & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Z \\ {\scriptstyle \langle \mathsf{nxt},\mathsf{out} \rangle^{\Phi}}\downarrow & & {\scriptstyle \gamma}\downarrow & & \downarrow{\scriptstyle \langle \mathsf{nxt},\mathsf{out} \rangle^{\Phi}} \\ Z \times \mathcal{H}Q & \xleftarrow[\pi_1 \times id]{} & R \times \mathcal{H}Q & \xrightarrow[\pi_2 \times id]{} & Z \times \mathcal{H}Q \end{array}$$

We say that states $z_1, z_2 \in Z$ are *coalgebraically Φ-bisimilar* (in symbols, $z_1 \sim^{\Phi} z_2$) if they are related by a coalgebraic Φ-bisimulation.

Given two functions $f, g : A \to B$, and relation $R \subseteq B \times B$, denote the condition $\forall a \in A.\, (f\, a)\, R\, (g\, a)$ by $f\, R\, g$. Definition 7 tells that a relation R is a coalgebraic Φ-bisimulation iff $z_1\, R\, z_2$ implies

$$(\mathsf{ev}\ z_1)\, \Phi\, (\mathsf{ev}\ z_2), \text{ and } (\mathsf{nxt}\ z_1)\, R\, (\mathsf{nxt}\ z_2).$$

Theorem 1. *Let* $\langle \mathsf{nxt}, \mathsf{out} \rangle^{\Phi} : Z \to Z \times \mathcal{H}Q$ *be induced by a hybrid automaton and an equivalence relation* $\Phi \subseteq Z \times Z$. *Then for any two states* $z_1, z_2 \in Z$, $z_1 \equiv^{\Phi} z_2$ *iff* $z_1 \sim^{\Phi} z_2$.

Proof. In [NB16].

4 When Different Transition Types Come into Play

4.1 The General Picture

The previous section introduced a coalgebraic semantics for hybrid automata in a deterministic setting. The behaviour of digital controllers, however, is far more complex, often combining nondeterministic, or probabilistic features. This calls for variations in the definition of hybrid automata, and, consequently, for a more general coalgebraic semantics, able to capture such variants in a uniform manner. Therefore, we consider coalgebras,

$$\langle \mathsf{nxt}, \mathsf{out} \rangle : S \to (\mathcal{F}S \times \mathcal{H}O)^I$$

where $\mathcal{F}$ determines an internal transition type, and set I denotes an input type. Technically, such arrows can be decomposed into $\mathsf{nxt} : S \times I \to \mathcal{F}S$, $\mathsf{out} : S \times I \to \mathcal{H}O$ (again by a slight abuse of notation). This makes clear that variations in functor $\mathcal{F}$ correspond to variations on how the system (discretely) jumps to a next state. In regard to hybrid automata, we will see that these changes are essentially reflected in relation E and the assignment function asg (recall Definition 1).

Table 1 lists several variations of the functor $\mathcal{F}$ and input type I. Each variant corresponds to a specific definition of hybrid automata. Some of the latter are already well known (*e.g.* the nondeterministic case in row 4), but others are new and thus have not been studied before (*e.g.* the replicating case in row 3).

Table 1. Possible variants for $\mathcal{F}$

Coalgebra	Functor $\mathcal{F}$	Behaviour	Input
$S \to (S \times \mathcal{H}O)$	$Id\ X = X$	Deterministic	No
$S \to (S \times \mathcal{H}O)^I$	$Id\ X = X$	Deterministic	Yes
$S \to (\Delta S \times \mathcal{H}O)$	$\Delta\ X = X \times X$	Replicating	No
$S \to (\mathcal{P}S \times \mathcal{H}O)$	$\mathcal{P}\ X = \{A \subseteq X\}$	Nondeterministic	No
$S \to (\mathcal{D}S \times \mathcal{H}O)$	$\mathcal{D}\ X = \{\mu \in [0,1]^X \mid \mu[X] = 1\}$[1]	Probabilistic	No
$S \to (\mathcal{P}\mathcal{D}S \times \mathcal{H}O)$	$\mathcal{P}\mathcal{D}$ —	Segala[2]	No

[1] $\mu[X] = \sum_{x \in X}\ \mu\ x$.
[2] Traditionally this expression refers to systems with both nondeterministic and probabilistic behaviour.

