Composing Abstractions of Hybrid Systems

Paulo Tabuada[1], George J. Pappas[1], and Pedro Lima[2]

[1] Department of Electrical Engineering, University of Pennsylvania
Philadelphia, PA 19104, {tabuadap,pappasg}@seas.upenn.edu

[2] Instituto de Sistemas e Robótica, Instituto Superior Técnico
1049-001 Lisboa - Portugal, pal@isr.ist.utl.pt

Abstract. The analysis and design of hybrid systems must exploit their hierarchical and compositional nature of in order to tackle complexity. In previous work, we presented a hierarchical abstraction framework for hybrid control systems based on the notions of simulation and bisimulation. In this paper, we build upon our previous work and investigate the compositionality of our abstraction framework. We present a composition operator that allows synchronization on inputs and states of hybrid systems. We then show that the composition operator is compatible with our abstraction framework in the sense that abstracting subsystems will the result in an abstraction of the overall system.

1 Introduction

The complexity of hybrid systems analysis and design motivate the development of methods and tools that scale well with dimension and exploit system structure. Hierarchical decompositions model hybrid systems using a hierarchy of models at different layers of abstraction. Analysis tasks are then performed on simpler, abstracted models that are equivalent with respect to the relevant properties. Design also benefits from this approach since the design starts at the top of the hierarchy on a simple model and is then successively refined by incorporating the modeling detail of each layer.

In addition, as systems are usually compositions of subsystems, one must take advantage of the compositional structure of hybrid systems. We seek, therefore, to take advantage of this compositional structure of hybrid systems to simplify the computation of abstractions. This simplification comes from the fact that it is much simpler to abstract subsystems individually and then interconnect them in order to obtain an abstraction, rather than to extract an abstraction of the system as a whole. In order to accomplish this, compositional operators need to be compatible with abstraction operators.

The notions of composition and abstraction are mature in theoretical computer science, and, in particular, in the areas of concurrency theory [10], [19], and computer aided verification [9]. Notions of abstraction such as language inclusion, simulation relations, and bisimulation relations have been considered in the context of hybrid systems. A formal model for hybrid systems allowing composition was proposed in [8], compositional refinements in a hierarchical setting are discussed in [2], and assume guarantee proof rules are presented in [4].

For purely continuous systems, the notions of simulation, and bisimulation had not received much attention [18]. Recently, similar notions were introduced in [11, 12] which has resulted in constructions of abstractions for linear control systems [11], and nonlinear control systems [12] while characterizing abstracting maps that preserve properties of interest such as controllability. Based on these results, in [16], we took the first steps towards constructing abstractions of hybrid systems while preserving timed languages. This allowed us to introduce in [17] an abstract notion of control systems comprising discrete, continuous and hybrid systems. This abstract framework was the natural setting to understand abstractions of hybrid control systems.

In this paper, we extend the hierarchical approach described in [17] towards compositionality. Following the approach described in [19], we introduce a general composition operator modeling the interconnection of subsystems and relate compositionality with abstractions. We prove that simulations and bisimulations of hybrid systems are compositional, and we also give necessary and sufficient conditions for bisimulations to be compositional.

This paper is structured as follows. In Section 2 we review the abstract control systems framework introduced in [17] and introduce the notions of simulation and bisimulation. In Section 3 we introduce a composition operator based on [19], modeling the interconnection of subsystems and relate compositionality with abstractions. We prove the main results of the paper showing that abstractions are compositional. We conclude at Section 4 by providing some topics for future research. In Appendix A we collect some mathematical facts and notational issues, and Appendix B contains the proofs of all the results.

2 Abstract Control Systems

In [17], we presented an abstract control systems framework which allows the treatment of discrete, continuous, and hybrid control systems in a unified way. This approach differs from other attempts of unification [7, 14] by regarding systems as *control* systems. We start by looking at discrete and continuous systems to gain some motivation for the general case.

Discrete Control Systems: Let (Q, Σ, δ) be a discrete labeled transition system, 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 next-state function. For simplicity, we restrict to deterministic transition systems, and note that δ is in general a partial function. Let us denote by Σ^* the set of all finite strings obtained by concatenating elements in Σ. In particular the empty string ε also belongs to Σ^*. Regarding concatenation of strings as a map from $\Sigma^* \times \Sigma^*$ to Σ^* we can give Σ^* the structure of a monoid. Furthermore, it is well known from automata theory [5], that the transition function δ defines a *unique* partial map from $Q \times \Sigma^*$ to Q satisfying the following properties:

$$\delta^*(q, \varepsilon) = q \tag{1}$$

$$\delta^*(q, \sigma_1\sigma_2) = \delta^*(\delta^*(q, \sigma_1), \sigma_2) \tag{2}$$

A similar description of control system can also be given.

