COMPOSITIONAL ABSTRACTIONS OF HYBRID CONTROL SYSTEMS

PAULO TABUADA, GEORGE J. PAPPAS, AND PEDRO LIMA

ABSTRACT. Abstraction is a natural way to hierarchically decompose the analysis and design of hybrid systems. Given a hybrid control system and some desired properties, one extracts an abstracted system while preserving the properties of interest. Abstractions of purely discrete systems is a mature area, whereas abstractions of continuous systems is a recent activity. In this paper we present a framework for abstraction that applies to abstract control systems capturing discrete, continuous, and hybrid systems. Parallel composition is presented in a categorical framework and an algorithm is proposed to *construct* abstractions of hybrid control systems. Finally, we show that our abstractions of hybrid systems are compositional.

CDC PAPER ID : CDC01-REG1777

1. INTRODUCTION

Networked, embedded systems are compositions of many complicated subsystems. The analysis and design of such systems is currently limited by their complexity. In order to tackle complexity, these systems are organized in a distributed or hierarchical manner. This structure must be exploited in the analysis and design of such systems in order to scale our models, methods, and tools to real life examples.

Hybrid systems have been used to model networked, embedded systems. Hybrid systems usually come equipped with composition operators which compose subsystems in order to form larger systems [1, 10]. The modeling formalisms also contain abstraction operators which hide irrelevant details resulting in simpler, higher level models.

The notions of composition and abstraction are mature in theoretical computer science, and, in particular, in the areas of concurrency theory [12] [19], and computer aided verification [11]. This has resulted in formal and very meaningful notions of abstraction which are used to tackle exponential explosion of purely discrete systems. Given a discrete system, an *abstraction* is simply a quotient system that preserves some properties of interest while ignoring detail. Language equivalence, simulation, and bisimulation are established notions of abstraction for discrete systems that preserve properties expressed in various temporal logics.

This work was partially performed while the first author was visiting the University of Pennsylvania. This research is partially supported by DARPA under grant F33615-00-C-1707, DARPA Grant N66001-99-C-8510, the University of Pennsylvania Research Foundation, and by Fundação para a Ciência e Tecnologia under grant PRAXIS XXI/BD/18149/98.

For purely continuous systems, the notions of simulation, and bisimulation had no counterparts. Recently, similar notions were introduced in the collection of papers [14, 13, 15]. This research resulted in automatic constructions of abstractions for linear control systems [13], while characterizing abstracting maps that preserve properties of interest such as controllability. This lead to hierarchical controllability algorithms from which we extracted the best performing algorithm [6]. Such notions were furthermore generalized for nonlinear but analytic systems [15].

Based on these results, in [18], we took the first steps towards *automatically* constructing abstractions of hybrid systems while preserving timed languages. Even though *only* the continuous part of the system was abstracted, the important property that needed to be preserved in this abstraction was the detectability of the discrete switching conditions. Related but orthogonal work considers purely discrete abstractions of hybrid systems [2, 4, 5, 16].

The similarities between notions of abstraction for discrete, continuous, and hybrid systems immediately raise the question of a more unified theory of abstraction. In this paper, we begin addressing this issue. We start by first considering a more unified and abstract model for control system. Our abstract control system is inspired by categorical definitions of systems that are as old as [3] and as recent as [17].

We show that purely discrete, continuous, and hybrid systems can be easily captured by our abstract model. Furthermore, at this level of abstraction, one can show many useful properties regarding abstraction or composition that are independent of the discrete, continuous, or hybrid structure of the system. In particular, we use the abstract composition operators defined in [19], and show that system abstraction is compositional. As a result, when abstracting a subsystem of a larger system, we obtain an abstraction of the overall system.

As abstraction clearly depends on the property to be preserved. In this paper, we focus on *simulations* where the properties that are preserved are the trajectories of the original system. In other words, given an original hybrid control system and an abstracting map, which performs state aggregation, we would like to automatically extract another hybrid control system which *simulates* all trajectories of the original system. As a byproduct of preserving trajectories, we also preserve reachability properties. We present an algorithm for the *automatic* abstraction of hybrid systems. Furthermore, our results on compositionality of abstract control systems can be used to show that our hybrid systems abstractions respect composition of hybrid systems of some classes of event synchronization.

The structure of this paper is as follows: In Section 2 we define an abstract notion of control systems which captures discrete, continuous, and hybrid control systems. At this level of generality, we show that abstractions are congruent to a standard composition operator. In Section 3 we specialize these results to hybrid control systems, and present an algorithm for constructing abstractions of hybrid control systems that simulate the trajectories of the original system. Finally, Section 4 lists many interesting areas for future research.

2. Abstract Control Systems

In order to capture continuous, discrete, and hybrid systems under a unified model, we need an abstract definition of control systems. The essence of a control system is reflected into two different aspects: a notion of evolution, and the ability to control the evolution. These two fundamental aspects are captured in the following definition.

Definition 2.1 (Abstract Control System)**.** Let S be a set and let A be a monoid.[1] An abstract control system over S is a partial map $\Phi : S \longrightarrow S^A$ respecting the monoid structure, that is $\Phi_s = \Phi(s) : A \longrightarrow S$ verifies:

1. **Identity:** $\Phi_s(\varepsilon) = s$
2. **Semi-group:** $\Phi_s(a_1 a_2) = \Phi_{\Phi_s(a_1)}(a_2)$ whenever Φ is defined.
3. **Prefix closure:** If $\Phi_s(a_1)$ is defined, then $\Phi_s(a_2)$ is defined, where a_2 is any prefix of a_1.