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 functor $\mathcal{F}$, and global constructions and results are defined parametric on $\mathcal{F}$ once and for all. For example, such is the case of coalgebraic Φ-bisimulation, which we will discuss in Sect. 4.4.

The cases listed in Table 1 will be discussed in more detail in the following sections.

4.2 Reactive and Replicating Behaviour

The arrows $S \to (S \times \mathcal{H}O)$ were studied in the previous section. We saw that they provide a suitable coalgebraic semantics for deterministic hybrid automata. Hence, we pass directly to arrows typed as, $S \to (S \times \mathcal{H}O)^I$.

These correspond to a variant of hybrid automata, qualified as *open* (or *reactive*), that takes input/output into consideration (*cf.* [LLK+99]); thus extending the classical definition of hybrid automata (Definition 1) as follows:

Definition 8 ([LLK+99])**.** *Fix an input set I. Then, add I to the domain of functions* $\mathsf{dyn}, \mathsf{inv}, \mathsf{grd}$*, and* asg*, keeping the remaining components equal.*

For example, while in the classic case each mode $m \in M$ gave rise to a predicate $(\mathsf{inv}\ m)$, now each pair $(m, i) \in M \times I$ induces a predicate $(\mathsf{inv}\ (m, i))$.

Similarly, function $\mathsf{flow} : (M \times \mathbb{R}^n) \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, induced by dyn, has now the signature

$$\mathsf{flow} : (M \times \mathbb{R}^n) \times I \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n.$$

In order to encode open hybrid automata as coalgebras, the condition as-soon-as also needs to be slightly changed: while previously each pair $(m, v) \in (M \times \mathbb{R}^n)$ was associated with a duration δ (see Sect. 2), now we require the same for each triple $(m, v, i) \in (M \times \mathbb{R}^n \times I)$. As in Sects. 2 and 3, we also assume that an open hybrid automaton cannot jump from a valid state into an invalid one.

Then, let us define function $\mathsf{nxt} : Z \times I \to Z$ as

$$\mathsf{nxt}(m, v, i) = \big(E(m), \mathsf{asg}(m \rightsquigarrow E(m), i)\ u\big)$$

where $u = \mathsf{flow}(m, v, i, \delta)$. Finally, given functions $\mathsf{nxt} : Z \times I \to Z$, $\mathsf{out} : Z \times I \to \mathcal{H}O$, we form coalgebra

$$\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to (Z \times \mathcal{H}O)^I.$$

Let us illustrate the expressive power of $(- \times \mathcal{H}O)^I$-coalgebras via an example related to the bouncing ball system.

Example 7. Suppose we can set the instants of time at which the ball bounces – this may be interpreted, for example, as a foot that kicks the ball up. To define such a behaviour one can construct the following coalgebra:

$$\langle \mathsf{nxt}, \mathsf{out} \rangle\ (m, p, v, i) = \big((m, 0, v'), \mathsf{mov}(p, v, -)\big)$$

where $v' = (v + gi) \times -0.5$, and function $\mathsf{mov}(p, v, -) : [0, i] \to \mathbb{R}$ is defined as before.

The category of $(- \times \mathcal{H}O)^I$-coalgebras also has a final coalgebra. Formally, the following diagram commutes uniquely,