Continuous Control Systems: Let U be the space of admissible control inputs. Define the set U^t as:

$$U^t = \{u : [0,t[\longrightarrow U \quad | \quad [0,t[\subseteq \mathbb{R}_0^+\} \tag{3}$$

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 $t \in \mathbb{R}_0^+$:

$$U^* = \coprod_{t\in\mathbb{R}_0^+} U^t \tag{4}$$

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:

$$u^{t_1}u^{t_2}(t) = \begin{cases} u^{t_1}(t) & \text{if} \quad 0 \leq t < t_1 \\ u^{t_2}(t-t_1) & \text{if} \quad t_1 \leq t < t_1 + t_2 \end{cases} \tag{5}$$

The identity element is given by the empty input, that is $\varepsilon = u^0$. Let $\dot{x} = f(x,u)$ be a smooth control system, where $x \in M$, a smooth manifold and $u \in U$, the set of admissible inputs. Choosing an admissible input trajectory u^t, $f(x,u^t)$ is a well defined vector field and as such it induces a flow which we denote by $\gamma_x : [0,t[\longrightarrow M$, such that $\gamma_x(0) = x$. We thus see that a smooth control system defines a partial map:

$$\begin{aligned} \varPhi : M \times U^* &\longrightarrow M \\ (x,u^t) &\mapsto \gamma_x(t) \end{aligned} \tag{6}$$

satisfying:

$$\varPhi(x,\varepsilon) = \varPhi(x,u^0) = \gamma_x(0) = x \tag{7}$$

$$\varPhi(x,u^{t_1}u^{t_2}) = \gamma_x(t_1+t_2) = \gamma_{\gamma_x(t_1)}(t_2) = \varPhi(\varPhi(x,u^{t_1}),u^{t_2}) \tag{8}$$

We think of the monoid as the set of control actions available to influence the evolution of the system. In many cases, however, these available actions change from state to state. This dependence of the available actions on the states forces us to work with generalized monoids, see Appendix A for the correct definition.

Definition 1 (Abstract Control System). *Let S be a set and $\mathcal{M}$ a generalized monoid over S. An abstract control system over S is a map $\varPhi : \mathcal{M} \longrightarrow S$ respecting the monoid structure, that is:*

1. **Identity:** $\varPhi(s,\varepsilon) = s$
2. **Semi-group:** $\varPhi(s,a_1a_2) = \varPhi(\varPhi(s,a_1),a_2)$

We now show how this definition is general enough to cover also hybrid control systems.

Hybrid Control Systems: The state space of an hybrid control system is a set of smooth manifolds X_q parameterized by the discrete states $q \in Q$, denoted by $X = \{X_q\}_{q\in Q}$. A point in X is represented by the pair (q, x). The set of available actions at each point is described by a subset of the following monoid:

$$\mathcal{M} = \coprod_{n\in\mathbb{N}_0} (U^* \cup \Sigma^*)^n \tag{9}$$

assuming that $U^* \cap \Sigma^* = \{\varepsilon\}$ and regarding U^* and Σ^* simply as sets. Let us elaborate on the product operation on $\mathcal{M}$. This operation is defined as the usual concatenation and therefore it requires finite length strings. To accommodate this requirement and still be able to have an infinite number of concatenations of elements in U^* we proceed as follows. Suppose that we want to show that $\sigma_1 u^{t_1} u^{t_2} \ldots u^{t_k} \ldots \sigma_2$ belongs to $\mathcal{M}$, where t_k is a convergent series. Instead of regarding each element in the string as an element in $\mathcal{M}$ (which would not allow us to define the last concatenation since it would happen after ∞) we regard σ_1 and σ_2 as elements of $\mathcal{M}$ and $u^{t_1}u^{t_2}\ldots u^{t_k}\ldots = u^{t'}$ as an element of U^* and consequently as an element of $\mathcal{M}$, where $t' = \lim_{k\longrightarrow\infty} t_k$. This string is then regarded as the map $m : \{1,2,3\} \to \mathcal{M}$ defined by $m(1) = \sigma_1$, $m(2) = u^{t'}$ and $m(3) = \sigma_3$. The product in $\mathcal{M}$ is then the usual concatenation on reduced strings, that is, strings where all consequent sequences of elements of U^* or Σ^* have been replaced by their product in U^* or Σ^*, respectively. Hybrid control systems are now cast into the abstract control systems framework as:

Definition 2 (Hybrid Control System). *An hybrid control system $H = (X, \mathcal{M}_X, \Phi_X)$ consists of:*

- *The state space $X = \{X_q\}_{q\in Q}$.*
- *A generalized monoid $\mathcal{M}_X$ over X.*
- *A map $\Phi_X : \mathcal{M}_X \to X$ respecting the monoid structure and such that for all $q \in Q$, there is a set $Inv(q) \subseteq X_q$ and for all $x \in Inv(q)$, $\mathcal{M}_X(q,x) \cap U^* \neq \{\varepsilon\}$ and $\Phi((q,x), u^{t'}) \in Inv(q)$ for every prefix $u^{t'}$ of every $u^t \in \mathcal{M}_X(q,x)$.*

The semantics associated with the evolution from (q, x) governed by Φ and controlled by $a \in \mathcal{M}_{(q,x)}$ is the standard transition semantics of hybrid automata [3]. Suppose that $a = u^{t_1}\sigma_1\sigma_2 u^{t_2}$, then $\Phi((q,x), a) = (q', x')$ means that the system starting at (q, x) 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 (q', x'), t_2 units of time after the last jump.

2.1 Control System Abstractions

We now review the notions of simulation and bisimulation in the context of abstract control systems while referring the reader to Appendix A for the relevant notation.