Intuitively, we can think of the set S as the state space, and the monoid A as the set of possible actions. The monoid structure of A allows us to concatenate inputs. The map Φ assigns to each point $s \in S$ a function from the monoid A to S. Thus Φ represents all the input choices that can be made at the point s. This abstract definition has been used in various contexts, and more recently in the categorical systems theory based on co-algebras [17].

Alternatively, the abstract control system Φ can be put in the more usual from $\Phi : S \times A \longrightarrow S$ and thought of as the action of the monoid A on the set S. It will be also very convenient to think about the domain of Φ, $dom(\Phi) \subseteq S \times A$ as a fiber bundle. The set of actions that are available at each point $s \in S$ are thought as a fiber over the base point s, and denoted by $A(s)$. To get a better felling for this notion of control systems, we show how it applies to three classes of systems.

Deterministic, Finite-State Automata: Let (Q, Σ, δ) be a finite state automaton, where Q is a finite set of states, Σ is a finite set of input symbols, and $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function. Denote by Σ^* the set of all finite words obtained by concatenating elements in Σ. In particular the empty string ε also belongs to Σ^*. With concatenation as a monoid operation, Σ^* can be taken as the action monoid A. The state space is naturally $S = Q$. The transition function δ defines a *unique* map from $Q \times \Sigma^*$ to Q which is just an abstract

[1] A monoid is a semigroup with identity.

control system $\Phi : S \times A \longrightarrow S$. Adding a set of initial and final states would allow us to model deterministic automata in our framework.

Continuous control systems: For simplicity, we consider only time-invariant control systems. Let $S = M$ be a smooth manifold, and let U be the space of admissible inputs. Define the set U^t as:

$$U^t = \{u : [0,t[\longrightarrow U \quad | \quad t \in \mathbb{R}_0^+\} \tag{2.1}$$

An element of U^t is denoted by u^t, and represents a map from $[0,t[$ to U. Consider now the set U^* which is the disjoint union of all U^t for all $0 \leq t < \infty$. The set U^* can be regarded as a monoid under the operation of concatenation, that is if $u^{t_1} \in U^{t_1} \subset U^*$ and $u^{t_2} \in U^{t_2} \subset U^*$ then $u^{t_1}u^{t_2} = u^{t_1+t_2} \in U^{t_1+t_2} \subset U^*$ with concatenation given by:

$$u^{t_1}u^{t_2} = \begin{cases} u^{t_1} & \text{if} \quad 0 \leq t < t_1 \\ u^{t_2} & \text{if} \quad t_1 \leq t < t_1 + t_2 \end{cases} \tag{2.2}$$

The identity element is given by the empty input that is $\varepsilon = u^0$. If we now let $A = U^*$, we can represent any smooth control system $\dot{x} = f(x,u)$ in the form $\Phi : M \times U^* \longrightarrow M$ by defining $\Phi(x, u^{t_i}) = \gamma_x(t_i)$, where $\gamma_x(t)$, $0 \leq t < t_i$ is the integral curve of the vector field $f(\gamma_x(t), u^{t_i})$ with $\gamma_x(0) = x$.

Hybrid Control Systems: Hybrid control systems also fit in the abstract control system framework. The state space of an hybrid control system is usually described as $Q \times M$, where Q is a finite set of states and M a smooth manifold. However it will be convenient to relax this concept and look at the state space as a fiber bundle. Instead of considering the same manifold M for every $q \in Q$ we consider a set of smooth (Hausdorf, second countable) manifolds X_q parameterized by the discrete states and denoted by $X = \{X_q\}_{q \in Q}$. The discrete set Q is thought as the base space, and for each base point $q \in Q$ we attach a fiber X_q. A point in X is represented by the pair (q, x).

As action monoid we will use the set $A = (U^* \cup \Sigma)^*$, considering that $U^* \cap \Sigma = \{\varepsilon\}$. Since the continuous control system will in general be different at each fiber, U will be a finite family of admissible continuous control input spaces parameterized by the discrete states, thats is $U = \{U_q\}_{q \in Q}$. Each element $a \in A$ is a word formed by finite concatenations of elements in $U^* \cup \Sigma$, where consecutive concatenations of elements belonging to U^* have been replaced by their product in the monoid U^*. This concatenation defines the product operation on A. We can now define hybrid control systems as abstract control systems.

Definition 2.2 (Hybrid Control System)**.** An hybrid control system $H = (X, \Sigma, U, \Phi)$ consists of:

- The state space $X = \{X_q\}_{q \in Q}$.
- A finite set of symbols Σ.
- A family of admissible control inputs $U = \{U_q\}_{q \in Q}$.

- A partially defined action Φ of $(U^* \cup \Sigma)^*$ on X such that for all $q \in Q$, there is a set $Inv_q \subseteq X_q$ and for all $x \in Inv_q$, $A(q,x) \cap U^* \neq \{\varepsilon\}$ and $\Phi((q,x),u^t) \in Inv_q$ for all $u^t \in A(q,x)$.

The semantics associated with the evolution from s governed by Φ and controlled by $a \in A$ is the standard transition semantics of hybrid automata [7]. Suppose that $a = u^{t_1}\sigma_1\sigma_2 u^{t_2}$, then $\Phi(s,a) = s'$ means that the system starting at s evolves during t_1 units of time under continuous input u^{t_1}, jumps under input σ_1 and them jumps again under σ_2. After the two consecutive jumps, the system evolves under the continuous control input u^{t_2} reaching s' after t_2 units of time after the last jump. From the hybrid system construction we can clearly extract the purely discrete case (X_q is a singleton, $U_q = \{\varepsilon\}$) as well as the purely continuous case (Q is a singleton, $\Sigma = \{\varepsilon\}$).