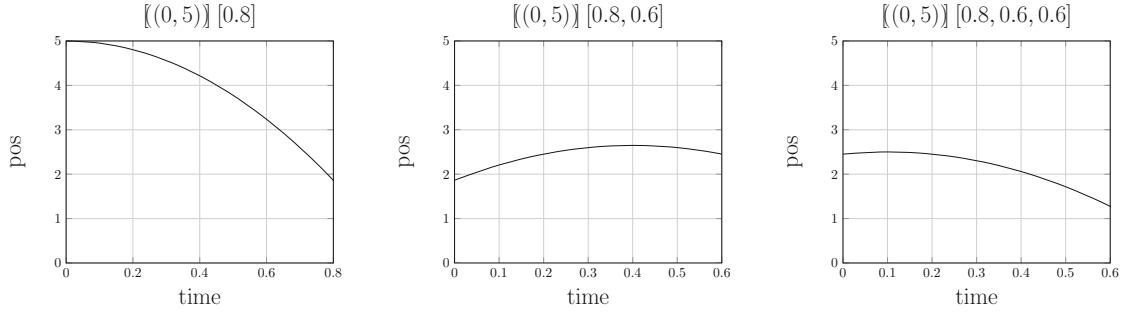
$$\begin{array}{ccc} S & \xrightarrow{\quad [\![-]\!] \quad} & (\mathcal{H}O)^{I^+} \\ {\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \gamma} \\ (S \times \mathcal{H}O)^I & \xrightarrow[([\![-]\!] \times id)^I]{} & ((\mathcal{H}O)^{I^+} \times \mathcal{H}O)^I \end{array}$$

where I^+ denotes the set of nonempty lists of elements in I, and

$$(\gamma\ \ f)\ i = (g,\ f\ [i]),\ \ g\ is = f\ (i : is).$$

Intuitively, any coalgebra $\alpha : S \to (S \times \mathcal{H}O)^I$, induces a unique function $[\![-]\!] : S \to (\mathcal{H}O)^{I^+}$ (*cf.* [Jac12]), such that $[\![s]\!]$ associates to each nonempty list of inputs the *last* evolution observed in α, starting in $s \in S$.

Example 8. In Example 7 we considered a bouncing ball system that allows to choose the instants of time at which the ball bounces. Expressions $[\![(0, 5)]\!]\ [0.8]$, $[\![(0, 5)]\!]\ [0.8, 0.6]$, and $[\![(0, 5)]\!]\ [0.8, 0.6, 0.6]$ denote the following sequence.



Functor Diagonal (Δ) gives rise to arrows of type $\langle \mathsf{nxt}, \mathsf{out} \rangle : S \to \Delta S \times \mathcal{H}O$. These correspond to deterministic hybrid automata, as studied in Sect. 3, but now able to jump to two different places at the same time. The intuition is that such systems replicate themselves at each discrete transition. For example, the bouncing ball would turn into two at each bounce. From a strict computer science point of view this may seem rather strange, but in other areas it is a common behaviour: *e.g.*, in biology, cells indeed replicate when a specific saturation point is reached. To the best of our knowledge, there is no variant of hybrid automata in the literature associated with this type of behaviour.

4.3 Nondeterministic and Probabilistic Behaviour

Let us now concentrate on the powerset functor ($\mathcal{P}$). Actually, for the sake of simplicity we will restrict to its *finitary* version, $\mathcal{P}_\omega$, which considers only the finite subsets of a given set X. As expected, it gives rise to arrows typed as,

$$Z \to (\mathcal{P}_\omega Z \times \mathcal{H}O)$$

which precisely correspond to a nondeterministic version of the hybrid automata explored in the previous section. More concretely,

- relation E, previously assumed to be a function, is now *finitely branching* (*i.e.*, each mode has a finite number of outgoing edges),
- the assignments are allowed to be *finitely non deterministic*, meaning that the value assigned to a variable is determined up to a finite number of possibilities.

Thus, function $\mathsf{nxt} : Z \to \mathcal{P}_\omega Z$ is defined as

$$\mathsf{nxt}(m, v) = \bigcup_{m' \in E(m)} \big(\{m'\} \times \mathsf{asg}(m \rightsquigarrow m')(u) \big)$$

where $u = \mathsf{flow}(m, v, \delta)$, and $\mathsf{asg}(m \rightsquigarrow m')$ is regarded as a function that given a tuple of valuations $v \in \mathbb{R}^n$, returns the assignments that are possible to perform.

Consider now probabilistic branching by taking $\mathcal{F} = \mathcal{D}$, or $\mathcal{F} = \mathcal{P}_\omega \mathcal{D}$, in $S \to (\mathcal{F}S \times \mathcal{H}O)$. Interestingly, hybrid automata whose internal transition type corresponds to $\mathcal{P}_\omega \mathcal{D}$ were already introduced in document [Spr00]. The idea is that these systems are able to nondeterministically choose a distribution function over the states (which, intuitively, gives the probability of a given state being the next one). Actually, not only this allows to equip edges with probabilities, but also gives rise to probabilistic assignments: for example, one may say $x := x + 10$ *with probability* 0.9.