Definition 3 (Simulations of Abstract Control Systems). *Let Φ_X and Φ_Y be two abstract control systems over X and Y with generalized monoids $\mathcal{M}_X$ and $\mathcal{M}_Y$, respectively and $F \subseteq \mathcal{M}_X \times \mathcal{M}_Y$ a generalized monoid respecting relation. Then Φ_Y is a simulation of Φ_X with respect to F or a F-simulation iff for any $x \in X$:*

$$y \in F_B(x) \;\Rightarrow\; \forall_{(x,a_x)\in dom(F)} \;\exists_{(y,a_y)\in F(x,a_x)} \qquad \Phi_Y(y,a_y) \in F_B(\Phi_X(x,a_x))$$

The above definition slightly generalizes the usual notions of morphisms between transition systems in [19], since the inputs in $\mathcal{M}_Y$, if obtained from F, depend on the inputs on $\mathcal{M}_X$ as well as the state. It is straightforward to see that abstract control systems and relations satisfying the above condition form a category, that we call the *abstract control systems category.* The notion of bisimulation is defined as a symmetric simulation:

Definition 4. *Let Φ_X and Φ_Y be abstract control systems over X and Y with generalized monoids $\mathcal{M}_X$ and $\mathcal{M}_Y$ respectively. If $F \subseteq \mathcal{M}_X \times \mathcal{M}_Y$ is a generalized monoid respecting relation we say that Φ_X is F-bisimilar to Φ_Y iff Φ_Y is a F-simulation of Φ_X and Φ_X is a F^{-1}-simulation of Φ_Y.*

Although we used relations to define simulations and bisimulations we will assume through the remaining paper that F is the relation induced by a map $f : \mathcal{M}_X \longrightarrow \mathcal{M}_Y$. The approach taken to define bisimulation is similar in spirit to the one in [10], however instead of preserving inputs between bisimulations, we relate them through the map f. If one chooses a map f which is the identity on inputs we recover the notion of bisimulation in [10]. Several other approaches to bisimulation are reported in the literature and we point the reader to the comparative study in [13] and the references therein.

The notion of simulation allows to define several different types of abstraction since when $f : \mathcal{M}_X \longrightarrow \mathcal{M}_Y$ defines a simulation from Φ_X to Φ_Y, the map f_B takes state trajectories of Φ_X to state trajectories of Φ_Y [15]. This shows, in particular, that $f(\mathcal{L}(\Phi_X)) \subseteq \mathcal{L}(\Phi_Y)$, where $\mathcal{L}(\Phi)$ denotes the language generated by abstract control system Φ. When f is simply the inclusion of $\mathcal{M}_X$ into $\mathcal{M}_Y$, that is $f(x,a) = (x,a) \in \mathcal{M}_Y$ for every $(x,a) \in \mathcal{M}_X$ we recover the popular notion of abstraction based on language inclusion since $\mathcal{L}(\Phi_X) = f(\mathcal{L}(\Phi_X)) \subseteq \mathcal{L}(\Phi_Y)$. Under certain conditions on the relation F the computation of a simulation can be done algorithmically as described in [17].

3 Compositional Abstractions

In this section, we follow the categorical description of composition of transition systems as described in [19]. A variety of composition operations can be modeled as the product operation followed by a restriction operation.

3.1 Parallel Composition with Synchronization

The first step of composition combines two abstract control systems into a single one by forming their product. Given two abstract control systems $\Phi_X : \mathcal{M}_X$

$\to X$ and $\Phi_Y : \mathcal{M}_Y \to Y$ we define their product to be the abstract control system $\Phi_X \times \Phi_Y : (\mathcal{M}_X \times \mathcal{M}_Y) \to (X \times Y)$, $\Phi_X \times \Phi_Y((x,y),(a_x,a_y)) = (\Phi_X(x,a_x), \Phi_Y(y,a_y))$, where the actions available at each $(x,y) \in X \times Y$ are subsets of the direct product monoid $\mathcal{M}_X \otimes \mathcal{M}_Y$. The trajectories of the product control system consist of all possible combinations of the initial control systems trajectories. The product can also be defined in a categorical manner.

Definition 5 (Product of abstract control systems). *Let $\Phi_X : \mathcal{M}_X \to X$ and $\Phi_Y : \mathcal{M}_Y \to Y$ 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 $\pi_X \subseteq (X \times Y) \times X$ and $\pi_Y \subseteq (X \times Y) \times Y$ are projection relations 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 for any other triple (Φ_Z, p_X, p_Y) of this type there is one and only one relation $\zeta \subseteq Z \times (X \times Y)$ such that $\Phi_X \times \Phi_Y$ is a ζ-simulation of Φ_Z, and the following diagram commutes:*

$$\begin{tikzcd}
\Phi_X & \Phi_X \times \Phi_Y \arrow[l, "\pi_X"'] \arrow[r, "\pi_Y"] & \Phi_Y \\
 & \Phi_Z \arrow[ul, "p_X"] \arrow[u, "\zeta"] \arrow[ur, "p_Y"'] &
\end{tikzcd} \tag{10}$$

The relations π_X and π_Y are in fact those induced by the canonical projection maps $\pi_X : X \times Y \to X$, $\pi_Y : X \times Y \to Y$ and the relation ζ 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 with respect to simulations a much simpler task.

Example 1. Consider the transition systems inspired from [19] and displayed on the left of Figure 1 where the ε evolutions are not represented. The product of these transitions systems will consist of all possible evolutions of both systems as displayed on the right of Figure 1.