2.1. **Control System Abstractions.** We now consider abstractions, and in particular simulation relations, between the general systems considered in Definition 2.1. Every abstract control system Φ_X over a set X with action monoid A_X, induces the relation $R_X \subseteq X \times X$ defined as

$$R_X = \{(x,x') \in X \times X \mid \exists a \in A_X, \Phi_X(x,a) = x'\} \tag{2.3}$$

A notion of simulation can now be introduced for abstract control systems.

Definition 2.3 (Simulations of Abstract Control Systems)**.** Let Φ_X and Φ_Y be two abstract control systems over X and Y with action monoids A_X and A_Y, respectively. Let map $\varphi : X \longrightarrow Y$ be given. Then Φ_Y is a simulation of Φ_X with respect to φ or a φ-simulation if and only if

$$((q,x),(q',x')) \in R_X \;\Rightarrow\; (\varphi(q,x),\varphi(q',x')) \in R_Y \tag{2.4}$$

This definition is weaker than the usual notions of morphisms between transition systems as in [19], since the acting monoid is not required to be preserved. This is important in the case of hybrid control system simulations and will allow us to abstract continuous controls by higher level discrete events. When Φ_Y is a φ-simulation of Φ_X we call Φ_Y a φ-abstraction of Φ_X. Simulations preserve many useful properties. In particular, they preserve reachable sets.

Definition 2.4 (Reachable Space)**.** Let Φ_X be an abstract control system over X with action set A_X. The reachable space from a point $x \in X$, and denoted by $Reach_x(\Phi_X)$ is given by:

$$Reach_x(\Phi_X) = \{x' \in X \mid \Phi_X(x,a) = x' \quad \exists a \in A_X\} \tag{2.5}$$

The reachable space from a set $X' \subset X$ is denoted by $Reach_{X'}(\Phi_X)$ and is defined as:

$$Reach_{X'}(\Phi_X) = \bigcup_{x \in X'} Reach_x(\Phi_X) \tag{2.6}$$

Proposition 2.5. *Let Φ_X and Φ_Y be two abstract control systems on X and Y, respectively. If Φ_Y is a φ-simulation Φ_X, then $\varphi(Reach_x(\Phi_X)) \subseteq Reach_{\varphi(x)}(\Phi_Y)$ for all $x \in X$.*

Proof. Let $x' \in Reach_x(\Phi_X)$. By definition there exists a control action $a \in A$ such that $\Phi_X(x,a) = x'$, therefore $(x,x') \in R_X$. Since Φ_Y is a φ-simulation of Φ_X, we have that $(\varphi(x), \varphi(x')) \in R_Y$, consequently $\varphi(x')$ also belongs to $Reach_{\varphi(x)}(\Phi_Y)$. □

From the above proposition, it is immediate that for all $X' \subseteq X$, $\varphi(Reach_{X'}(\Phi_X)) \subseteq Reach_{\varphi(X')}(\Phi_Y)$.

2.2. **Compositional Abstractions.** In this section, we follow the categorical description of transitions systems in [19], and introduce a notion of parallel composition for abstract control systems. We defer the reader to [9] for further details on the elementary notions of category theory used in this section. The first step of composition combines two abstract control systems into a single one by forming their Cartesian product. Given two abstract control systems $\Phi_X : X \times A_X \to X$ and $\Phi_Y : Y \times A_Y \to Y$ we define their product to be the abstract control system $\Phi_X \times \Phi_Y : (X \times Y) \times (A_X \otimes A_Y) \to (X \times Y)$, where we have denoted by $\otimes$ the direct product of monoids[2]. The trajectories of the product control system consist of all possible combinations of the trajectories of the initial control systems. The product can also be defined in a categorical manner.

Definition 2.6 (Product of abstract control systems)**.** Let $\Phi_X : X \times A_X \to X$ and $\Phi_Y : Y \times A_Y \to Y$ be two abstract control systems. The product of these abstract control systems is a triple $(\Phi_X \times \Phi_Y, \pi_X, \pi_Y)$ where $\Phi_X \times \Phi_Y$ is an abstract control system and $\pi_X : X \times Y \to X$ and $\pi_Y : X \times Y \to Y$ are projection maps such that Φ_X is a π_X-simulation of $\Phi_X \times \Phi_Y$, Φ_Y is a π_Y-simulation of $\Phi_X \times \Phi_Y$, and such that for any other triple (Φ_Z, p_X, p_Y) of this type there is one and only one map $\zeta : Z \to X \times Y$ such that $\Phi_X \times \Phi_Y$ is a ζ-simulation of Φ_Z, and that the following diagram commutes:

$$\begin{tikzcd}
\Phi_X & \Phi_X \times \Phi_Y \arrow[l, "\pi_X"'] \arrow[r, "\pi_Y"] & \Phi_Y \\
& \Phi_Z \arrow[ul, "p_X"] \arrow[u, "\zeta"] \arrow[ur, "p_Y"'] &
\end{tikzcd} \tag{2.7}$$

The map $\zeta : Z \to X \times Y$ is easily seen to be given by $\zeta = (p_X, p_Y)$. This definition of product may seem unnecessarily abstract and complicated at the first contact, it will, however, render the proof of the main result on the compatibility of parallel composition an almost trivial task.

[2]Note that the choice of $A_X \otimes A_Y$ for the monoid on the product control system is somewhat arbitrary since the notion of simulation does not distinguish between abstract control systems with the same relation R and different action monoids.

In the product system, the behavior of one system does not influence the behavior of the other system. To model the fact that in general the behavior of one system interacts with the behavior of the other, we need to refine the behaviors of the product $\Phi_X \times \Phi_Y$, through the operation of restriction.