We refer the interested reader to this paper's extended version [NB16] for a more detailed overview of arrows $S \to \mathcal{D}S \times \mathcal{H}O$, $S \to \mathcal{P}_\omega \mathcal{D}S \times \mathcal{H}O$ and their correspondence to the probabilistic hybrid automata introduced in [Spr00].

4.4 Bisimulation and Observational Semantics

Let us now generalise the notion of coalgebraic Φ-bisimulation (Definition 7) to coalgebras typed as $\langle \mathsf{nxt}, \mathsf{out} \rangle : Z \to (\mathcal{F}Z \times \mathcal{H}O)^I$. As before, assume that $Z \subseteq M \times \mathbb{R}^n$. Then given an equivalence relation $\Phi \subseteq Z \times Z$, we define coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi : Z \to (\mathcal{F}Z \times \mathcal{H}Q)^I$ similarly to before. More concretely,

$$\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi(z, i) = (\mathsf{nxt}(z, i), q \cdot (\mathsf{ev}(z, i)))$$

where $\mathsf{ev} : Z \times I \to \mathcal{H}Z$ is a function such that for any $z = (m, v) \in Z, i \in I$, $(\mathsf{ev}(z, i))\ t = (m, (\mathsf{out}(z, i))\ t)$, and $q : Z \to Z/_\Phi$ is the quotient map induced by Φ. Then denoting $Z/_\Phi$ by Q,

Definition 9. *Consider a coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi : Z \to (\mathcal{F}Z \times \mathcal{H}Q)^I$ induced by an equivalence relation $\Phi \subseteq Z \times Z$. A relation $R \subseteq Z \times Z$ is a coalgebraic Φ-bisimulation if there is a coalgebra $R \to (\mathcal{F}R \times \mathcal{H}Q)^I$ that makes the following diagram to commute.*

$$\begin{array}{ccccc} Z & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & Z \\ {\scriptstyle \langle \mathsf{nxt},\mathsf{out} \rangle^\Phi}\downarrow & & \downarrow & & \downarrow{\scriptstyle \langle \mathsf{nxt},\mathsf{out} \rangle^\Phi} \\ (\mathcal{F}Z \times \mathcal{H}Q)^I & \xleftarrow[(\mathcal{F}\pi_1 \times id)^I]{} & (\mathcal{F}R \times \mathcal{H}Q)^I & \xrightarrow[(\mathcal{F}\pi_2 \times id)^I]{} & (\mathcal{F}Z \times \mathcal{H}Q)^I \end{array}$$

We say that states $z_1, z_2 \in Z$ are *coalgebraically Φ-bisimilar* (in symbols, $z_1 \sim^\Phi z_2$) if they are related by a coalgebraic Φ-bisimulation.

Observe that a coalgebraic Φ-bisimulation R is, in fact, a coalgebraic bisimulation in the category of $(\mathcal{F} \times \mathcal{H}Q)^I$-coalgebras. Moreover, note that this definition coincides with Definition 7 when $\mathcal{F} = Id$ and $I = 1$. Actually, for $\mathcal{F} = \mathcal{P}_\omega$, $\mathcal{F} = \mathcal{P}_\omega\mathcal{D}$ (with $I = 1$) we have the following results relating classic and coalgebraic Φ-bisimilarity, $\equiv^\Phi$ and $\sim^\Phi$, respectively.

Theorem 2. *Consider a coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi : Z \to (\mathcal{P}_\omega Z \times \mathcal{H}Q)$ induced by a nondeterministic hybrid automaton and an equivalence relation $\Phi \subseteq Z \times Z$. Then for any two states $z_1, z_2 \in Z$, $z_1 \equiv^\Phi z_2$ iff $z_1 \sim^\Phi z_2$.*

Proof. In [NB16].