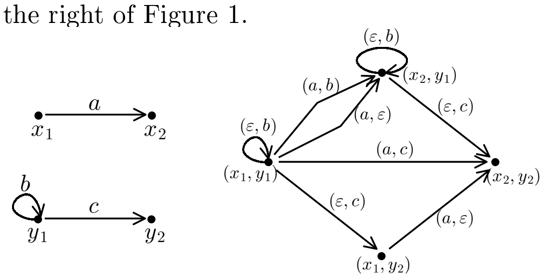

Fig. 1. Two transition systems on the left and the corresponding product transition system on the right.

In the product abstract control system, the behavior of one system does not influence the behavior of the other system. Since in general the behavior of a system composed of several subsystems depends strongly on the interaction between the subsystems, one tries to capture this interaction by removing undesired evolutions from the product system $\Phi_X \times \Phi_Y$ through the operation of restriction. Given a generalized submonoid $\mathcal{M}_L \subseteq \mathcal{M}_W$ we define the restriction of control system $\Phi_W : \mathcal{M}_W \longrightarrow W$ to $\mathcal{M}_L$ as a new control system $\Phi_W|_{\mathcal{M}_L} : \overline{\mathcal{M}_L} \longrightarrow L$ which is given by $\Phi_W|_{\mathcal{M}_L}(x, a) = \Phi_W(x, a)$ iff $(x, a) \in \mathcal{M}_L$ and $\Phi_W(x, a')$ belongs to L for any prefix a' of a. In general the domain of $\Phi_W|_{\mathcal{M}_L}$, $\overline{\mathcal{M}_L}$, may be strictly contained in $\mathcal{M}_L$ since restricting the base space implies also restricting the available inputs to those that do not force the abstract control system to leave the restricted base. If the generalized submonoid $\mathcal{M}_L$ has the same state space as $\mathcal{M}_W$ but "less" control inputs available at each state, then restriction is modeling synchronization of both systems on the control inputs. If on the other hand the available control inputs are equal but the state space of $\mathcal{M}_L$ is "smaller" then the state space of $\mathcal{M}_W$ then both systems are being synchronized on the state space. Synchronization on inputs and states is also captured by the operation of restriction by choosing a generalized submonoid with "less" available inputs and "smaller" state space. This operation also admits a categorical characterization.

Definition 6 (Restriction of abstract control systems). *Let $\Phi_W : \mathcal{M}_W \longrightarrow W$ be an abstract control system, $\mathcal{M}_L$ a generalized submonoid of $\mathcal{M}_W$ and g and h two simulation relations such that $\mathcal{M}_L = \{(w, a_w) \in \mathcal{M}_W \mid g(w, a_w) = h(w, a_w)\}$. The restriction of Φ_W to $\mathcal{M}_L$ is a pair $(\Phi_W|_{\mathcal{M}_L}, i_{\mathcal{M}_L})$ where $\Phi_W|_{\mathcal{M}_L}$ is an abstract control system and $i_{\mathcal{M}_L} \subseteq \mathcal{M}_L \times \mathcal{M}_W$ is an inclusion relation such that Φ_W is a $i_{\mathcal{M}_L}$-simulation of $\Phi_W|_{\mathcal{M}_L}$ satisfying $g \circ i_{\mathcal{M}_L} = h \circ i_{\mathcal{M}_L}$ and for any other pair $(\Phi_Z, i_{\mathcal{M}_Z})$ of this type there is one and only one relation η such that $\Phi_W|_{\mathcal{M}_L}$ is a η-simulation of Φ_Z, and the following diagram commutes:*

$$\begin{tikzcd}
\Phi_W|_{\mathcal{M}_L} \arrow[r, "i_{\mathcal{M}_L}"] & \Phi_W \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & \Phi_V \\
\Phi_Z \arrow[u, "\eta"] \arrow[ur, "i_{\mathcal{M}_Z}"'] & &
\end{tikzcd} \tag{11}$$

It is not difficult to see that the relation $i_{\mathcal{M}_L}$ is simply the inclusion $i_{\mathcal{M}_L}(a_l) = a_l \in \mathcal{M}_W$ for every $a_l \in \overline{\mathcal{M}_L}$. With the notions of product and restriction at hand, we can now define a general operation of parallel composition with synchronization.

Definition 7 (Parallel Composition with synchronization). *Let $\Phi_X : \mathcal{M}_X \longrightarrow X$ and $\Phi_Y : \mathcal{M}_Y \longrightarrow Y$ be two abstract control systems and consider a generalized submonoid $\mathcal{M}_L \subseteq \mathcal{M}_X \times \mathcal{M}_Y$. The parallel composition of Φ_X and Φ_Y with synchronization over $\mathcal{M}_L$ is the abstract control system defined as:*

$$\Phi_X \,\|_{\mathcal{M}_L}\, \Phi_Y = (\Phi_X \times \Phi_Y)|_{\mathcal{M}_L} \tag{12}$$

Example 2. Consider the transition systems displayed on the left of Figure 1. By specifying the generalized submonoid:

$$\mathcal{M}_L = \{((x_1, y_1), (a, b)), ((x_1, y_1), (\varepsilon, c))((x_1, y_1), (\varepsilon, \varepsilon)), ((x_2, y_1), (\varepsilon, c)), \\ ((x_2, y_1), (\varepsilon, \varepsilon)), ((x_2, y_2), (\varepsilon, \varepsilon)), ((x_1, y_2), (\varepsilon, \varepsilon))\} \quad (13)$$

it is possible to synchronize the event a with the event b on the parallel composition of these systems, while the remaining evolutions not controlled by a neither by b remain unchanged. The resulting transition system is displayed in Figure 2.