Given a set $L \subseteq W$ we define the restriction of control system $\Phi_W : W \times A_W \longrightarrow W$ to L as a new control system $\Phi_W|_L : L \times A_W \longrightarrow L$ which is given by $\Phi_L(x, a) = \Phi_W(x, a)$ iff both x and $\Phi_W(x, a)$ belong to L. This operation also admits a categorical characterization.

Definition 2.7 (Restriction of abstract control systems)**.** Let $\Phi_W : W \times A_W \longrightarrow W$ be an abstract control system and let L be a subset of W. The restriction of Φ_W to L is a pair $(\Phi_W|_L, i_L)$ where $\Phi_W|_L$ is an abstract control system and $i_L : L \hookrightarrow W$ is an inclusion map such that Φ_W is a i_L-simulation of $\Phi_W|_L$, and for any other pair (Φ_Z, j_Z) of this type with $j_Z(Z) = i_L(L)$ there is one and only one map η such that $\Phi_W|_L$ is a η-simulation of Φ_Z, and the following diagram commutes:

$$\begin{tikzcd} \Phi_W|_L \arrow[r, "i_L"] & \Phi_W \\ \Phi_Z \arrow[u, "\eta"] \arrow[ur, "j_Z"'] & \end{tikzcd} \tag{2.8}$$

The inclusion map $i_L : L \hookrightarrow W$ is the map sending $l \in L$ to $i_L(l) = l \in W$, and consequently the map η is trivially given by $\eta = j_Z$. With the notions of products and restriction at hand, we can now define a general operation of parallel composition.

Definition 2.8 (Parallel composition)**.** Let $\Phi_X : X \times A_X \longrightarrow X$ and $\Phi_Y : Y \times A_Y \longrightarrow Y$ be two abstract control systems and consider a set $L \subseteq X \times Y$. The parallel composition of Φ_X and Φ_Y with synchronization over L is the abstract control system denoted by $\Phi_X \circ_L \Phi_Y$ and defined as:

$$\Phi_X \circ_L \Phi_Y = (\Phi_X \times \Phi_Y)|_L \tag{2.9}$$

This notion of parallel composition defined above is general enough to capture standard notions of synchronous composition as well as asynchronous composition by choosing $L = X \times Y$.

Any theory of abstraction only makes sense if it is compositional. Since the system being analyzed is given by the composition of several smaller subsystems, one can perform abstraction of individual subsystems, resulting in abstractions of the overall system. This is now asserted in the next theorem which is inspired from [19] .

Theorem 2.9 (Compositionality of abstractions)**.** *Given abstract control systems Φ_X, Φ_Z (which is a φ_X-simulation of Φ_X), Φ_Y, Φ_W (which is a φ_Y-simulation of Φ_Y) and the set $L \subseteq X \times Y$ we have that the parallel composition of the simulations Φ_Z and Φ_W is a simulation of the parallel composition of Φ_X with Φ_Y. More* *precisely we have the following equality:*

$$\varphi(\Phi_X \circ_L \Phi_Y) = \varphi_X(\Phi_X) \circ_{\varphi(L)} \varphi_Y(\Phi_Y) \tag{2.10}$$

that can also be expressed through the following commutative diagram:

$$\begin{array}{ccccc} \Phi_Z & \Phi_W & \xrightarrow{\circ_{\varphi(L)}} & \Phi_Z \circ_{\varphi(L)} \Phi_W \\ \varphi_X \uparrow & \varphi_Y \uparrow & & \uparrow \varphi = (\varphi_X, \varphi_Y) \\ \Phi_X & \Phi_Y & \xrightarrow[\circ_L]{} & \Phi_X \circ_L \Phi_Y \end{array}$$

Proof. Consider the product $(\Phi_Z \times \Phi_W, \pi_Z, \pi_W)$, we know that for the triple $(\Phi_X \times \Phi_Y, \varphi_X, \varphi_Y)$ there is one and only one map ζ that makes diagram 2.7 commute, and that map is given by $\zeta = \varphi = (\varphi_X, \varphi_Y)$, meaning that $\Phi_Z \times \Phi_W$ is a φ-simulation of $\Phi_X \times \Phi_Y$. Considering the inclusion map $\varphi \circ i_L : L \longrightarrow Z \times W$ which verifies $\varphi \circ i_L(L) = i_{\varphi(L)}(\varphi(L))$ we see that $\Phi_Z \times \Phi_W$ is a $\varphi \circ i_L$-simulation of $\Phi_X \circ_L \Phi_Y$ and by definition of restriction there is a single map η that makes diagram 2.8 commutative. Clearly we have that $\eta = \varphi$ and we conclude that:

$$\varphi(\Phi_X \circ_L \Phi_Y) = \varphi_X(\Phi_X) \circ_{\varphi(L)} \varphi_Y(\Phi_Y) \tag{2.11}$$

which concludes the proof. □

The above result was stated for composition of two abstract control systems but it can be easily extended to any finite number of abstract control systems. The relevance of the result lies in the fact that, in general, it is much easier to abstract each individual subsystem and by composition obtain an abstraction of the overall system.

So far we have presented the main results on compositionality of abstractions at a fairly abstract (categorical) level. This was clearly helpful in providing a unifying framework for discrete, continuous and hybrid control systems. Furthermore the presented results remain valid for any instantiation of a particular class of control systems. Nevertheless we are interested in providing *constructive* procedures for computing abstractions of hybrid control systems. To achieve this we specialize in the next section our results to hybrid control systems and propose concrete algorithms to compute abstractions and parallel compositions.