Theorem 3. *Consider a coalgebra $\langle \mathsf{nxt}, \mathsf{out} \rangle^\Phi : Z \to (\mathcal{P}_\omega\mathcal{D}Z \times \mathcal{H}Q)$ induced by a probabilistic hybrid automaton [Spr00] and an equivalence relation $\Phi \subseteq Z \times Z$. Then, for any two states $z_1, z_2 \in Z$, $z_1 \equiv^\Phi z_2$ iff $z_1 \sim^\Phi z_2$.*

Proof. In [NB16].

Another interesting aspect to mention concerns open hybrid automata and the apparent absence of a suitable notion of Φ-bisimulation for them (see the previous subsection and also [LLK+99]). However, instantiating Definition 9 with $\mathcal{F} = Id$, we obtain a suitable notion of Φ-bisimulation for such automata, which gives evidence to the generality of the coalgebraic framework.

In order to characterise the observational semantics associated with the arrows $S \to (\mathcal{F}S \times \mathcal{H}O)^I$, we need to guarantee the existence of a final $(\mathcal{F} \times \mathcal{H}O)^I$-coalgebra. In **Set**, the existence of an observational semantics (*i.e.*, a final coalgebra) for systems of type $S \to (\mathcal{F}S \times \mathcal{H}O)^I$ is ensured whenever functor $\mathcal{F}$ is *bounded* (*cf.* [Rut00]). This is not a strong condition. Actually, it holds for all polynomial functors, the finite powerset ($\mathcal{P}_\omega$), and all composites made up of these cases (the reader will find in [Rut00] a complete characterisation of this condition and corresponding proofs). Another case is the distribution functor with finite support ($\mathcal{D}_\omega$); more explicitly, the restriction of functor $\mathcal{D}$ that only considers distributions $\mu \in \mathcal{D}_\omega X$ with a finite number of elements $x \in X$ such that $\mu\, x > 0$ (see the proof, for example, in [Jac12], Theorem 4.6.9).

Therefore, all cases enumerated in Table 1 have a final coalgebra provided that functors $\mathcal{P}$ and $\mathcal{D}$ are restricted to their finitary versions.

4.5 A Hierarchy of Hybrid Automata

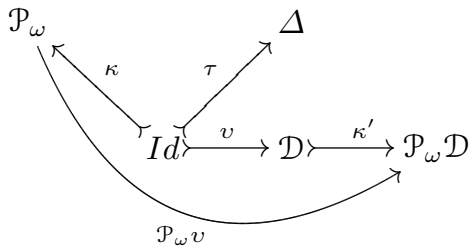
Natural transformations are a suitable mechanism to transform a coalgebra into another of a different transition type, because naturality entails preservation of bisimilarity [Sok05]. The case for reflection, however, is more complex: as described in [Sok05], in **Set** bisimilarity is reflected when the natural transformation is injective (*i.e.*, all its components are injective), and the underlying functor of the resulting system preserves weak pullbacks.

Fortunately, it is known that all polynomial functors, the powerset, and the distribution functor, preserve weak pullbacks (*cf.* [Sok05]). Moreover, preservation of weak pullbacks is closed by composition. Therefore, in many cases checking for reflectivity reduces to checking for injectivity. Actually, this is precisely the case for all variants of $S \to (\mathcal{F}S \times \mathcal{H}O)^I$ considered in this paper.

Observe that from a natural transformation $\tau : \mathcal{F} \to \mathcal{G}$ we can construct the natural transformation $(\tau \times id)^I : (\mathcal{F} \times \mathcal{H}O)^I \to (\mathcal{G} \times \mathcal{H}O)^I$.

Then, given a coalgebra $\alpha : S \to (\mathcal{F} \times \mathcal{H}O)^I$, via the natural transformation above, we define $((\tau \times id)^I)_S \cdot \alpha : S \to (\mathcal{G}S \times \mathcal{H}O)^I$.
Since all internal transition types (functors) considered in this paper preserve weak pullbacks, from the existence of injective natural transformations (between transition types), it is possible generate a hierarchy of systems in terms of their expressive power.