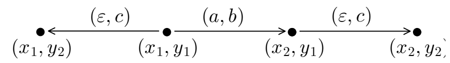

Fig. 2. Parallel composition with synchronization of the transition systems displayed on the left of Figure 1.

3.2 Compositionality of Simulations

We now determine if composition of subsystems is compatible with abstraction. A positive answer to this question is given by the next theorem which describes how the process of computing abstractions can be rendered more efficient by exploring the interconnection structure of hybrid systems.

Theorem 1 (Compositionality of Simulations). *Given abstract control systems Φ_X, Φ_Z (which is a F-simulation of Φ_X), Φ_Y, Φ_W (which is a G-simulation of Φ_Y) and the generalized submonoid $\mathcal{M}_L \subseteq \mathcal{M}_X \times \mathcal{M}_Y$, the parallel composition of the simulations Φ_Z and Φ_W with synchronization over $(F \times G)(\mathcal{M}_L)$ is a $(F \times G)|_{\overline{\mathcal{M}_L}}$-simulation of the parallel composition of Φ_X with Φ_Y with synchronization over $\mathcal{M}_L$.*

The above result was stated for parallel 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 parallel composition obtain an abstraction of the overall system.

Example 3. To illustrate the use of Theorem 1 we shall make use of the celebrated water tank system from [1]. Consider two water tanks that can be filled by water coming from a pipe as displayed on the left of Figure 3. The water level at tank A is measured by x_1 while the water level at tank B is measured by x_2. Each tank has also an outflow that causes a decrease in the water level. The outflow rate at tank A is v_1 while at tank B is v_2. This outflow can be compensated by a water inflow coming from the pipe on top of the tanks. This pipe has an inflow rate of $\overline{w}$ which can be directed to tank A or to tank B by means of a valve located in the pipe. Contrary to [1], we explicitly incorporate a first order model of the valve in the hybrid automaton describing this hybrid control system, displayed on the right of Figure 3. We now seek to abstract away the valve

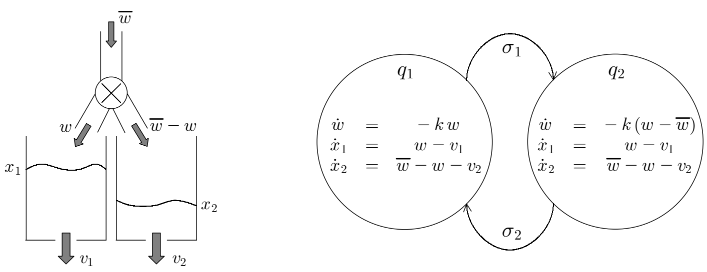

Fig. 3. Water tank system: Physical setup on the left and hybrid model on the right.

dynamics to obtain the usual model that considers the switching of the inflow from one tank to the other instantaneous[1]. Instead of computing an abstraction directly from this hybrid automaton we start by realizing that this automaton can be obtained by parallel composition of hybrid control systems H_X and H_Y modeling the pipe and the tanks, respectively, as shown in Figure 4. This compo-

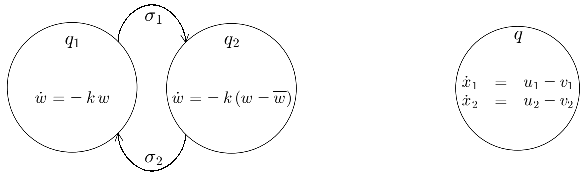

Fig. 4. Hybrid model of the pipe and water tanks on the left and right, respectively.

sition is synchronized on the generalized submonoid $\mathcal{M}_L \subseteq \mathcal{M}_X \times \mathcal{M}_Y$ defined by the equalities $u_1 = w$ and $u_2 = \overline{w} - w$. We now abstract the pipe model by aggregating all the continuous states in discrete state q_1 to 0 and all the continuous states in discrete state q_2 to $\overline{w}$. Theorem 1 ensures that composing H_Y with this abstraction will result in an abstraction of hybrid control system $H_X \|_{\mathcal{M}_L} H_Y$. The new synchronizing generalized monoid is obtained from $\mathcal{M}_L$ by replacing w by 0 on the continuous inputs in state q_1 and replacing w by $\overline{w}$ in the continuous inputs at discrete state q_2. This is also be described by the equalities $u_1 = 0$, $u_2 = \overline{w}$ and $u_1 = \overline{w}$, $u_2 = 0$ valid at discrete states q_1 and q_2, respectively. The resulting hybrid control system is displayed in Figure 5. This example illustrates the clear advantage of exploring compositionality in computing hybrid abstractions. We have only computed continuous abstractions of one-dimensional control systems (for the pipe automaton), whereas if one would have proceeded directly from hybrid control system $H_X \|_{\mathcal{M}_L} H_Y$ without exploring the compositional structure, one would have computed continuous abstractions of the three-dimensional continuous control systems at each discrete location.

[1] We remark that considering the water switching instantaneous leads to zeno trajectories [6], however this problem falls beyond the scope of the current paper.

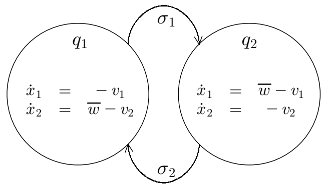

Fig. 5. Abstracted hybrid model of the water tank system.