3. Compositional Abstractions of Hybrid Control Systems

3.1. **Hybrid Control Systems Abstractions.** Simulations of hybrid control systems are a simple instantiation of the previously introduced notion of simulation. However, hybrid control systems usually come equipped with a set of initial conditions which must also be related with the set of initial conditions of its simulation. The proper relation is expressed as follows:

Definition 3.1 (Simulations of Hybrid Control Systems)**.** Let $H_X = (X_0, X, U_X, \Sigma_X, \Phi_X)$ and $H_Y = (Y_0, Y, U_Y, \Sigma_Y, \Phi$ be two hybrid control systems over X and Y respectively. Let $\varphi : X \longrightarrow Y$ be a given map, then H_Y is a φ-simulation of H_X iff

1. $\varphi(X_0) \subseteq Y_0$.
2. $((q, x), (q', x')) \in R_X \;\Rightarrow\; (\varphi(q, x), \varphi(q', x')) \in R_Y$

The goal of obtaining algorithmic procedures for computing abstractions guide us to more amenable characterizations of hybrid control systems. This is where we drift from the standard computer science approaches purely based on transition semantics. A first step in this direction is given by characterizing hybrid control systems in terms of its generators.

Proposition 3.2 (Hybrid Generators)**.** *A set of initial conditions $X_0 \subseteq X$, a map $\delta : X \times \Sigma_X \longrightarrow X$ and a smooth family of maps $f = \{f_q\}_{q \in Q}$, $f_q : X_q \times U_q \longrightarrow TX_q$ defined on a open subset of X_q for each $q \in Q$ uniquely define a hybrid control system H_X. The maps δ and f are called the discrete and continuous generators of H_X, respectively.*

Proof. We start by showing that δ extends uniquely to $\delta^* : X \times \Sigma_X^* \longrightarrow X$. This action is obtained from δ by:

$$\delta^*((q, x), \varepsilon) = (q, x) \tag{3.1}$$

$$\delta^*((q, x), \sigma_1\sigma) = \delta^*(\delta^*((q, x), \sigma_1), \sigma) \qquad \sigma_1\sigma \in \Sigma_X^*,\ \sigma_1 \in \Sigma_X \tag{3.2}$$

defining δ^* uniquely.

A similar construction is possible for f. Denote by C_q the open subset of X_q where each f_q is defined. A unique action ${f_q}^* : cl(C_q) \times U_q^* \longrightarrow cl(C_q)$ can be obtained from f_q, where we denote by $cl(C_q)$ the closure of C_q in the topology of X_q. This is accomplished by defining ${f_q}^*$ as:

$${f_q}^*(x, u^{t_i}) = \gamma_x(t_i) \tag{3.3}$$

where $\gamma_x(t)$ is the integral curve of the vector field $f_q(\gamma_x(t), u^t)$ satisfying $\gamma_x(0) = x$ for $0 \le t < t_i$. By existence and uniqueness of integral curves of vector fields follows existence and uniqueness of the action ${f_q}^* : C_q \times U_q^*$ $\longrightarrow C_q$ since f_q is smooth. Moreover we can extend ${f_q}^* : C_q \times U_q^* \longrightarrow C_q$ to ${f_q}^* : cl(C_q) \times U_q^* \longrightarrow cl(C_q)$ in a unique way by continuity.

We can now combine δ^* and ${f_q}^*$ to get an hybrid control system $H_X = (X_0, X, \Sigma_X, U_X, \Phi_X)$ with $U_X = \{U_q\}_{q\in Q}$. Let $a_1 \in U^* \cup \Sigma_X$ and define:

$$\Phi_X((q,x),a_1) = \begin{cases} (q,x) & \text{if} \quad a_1 = \varepsilon \\ \delta^*((q,x),a_1) & \text{if} \quad a_1 \in \Sigma_X \\ {f_q}^*(x,a_1) & \text{if} \quad a_1 \in U^* \end{cases} \tag{3.4}$$

for a general $a \in (U^* \cup \Sigma_X)^*$ assume that $a = a_1 a_2$ with $a_1 \in U^* \cup \Sigma_X$, then Φ_X is given by:

$$\Phi_X((q,x),a) = \Phi_X((q,x),a_1 a_2) = \Phi_X(\Phi_X((q,x),a_1),a_2) \tag{3.5}$$

This construction always provides a unique abstract control system Φ_X. □

This result tells us that it is enough to work with vector fields and single event jumps, which is how hybrid automata are usually defined in the literature [7]. This means that a hybrid control system can equivalently be denoted by the tuple $H_X = (X, X_0, \Sigma_X, U_X, \delta_X, f_X)$. This representation of hybrid control systems will allow constructive methods to generate abstractions by combining discrete and continuous abstraction methodologies.

We have already introduce all the required tools to extract an abstraction of any given hybrid control system H_X with respect to a map φ. This map will aggregate several states from H_X to its φ-simulation, in particular we allow φ to map continuous states into discrete ones. This feature will be fundamental to reduce the complexity of hybrid control systems since by aggregating continuous states into discrete ones we are also trading continuous dynamics by discrete dynamics thereby considerably simplifying the analysis or synthesis processes. When this is the case we assume that there exists a finite partition of some or several fibers and that φ maps each partition block into discrete state. More formally, we denote a map $\varphi : X \longrightarrow Y$ between hybrid state spaces X and Y by $\varphi = (\varphi_D, \varphi_C)$ where φ_D and φ_C gives the discrete and continuous part of the state, respectively. This map naturally defines an equivalence relation on X by declaring states (q,x) and (q',x') to be equal iff $\varphi(q,x) = \varphi(q',x')$. The equivalence classes (or partition blocks) on X are given by $\varphi^{-1}(p)$, and represent all the data that is going to be aggregated into the discrete state p. Since these equivalence classes may be spread by several discrete state we restrict $\varphi^{-1}(p)$ to a particular state q' and denote it by $\varphi^{-1}(p)|_q$ when referring to the set of continuous points over q that are aggregated into p, as displayed in Figure 1. We assume that the partition in the continuous state space over each discrete state is regular enough so that it induces a well defined transition system across its blocks. A sufficient condition is given by the use of subanalytic stratifications to define the partition as explained in [8]. An abstracting algorithm allowing continuous to discrete aggregation can be formulated as follows:

Algorithm 3.3 (Abstracting Algorithm)**. Input data:** $H_X = (X_0, X, \Sigma_X, U_X, \delta_X, f_X)$, $\varphi : X \longrightarrow Y$

1. $Y := \varphi(Y)$
2. $Y_0 := \varphi(X_0)$
3. $R_Y^{f\to\delta} := \{((p,y),(p',y)) \in Y \times Y \mid \exists x \in \varphi_D^{-1}(p)|_q \; \exists u \in U_X(q,x)$ *such that* $f_q(x,u)$ *is transversal to the border from* $\varphi_D^{-1}(p)|_q$ *to* $\varphi_D^{-1}(p)|_{q'}$ *and* $y = \varphi_C(q,x)\}$
4. $R_Y^{\delta} := \varphi(R_X^{\delta}) \cup R_Y^{f\to\delta}$ *where* R_X^{δ} *is the relation induced by* δ *as in (2.3).*
5. $\Sigma_Y :=$ *a set of labels such that* $|\Sigma_Y| = |R_Y^{\delta}|$.
6. $\delta_Y : Y \times \Sigma_Y \longrightarrow Y$ *is the discrete generator defined as* $\delta_Y((p_i,y_i),\sigma_{ij}) = (p_j,y_j)$ *iff* $((p_i,y_i),(p_j,y_j)) \in R_Y^{\delta}$ *with* $\sigma_{ij} \in \Sigma_Y$.
7. $f_{Yp} :=$ *is the continuous generator* φ_C*-related to* f_{Xq} *with domain* $Int(\varphi_D^{-1}(p)|_q)$ *for every* $q \in Q$ *and* $p \in P$.

Output data: $H_Y = (Y_0, Y, \Sigma_Y, U_Y, \delta_Y, f_Y)$

The last step of the algorithm requires the computation of an abstraction of a continuous control system. Recent results on this subject are reported in [13] for linear systems and in [15] for nonlinear analytic systems. For the sake of simplicity we shall outline the procedure for linear systems only. Given a linear control system of the form $\dot{x} = Ax + Bu$ with matrices A and B of the appropriate dimensions and input vector u we can determine an abstractions based on a linear aggregating map of the form $y = Cx$. The abstracting linear

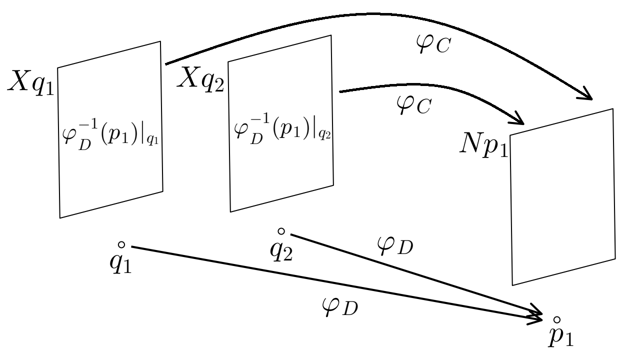

FIGURE 1. The sets $\varphi_D^{-1}(p)|_q$.

control system $\dot{y} = Fy + Gv$ is determined by computing the matrices F and G as:

$$F = CAC^{+} \qquad G = [CB \; CAv_1 \; \ldots \; CAv_r] \tag{3.6}$$

where the vectors $v_1, v_2, \ldots, v_r$ span $Ker(C)$.

The above algorithm does compute a simulation of H_X as asserted in the next theorem:

Theorem 3.4. *Let H_X be an hybrid control system over X and $\varphi : X \longrightarrow Y$ an abstracting map such that each restriction of $\varphi_C(q, -) : M_q \longrightarrow Y$ to $Int(\varphi_D^{-1}(p)|_q)$ is smooth for every $p \in P$ and $q \in Q$. Then hybrid control system H_Y obtained through Algorithm 3.3 is a φ-simulation of H_X.*

Proof. The data resulting from Algorithm 3.3 define and hybrid control system since they consist of a set of initial conditions an the hybrid generators δ_Y and f_Y which extend to an hybrid control system as asserted in Proposition 3.2. The reaming proof will be divided in four parts. First we will show that the image of elements of R_X (originated by discrete jumps) under φ is in R_Y. Then we will consider elements of R_X originated by continuous flows inside a single partition block. On the third part of the proof we will address the case where continuous flows cross partition blocks. Finally it will be show that the first three cases also imply that any element of R_X induced by any sequence of jumps and continuous flows has its image under φ in R_Y.

Discrete Jumps

Suppose that $((q, x), (q', x')) \in R_X$ was originated by a sequence of jumps, say $(q, x) \rightarrow (q_1, x_1) \rightarrow (q_2, x_2) \rightarrow \ldots \rightarrow (q', x')$. Since by step 3 of the Algorithm 3.3, for every jump $((q_i, x_i), (q_j, x_j)) \in R_X^d$ we have that $(\varphi(q_i, x_i), \varphi(q_j, x_j)) \in R_Y^d$ and consequently there is some $\sigma \in \Sigma_Y$ possibly depending on i and/or j such that $\phi_Y^d((q_i, x_i), \sigma) = (q_j, x_j)$. Finally Proposition 3.2 ensures that $(\varphi(q, x), \varphi(q', x')) \in R_Y$.