'To be more expressive' here means that looking at an $(\mathcal{F} \times \mathcal{H}O)^I$-coalgebra as a $(\mathcal{G} \times \mathcal{H}O)^I$-coalgebra – through the natural transformation $\tau : \mathcal{F} \to \mathcal{G}$ – never entails *loss of observable information*. In other words, if two states of a $(\mathcal{F} \times \mathcal{H}O)^I$-coalgebra are bisimilar when looking at the latter as a $(\mathcal{G} \times \mathcal{H}O)^I$-coalgebra, then the same is true before the application of τ (*i.e.* coalgebraic bisimilarity is reflected). The hierarchy is expressed in the following diagram of injective natural transformations,



where for any set X, $\tau_X\ x = (x, x)$, $\upsilon_X\ x = \mu$ where $\mu\ x = 1$, $\kappa_X\ x = \{x\}$, and $\kappa'_X\ \mu = \{\mu\}$. 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.

We conclude by mentioning the canonical injective natural transformation $(\mathcal{F} \times \mathcal{H}O) \to (\mathcal{F} \times \mathcal{H}O)^I$ (assuming that $I \neq \emptyset$), which, given an element, returns the constant function over it. This adds to the hierarchy the obvious relation between a family of systems and the corresponding extended version that harbours the input/output dimension.

5 Conclusions and Future Work

Even if hybrid automata are the standard formalism for hybrid systems, their definition often needs to cater for different computational behaviours found in practice. In order to make such a process systematic, this paper proposes a coalgebraic rendering of hybrid automata. This allows the study of several variants of the latter, as well as related notions, (*e.g.*, bisimulation, observational semantics) in a uniform manner, at the same time promoting a black-box perspective in which discrete actions are hidden from the environment while continuous evolutions make up the observable behaviour. Furthermore, this characterises hybrid automata as (coalgebraic) components, in the spirit of [Bar03,HJ11].

Interestingly, a somewhat dual perspective appears in the work of Jacobs [Jac00], where an object-oriented approach for hybrid systems is pursued. More concretely, hybrid systems are viewed there 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. Such a view allows to express physical processes that (continuously) evolve internally, and are possible to interact with at specific instants of time.

It is also relevant to mention the work of Haghverdi *et al.* [HTP05], whose aim is to provide an abstract notion of bisimulation for dynamical, control, and hybrid systems (the latter being understood as hybrid automata). To achieve this, they resort to the notion of an *open map*, which has a close relation to that of coalgebras. Variants of hybrid automata, however, are not taken into consideration.

As future work, we intend to further explore different variants of hybrid automata by varying the functor that gives shape to the internal transitions. For example, arrows of type $S \to (\mathcal{D}S \times \mathcal{H}O)^I$, giving rise to what we call

'*reactive Markov* hybrid automata', deserve an independent study. Other interesting cases are replicating hybrid systems (which we briefly addressed here) and the arrows $S \to \mathcal{W}S \times \mathcal{H}O$ ($\mathcal{W}S = K^S$, for K a set of weights), which makes possible to prescribe costs to discrete transitions and assignments.

Going more generic, and in order to drop the condition as-soon-as (see Sect. 2), one may extend the internal transition type to the 'continuous part' by considering arrows of type $S \to (\mathcal{F}(S \times \mathcal{H}O))^I$ instead. In some cases, however, this may be problematic, as the transition type would need to have a continuous nature. For instance, probabilistic behaviour should be replaced with a stochastic counterpart instead.

On a different note, recall that the results established in this paper allow to define a general characterisation of bisimulation for (different types of) hybrid automata. Such results pave the way to do the same for other notions of bisimulation, one interesting example being *approximate bisimulation* for hybrid automata [GP11].

Finally, a coalgebraic characterisation of hybrid automata makes possible to see them as *hybrid components* (*cf.* [NBHM16]), in the spirit of [Bar03,HJ11]. Generally speaking, this sort of component reproduces the black-box perspective here adopted; and the associated calculus brings to hybrid automata several forms of composition operators (*e.g.*, parallel, pipelining, sum), refinement techniques, and wiring mechanisms, as well as the corresponding algebraic laws. We are currently studying the results brought by this development to the theory of hybrid automata.