3.3 Compositionality of Bisimulations

In this section we extend the previous compatibility results from simulations to bisimulations. Although the product respects bisimulations the same does not happen with the operation of restriction so we need additional assumptions to ensure that bisimulations are respected by composition as stated in the next result.

Theorem 2 (Compositionality of Bisimulations). *Given abstract control systems Φ_X, Φ_Z (a F-bisimulation of Φ_X), Φ_Y, Φ_W (a G-bisimulation of Φ_Y) and a generalized submonoid $\mathcal{M}_L \subseteq \mathcal{M}_X \times \mathcal{M}_Y$ we have that the parallel composition of the bisimulations Φ_Z and Φ_W with synchronization over $(F \times G)(\mathcal{M}_L)$ is a $(F \times G)|_{\overline{\mathcal{M}_L}}$-bisimulation of the parallel composition of Φ_X with Φ_Y with synchronization over $\mathcal{M}_L$ iff $(F \times G)^{-1}|_{\overline{(F\times G)(\mathcal{M}_L)}} = (F \times G)|_{\overline{\mathcal{M}_L}}{}^{-1}$ where $\overline{(F \times G)(\mathcal{M}_L)}$ is the domain of $\Phi_Z \parallel_{(F\times G)(\mathcal{M}_L)} \Phi_W$.*

From the previous result we conclude that if we have a means of computing bisimulations and if we choose the synchronization generalized submonoid carefully we can compute bisimulations by exploring the interconnection structure of large-scale systems.

4 Conclusions

In this paper, we addressed the interplay between abstractions and compositionality of hybrid systems. Based on previous work on abstractions of hybrid control systems, we introduced a composition operator, and showed that this composition operator is compatible with abstractions based on simulations. Furthermore, we presented necessary and sufficient conditions for this operator to be also compatible with bisimulations. Current research is focusing on classes of hybrid systems and composition operators for which the abstraction process can be fully automated. Another important topic for future research is to understand which conditions guarantee that hybrid systems relevant properties are preserved by abstractions, and specially by composition operators.

Acknowledgments: The authors would like to thank Esfandiar Haghverdi for extremely stimulating discussions on category theory, and its use for hybrid systems. The first author was supported by Fundação para a Ciência e Tecnologia under grant PRAXIS XXI/BD/18149/98 while the second author was partially supported by DARPA ITO MoBIES Grant F33615-00-C-1707.

References

1. R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of the 9th International Conference on Concurency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 74–88. Springer-Verlag, 1997.
2. Rajeev Alur, Radu Grosu, Insup Lee, and Oleg Sokolsky. Compositional refinements for hierarchical hybrid systems. In *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 33–48. Springer Verlag, 2001.
3. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
4. Thomas A. Henzinger, Marius Minea, and Vinayak Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 275–290. Springer Verlag, 2001.
5. John E. Hopcroft and Jeffery D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company, USA, 1979.
6. Karl Henrick Johansson, Magnus Egersted, John Lygeros, and S. Sastry. On the regularization of hybrid automata. *Systems and Control Letters*, 38:141–150, 1999.
7. E.A. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on Computer Aided Design*, 17(12), December 1998.
8. Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid I/O automata revisited. In *Hybrid Systems : Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 403–417. Springer Verlag, 2001.
9. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer Verlag, New York, 1995.
10. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
11. George J. Pappas, Gerardo Lafferriere, and Shankar Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, June 2000.
12. 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.
13. Markus Roggenbach and Mila Majster-Cederbaum. Towards a unified view of bisimulation: a comparative study. *Theoretical Computer Science*, (238):81–130, 2000.
14. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.
15. Paulo Tabuada. *Hierarchies and Compositional Abstractions of Hybrid Systems.* PhD thesis, Instituto Superior Técnico, Lisbon, Portugal, January 2002.
16. Paulo 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*, pages 501–514. Springer Verlag, 2001.
17. Paulo Tabuada, George J. Pappas, and Pedro Lima. Compositional abstractions of hybrid control systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, Florida, 2001.
18. A. J. van der Schaft and J. M. Schumacher. Compositionality issues in discrete, continuous, and hybrid systems. *International Journal of Robust and Nonlinear Control*, 11(5):417–434, April 2001.

19. Glynn Winskel and Mogens Nielsen. Models for concurrency. In Abramsky, Gabbay, and Maibaum, editors, *Handbook of Logic and Foundations of Theoretical Computer Science*, volume 4. Oxford University Press, London, 1994.

A Notation and Mathematical Facts

A relation is a generalization of a function in the sense that it assigns to each element in its domain a *set* of elements in its codomain. Mathematically a relation F between the sets S_1 and S_2 is simply a subset of their Cartesian product, that is $F \subseteq S_1 \times S_2$. Given two relations $F \subseteq S_1 \times S_2$ and $G \subseteq S_2 \times S_3$ we can define their composition to be the relation $G \circ F \subseteq S_1 \times S_3$ defined by $G \circ F = \{(s_1, s_3) \in S_1 \times S_3 \;:\; \exists s_2 \in S_2 \quad (s_1, s_2) \in F \wedge (s_2, s_3) \in G\}$. Given a relation $F \subseteq S_1 \times S_2$ we call $F^{-1} \subseteq S_2 \times S_1$ given by $F^{-1} = \{(s_2, s_1) \in S_2 \times S_1 \;:\; (s_1, s_2) \in F\}$ the inverse relation. An object that we will use frequently is the set valued map $F : S_1 \longrightarrow 2^{S_2}$ induced by a relation F and defined by $F(s_1) = \{s_2 \in S_2 \;:\; (s_1, s_2) \in F\}$.