Continuous flows inside a partition block

The continuous abstraction (see [13] and [15]) and step 5 in Algorithm 3.3 assures us that every $((q, x), (q, x')) \in R_X$ originated by a continuous flow inside a partition block verifies $(\varphi(q, x), \varphi(q, x')) \in R_Y$.

Continuous flows crossing partition blocks

Any pair $((q, x), (q, x')) \in R_X$ induced by a continuous flow crossing the boundary between blocks $\varphi_D^{-1}(p)|_q$ and $B(q, p')$ and not crossing any other border verifies $(\varphi(q, x), \varphi(q, x')) \in R_Y$. This assertion follows from transitivity of R_Y and

1. $((p, \varphi_C(q, x)), (p, \varphi_C(q, x_b))) \in R_Y$
2. $((p, \varphi_C(q, x_b)), (p', \varphi_C(q, x_b))) \in R_Y$
3. $((p', \varphi_C(q, x_b)), (p', \varphi_C(q, x'))) \in R_Y$

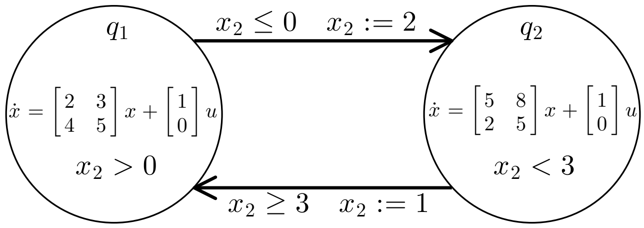

FIGURE 2. Hybrid control system H_X.

where we have denoted by x_b a point belonging to the border between the partition blocks. Let's prove now the enumerated statements. The first statement follows from the continuous abstraction theory of Pappas and the fifth step in Algorithm 3.3. If the flow crosses the border between p and p' then the associated vector field is transversal to the border, and steps 2 and 3 of Algorithm 3.3 imply the second statement. Finally the third statement holds by the same reason of the first one.

Any sequence of discrete jumps and continuous flows

Transitivity of the relation R_Y and the above arguments guarantee that any pair $((q, x), (q', x')) \in R_X$ induced by any sequence of discrete jumps and continuous flows (crossing partition blocks or not) verify $(\varphi(q, x), \varphi(q', x')) \in R_Y$. Since all the initial conditions of H_X are contained in the initial conditions of H_Y by step 1 of Algorithm 3.3, H_Y is a φ-simulation of H_X. □

As an illustration of the construction given by Algorithm 3.3 we present a simple example. Consider the hybrid control system H_X shown in Figure 2. Using as abstracting map $\varphi = (q, x_2)$, that is $\varphi_D : \{q_1, q_2\} \longrightarrow \{q\}$, $\varphi_D(q_i) = q$ and $\varphi_C : \mathbb{R}^2 \longrightarrow \mathbb{R}$, $\varphi_C(x_1, x_2) = y$ we get an abstraction with a single discrete state q. The continuous dynamics in location q_1 is abstracted to $\dot{y} = 3y + 4v$ and on location q_2 to $\dot{y} = 3y + 2v'$ so that the resulting continuous control system is given by $\dot{y} = 3y + 2v' + 4v$. Introducing a new control input defined as $w = 2v' + 4v$ we simplify the dynamics to $\dot{y} = 3y + w$. The discrete jumps triggered at $x_2 \leq 0$ and $x_2 \geq 3$ are mapped by φ to $y \leq 0$ and $y \geq 3$, respectively so that the new invariant is now given by $0 < y < 3$. The reset maps are similarly transformed as can be seen in Figure 3.

3.2. **Compositional hybrid abstractions.** The results presented for compositionality of abstract control systems in Section 2.2 also carry over to hybrid control systems. We consider standard notions of composition

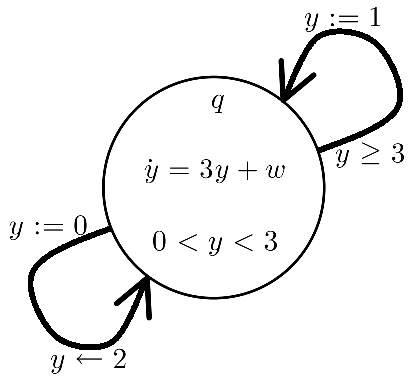

FIGURE 3. Hybrid control system H_Y obtained by Algorithm 3.3, a φ-simulation of H_X.

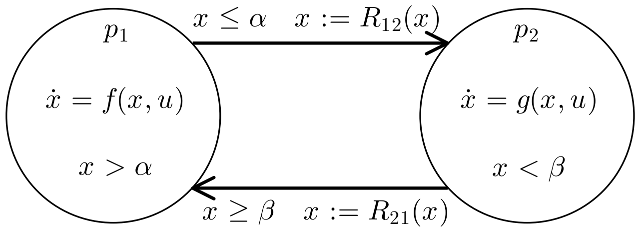

FIGURE 4. Hybrid control system H_Z.

with event synchronization as defined for example in [7]. When discrete events are triggered by the continuous dynamics, event synchronization can easily be accomplished by restricting the product hybrid control system to a subset of the state space. Consider for example hybrid control system H_Y from the previous example and hybrid control system H_Z presented in Figure 4 where $f(x, u)$ and $g(x, u)$ are smooth control systems and R_{12} and R_{21} are reset maps. Suppose that one wants to compose H_Y with H_Z while synchronizing the event that occurs on H_Y when $y \leq 0$ with the jump from p_1 to p_2. This jump happens when $x \leq \alpha$ which should only happen when $y \leq 0$. This synchronization can be accomplished by restricting the behaviors of the product