Acknowledgements. This work is funded by ERDF - European Regional Development Fund, through the COMPETE Programme, and by National Funds through FCT within project PTDC/EEI-CTP/4836/2014. The first author is also sponsored by FCT grant SFRH/BD/52234/2013, and the second by FCT grant SFRH/BSAB/113890/2015

References

[ACH+95] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**(1), 3–34 (1995)

[AH97] Alur, R., Henzinger, T.A.: Modularity for timed and hybrid systems. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 74–88. Springer, Heidelberg (1997). doi:10.1007/3-540-63141-0_6

[AMP+03] Antoniotti, M., Mishra, B., Piazza, C., Policriti, A., Simeoni, M.: Modeling cellular behavior with hybrid automata: bisimulation and collapsing. In: Priami, C. (ed.) CMSB 2003. LNCS, vol. 2602, pp. 57–74. Springer, Heidelberg (2003)

[Bar03] Barbosa, L.S.: Towards a calculus of state-based software components. J. Univ. Comput. Sci. **9**, 891–909 (2003)

[BCB+09] Bartocci, E., Corradini, F., Di Berardini, M.R., Entcheva, E., Smolka, S.A., Grosu, R.: Modeling, simulation of cardiac tissue using hybrid i, o automata. Theor. Comput. Sci. **410**(33–34), 3149–3165 (2009). Concurrent Systems Biology: To Nadia Busi (1968–2007)

[GP11] Girard, A., Pappas, G.J.: Approximate bisimulation: a bridge between computer science and control theory. Eur. J. Control **17**(5–6), 568–578 (2011)

[Hen96] Henzinger, T.A.: The theory of hybrid automata. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, pp. 278–292. IEEE Computer Society (1996)

[HJ11] Hasuo, I., Jacobs, B.: Traces for coalgebraic components. Math. Struct. Comput. Sci. **21**(2), 267–320 (2011)

[HTP05] Haghverdi, E., Tabuada, P., Pappas, G.J.: Bisimulation relations for dynamical, control, and hybrid systems. Theor. Comput. Sci. **342**(2–3), 229–261 (2005)

[Jac00] Jacobs, B.: Object-oriented hybrid systems of coalgebras plus monoid actions. Theor. Comput. Sci. **239**(1), 41–95 (2000)

[Jac12] Jacobs, B.: Introduction to coalgebra. Towards mathematics of states and observations (2012)

[LLK+99] Liu, J., Liu, X., Koo, T.-KJ., Sinopoli, B., Sastry, S., Lee, E.A.: A hierarchical hybrid system model and its simulation. In: 38th IEEE Decision and Control, vol. 4, pp. 3508–3513. IEEE (1999)

[Nad97] Nadjm-Tehrani, S.: Time-deterministic hybrid transition systems. In: Antsaklis, P., Lemmon, M., Kohn, W., Nerode, A., Sastry, S. (eds.) HS 1997. LNCS, vol. 1567, pp. 238–250. Springer, Heidelberg (1999). doi:10.1007/3-540-49163-5_13

[NB16] Neves, R., Barbosa, L.S.: Hybrid automata as coalgebras (extended version) (2016). http://alfa.di.uminho.pt/~nevrenato/pdfs/HAExtended.pdf

[NBHM16] Neves, R., Barbosa, L.S., Hofmann, D., Martins, M.A.: Continuity as a computational effect. CoRR, abs/1507.03219 (2016). To appear in J. Logical Algebraic Methods Programm

[Rut00] Rutten, J.: Universal coalgebra: a theory of systems. Theor. Comput. Sci. **249**(1), 3–80 (2000). Modern Algebra

[Sok05] Sokolova, A.: Coalgebraic analysis of probabilistic systems. Ph.D. thesis, Technische Universiteit Eindhoven (2005)

[Spr00] Sproston, J.: Decidable model checking of probabilistic hybrid automata. In: Joseph, M. (ed.) FTRTFT 2000. LNCS, vol. 1926, pp. 31–45. Springer, Heidelberg (2000). doi:10.1007/3-540-45352-0_5

[Szy98] Szyperski, C.: Component Software. Beyond Object-Oriented Programming. Addison-Wesley, New York (1998)