We also introduce some notation for later use. Given relations $F \subseteq S_1 \times S_2$, $G \subseteq S_3 \times S_4$ and a subset $L \subseteq S_1 \times S_3$ we define the new relations $F \times G$ and $(F \times G)|_L$ as $F \times G = \{((s_1, s_3), (s_2, s_4)) \in (S_1 \times S_3) \times (S_2 \times S_4) \;:\; (s_1, s_2) \in F \wedge (s_3, s_4) \in G\}$ and $(F \times G)|_L = \{((s_1, s_3), (s_2, s_4)) \in F \times G \;:\; (s_1, s_3) \in L\}$.

As explained in Section 2 we will need to work with generalized monoids. We start by recalling the notion of monoid. A monoid is a triple $(\mathcal{M}, \cdot, \varepsilon)$ where $\mathcal{M}$ is a set closed under the associative operation $\cdot : \mathcal{M} \times \mathcal{M} \longrightarrow \mathcal{M}$ and ε is a special element of $\mathcal{M}$ called identity. This element satisfies $\varepsilon \cdot m = m \cdot \varepsilon = m$ for any $m \in \mathcal{M}$. We will usually denote $m_1 \cdot m_2$ simply by $m_1 m_2$ and refer to the monoid simply as $\mathcal{M}$. Given two elements m_1 and m_2 from $\mathcal{M}$ we say that m_1 is a prefix of m_2 iff there exists another $m \in \mathcal{M}$ such that $m_1 m = m_2$. We will be specially interested in generalized monoids obtained as follows. Let X be a set and $\mathcal{M}$ a monoid. Then we can regard $X \times \mathcal{M}$ as a set valued function $F : X \longrightarrow 2^{\mathcal{M}}$ which assigns to each $x \in X$ the monoid $F(x) = \mathcal{M}$. However, in general, not all the elements of $\mathcal{M}$ will be available at each point in X so that we need[2] a map $G : X \longrightarrow 2^{\mathcal{M}}$ such that $G(x)$ may be a strict subset of $\mathcal{M}$ with the property that $G(x)$ is prefix closed for every $x \in X$. Such a map will be called a generalized monoid over the set X and we shall denote it by $\mathcal{M}_X$. We will, interchangeably, regard a generalized monoid as a map from X to $2^{\mathcal{M}}$ or as the subset of $X \times \mathcal{M}$ defined by $(x, m) \in \mathcal{M}_X$ iff $m \in \mathcal{M}_X(x)$. A subset $\mathcal{M}_L$ of $\mathcal{M}_X$ which is also a generalized monoid will be called a generalized submonoid.

We now relate generalized monoids through relations. Let $F \subseteq \mathcal{M}_X \times \mathcal{M}_Y$ be a relation between generalized monoids. Then F induces a relation $F_B \subseteq X \times Y$ by $y \in F_B(x)$ iff $(y, m) \in F(x, m')$ for any $(y, m) \in \mathcal{M}_Y$ and $(x, m') \in \mathcal{M}_X$. We then say that the relation F is generalized monoid respecting iff satisfies:

- **Identity:** $y \in F_B(x) \;\Rightarrow\; (y, \varepsilon) \in F(x, \varepsilon)$

[2] In general, a generalized monoid over a set X can be seen as a small category with elements of X as objects.

– **Semi-group:** $(y_1, m_1') \in F(x_1, m_1), (y_2, m_2') \in F(x_2, m_2)$ and $(x_1, m_1 m_1') \in \mathcal{M}_X \Rightarrow (y_1, m_1' m_2') \in F(x_1, m_1 m_2)$.

B Proofs

Proof (of Theorem 1). Consider the product system $(\Phi_Z \times \Phi_W, \pi_Z, \pi_W)$ and the triple $(\Phi_X \times \Phi_Y, F \circ \pi_X, G \circ \pi_Y)$. By definition of product we know that there is one and only one relation ζ such that:

$$\begin{tikzcd}
\Phi_Z & \Phi_Z \times \Phi_W \arrow[l, "\pi_Z"'] \arrow[r, "\pi_W"] & \Phi_W \\
& \Phi_X \times \Phi_Y \arrow[ul, "F \circ \pi_X"] \arrow[u, "\zeta"] \arrow[ur, "G \circ \pi_Y"'] &
\end{tikzcd}$$

commutes and this relation is given by $\zeta = (F, G) = F \times G$, meaning that $\Phi_Z \times \Phi_W$ is a $F \times G$-simulation of $\Phi_X \times \Phi_Y$. Consider now the following diagram:

$$\begin{tikzcd}
(\Phi_X \times \Phi_Y)|_{\mathcal{M}_L} \arrow[r, "\zeta \circ i_{\mathcal{M}_L}"] & \Phi_Z \times \Phi_W \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & \Phi_V
\end{tikzcd}$$

where g and h are equal only on the generalized submonoid $\zeta(\mathcal{M}_L)$. It is clear that $g \circ \zeta \circ i_{\mathcal{M}_L} = h \circ \zeta \circ i_{\mathcal{M}_L}$ since $\overline{\mathcal{M}_L} \subseteq \mathcal{M}_L$ implies $\zeta \circ i_{\mathcal{M}_L}(\overline{\mathcal{M}_L}) = \zeta(\overline{\mathcal{M}_L}) \subseteq \zeta(\mathcal{M}_L)$. Therefore, by definition of restriction there exists one and only one simulation relation η from $\Phi_X \parallel_{\mathcal{M}_L} \Phi_Y$ to $\Phi_Z \parallel_{\zeta(\mathcal{M}_L)} \Phi_W$ which is given by $\eta = \zeta \circ i_{\mathcal{M}_L} = (F \times G) \circ i_{\mathcal{M}_L} = (F \times G)|_{\overline{\mathcal{M}_L}}$. □