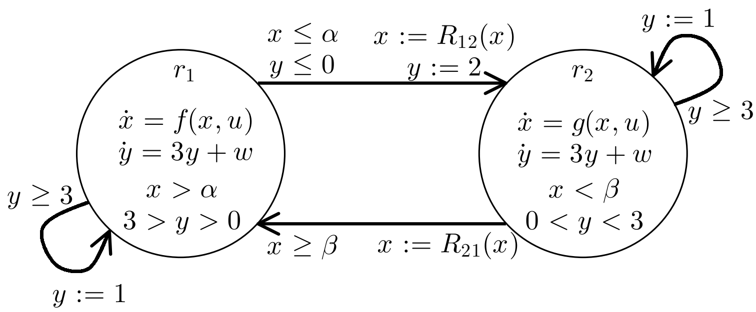

FIGURE 5. Hybrid control system $H_Y \circ_L H_Z$.

hybrid control system to the set:

$$\begin{aligned} L &= \{(q,y,p,x) \in Y \times Z \mid p = p_1 \wedge y > 0 \wedge x > \alpha\} \\ &\cup \{(q,y,p,x) \in Y \times Z \mid p = p_1 \wedge y \leq 0 \wedge x \leq \alpha\} \\ &\cup \{(q,y,p,x) \in Y \times Z \mid p = p_2 \wedge y > 0\} \end{aligned}$$

The resulting hybrid control system $H_Y \circ_L H_Z$, displayed in Figure 5 could also be obtained from H_X and H_Z as asserted by Theorem 2.9. As the reader can easily verify by parallel composing H_X and H_Z with synchronization over L followed by an abstraction based on the map $\varphi' = (\varphi, id_Z) : X \times Z \longrightarrow Y \times Z$ we get the same hybrid control system. However by proceeding this way the abstraction process would be much more complex since $H_X \times H_Z$ is an hybrid control system with 4 discrete states and 4 continuous states in each discrete location.

4. Conclusions and Future Work

We have considered an unified framework for a general class of control systems that captures discrete, continuous and hybrid control systems. In this framework we presented the simulation notion of abstraction that preserves trajectories. A categorical notion of parallel composition was also presented. It was shown that these notions are compatible, that is the composition of abstractions is an abstraction of the composition.

These notions were then instantiated for hybrid control systems where a concrete algorithm was presented to automatically extract abstractions. It was also shown how synchronization of forced jumps can be captured in our framework by performing restriction on the state space. An example has demonstrated the concepts and illustrated the simplification introduced in the abstraction process by taking advantage of parallel composition with synchronization between blocks.

Future research will consider several other important properties of hybrid systems need to be considered. For example, it is crucial to determine when abstractions of non-zeno hybrid control systems are non-zeno. These issues are currently under investigation as well as design methodologies that take advantage of the hierarchical and compositional structure of hybrid control systems.

Acknowledgments: The authors would like to thank Esfandiar Haghverdi for extremely stimulating discussions on category theory, and its use for hybrid systems.

References

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

[2] Rajeev Alur, Tom Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[3] M.A. Arbib and E. Manes. Machines in a category : An expository introduction. *SIAM Review*, 16(2):163–192, April 1974.

[4] P.E. Caines and Y.J. Wei. Hierarchical hybrid control systems: A lattice theoretic formulation. *IEEE Transactions on Automatic Control : Special Issue on Hybrid Systems*, 43(4):501–508, April 1998.

[5] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control : Special Issue on Hybrid Systems*, 43(4):564–568, April 1998.

[6] P. M. Van Dooren. The generalized eigenstructure problem in linear system theory. *IEEE Transactions on Automatic Control*, 26(1):111–129, 1981.

[7] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[8] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. Subanalytic stratifications and bisimulations. In T. Henzinger and S. Sastry, editors, *Hybrid Systems : Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 205–220. Springer Verlag, Berlin, 1998.

[9] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[10] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996.

[11] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer Verlag, New York, 1995.

[12] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[13] George J. Pappas, Gerardo Lafferriere, and Shankar Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, June 2000.

[14] George J. Pappas and Shankar Sastry. Towards continuous abstractions of dynamical and control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 329–341. Springer Verlag, Berlin, Germany, 1997.

[15] George J. Pappas and Slobodan Simic. Consistent hierarchies of nonlinear abstractions. In *Proceedings of the 39th IEEE Conference in Decision and Control*. Sydney, Australia, December 2000.

[16] J. Raisch and S.D. O'Young. Discrete approximations and supervisory control of continuous systems. *IEEE Transactions on Automatic Control : Special Issue on Hybrid Systems*, 43(4):569–573, April 1998.

[17] J.J. Rutten. Universal coalgebra : A unified theory of systems. *Theoretical computer science*, 2000.

[18] P. Tabuada and George J. Pappas. Hybrid abstractions that preserve timed languages. In *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*. Springer Verlag, 2001.

[19] Glynn Winskel and Mogens Nielsen. *Handbook of Logic and Foundations of Theoretical Computer Science*, chapter Models for Concurrency. Oxford University Press, 1994.

Instituto de Sistemas e Robótica, Instituto Superior Técnico, Torre Norte - Piso 6, Av. Rovisco Pais, 1049-001 Lisboa - Portugal

E-mail address: tabuada@isr.ist.utl.pt

Department of Electrical Engineering, 200 South 33rd Street, University of Pennsylvania, Philadelphia, PA 19104

E-mail address: pappasg@ee.upenn.edu

Instituto de Sistemas e Robótica, Instituto Superior Técnico, Torre Norte - Piso 6, Av. Rovisco Pais, 1049-001 Lisboa - Portugal

E-mail address: pal@isr.ist.utl.pt