Proof (of Theorem 2). We now prove Theorem 2 through a series of results. We start by showing that product respects bisimulations:

Lemma 1. *Given abstract control systems Φ_X, Φ_Z (a F-bisimulation of Φ_X), Φ_Y and Φ_W (a G-bisimulation of Φ_Y) the product abstract control system $\Phi_Z \times \Phi_W$ is a $F \times G$-bisimulation of $\Phi_X \times \Phi_Y$.*

Proof. Consider the following commutative diagrams:

$$\begin{tikzcd}
& \Phi_X \times \Phi_Y \arrow[dl, "\pi_X"'] \arrow[dr, "\pi_Y"] \arrow[dddd, "\eta_1"] & \\
\Phi_X \arrow[dd, "F"'] & & \Phi_Y \arrow[dd, "G"'] \\
& & \\
\Phi_Z & & \Phi_W \\
& \Phi_Z \times \Phi_W \arrow[ul, "\pi_Z"] \arrow[ur, "\pi_W"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \Phi_X \times \Phi_Y \arrow[dl, "\pi_X"'] \arrow[dr, "\pi_Y"] & \\
\Phi_X & & \Phi_Y \\
& & \\
\Phi_Z \arrow[uu, "F^{-1}"'] & & \Phi_W \arrow[uu, "G^{-1}"'] \\
& \Phi_Z \times \Phi_W \arrow[ul, "\pi_Z"] \arrow[ur, "\pi_W"'] \arrow[uuuu, "\eta_2"] &
\end{tikzcd}$$

By definition of product there exists one and only one relation η_1 and one and only one relation η_2 such that the diagrams commute. In fact, η_1 is the relation $\eta_1 = (F \circ \pi_X, G \circ \pi_Y) = F \times G$ and $\eta_2 = (F^{-1} \circ \pi_Z, G^{-1} \circ \pi_W) = (F \times G)^{-1}$ meaning that $\Phi_X \times \Phi_Y$ is $F \times G$-bisimilar to $\Phi_Z \times \Phi_W$. □

Under the proper assumptions the operation of restriction is also compatible with bisimulations:

Proposition 1. *Let Φ_X be an abstract control system, Φ_Y a F-bisimulation of Φ_X and $\mathcal{M}_L$ a generalized submonoid of $\mathcal{M}_X$ such that $F^{-1}|_{\overline{F(\mathcal{M}_L)}} = (F|_{\overline{\mathcal{M}_L}})^{-1}$. The restriction $\Phi_X|_{\mathcal{M}_L}$ is a $F|_{\mathcal{M}_L}$-bisimulation of $\Phi_Y|_{F(\mathcal{M}_L)}$.*

Proof. A similar argument to the proof of Proposition 1 shows that Φ_Y is a $F|_{\overline{\mathcal{M}_L}}$-simulation of Φ_X so that we will only show that Φ_X is a $F|_{\overline{\mathcal{M}_L}}^{-1}$-simulation of Φ_Y. Consider the following diagram:

$$\begin{tikzcd}
\Phi_Y|_{F(\mathcal{M}_L)} \arrow[dr, "F^{-1}\circ i_{F(\mathcal{M}_L)}"] & & \\
\Phi_X|_{\mathcal{M}_L} \arrow[r, "i_{\mathcal{M}_L}"'] & \Phi_X \arrow[r, shift left, "g"] \arrow[r, shift right, "h"'] & \Phi_V
\end{tikzcd} \tag{17}$$

where g and h are equal only on the generalized submonoid $\mathcal{M}_L$. We will show that (17) commutes by proving the only nontrivial equality, $g \circ F^{-1} \circ i_{F(\mathcal{M}_L)} = h \circ F^{-1} \circ i_{F(\mathcal{M}_L)}$. Recall that the equality $F^{-1}|_{\overline{F(\mathcal{M}_L)}} = F|_{\overline{\mathcal{M}_L}}^{-1}$ implies that the domains of the relations are the same, that is $\overline{F(\mathcal{M}_L)} = F(\overline{\mathcal{M}_L})$. This allows to conclude that:

$$\begin{aligned}
F^{-1} \circ i_{F(\mathcal{M}_L)}(\overline{F(\mathcal{M}_L)}) &= F^{-1}|_{\overline{F(\mathcal{M}_L)}} \circ F(\overline{\mathcal{M}_L}) \\
&= F|_{\overline{\mathcal{M}_L}}^{-1} \circ F|_{\overline{\mathcal{M}_L}}(\overline{\mathcal{M}_L}) = \overline{\mathcal{M}_L} \subseteq \mathcal{M}_L
\end{aligned}$$

Since (17) commutes we can invoke the definition of restriction to ensure the existence of a unique simulation relation from $\Phi_Y|_{F(\mathcal{M}_L)}$ to $\Phi_X|_{\mathcal{M}_L}$ which is given by $\eta = F^{-1} \circ i_{F(\mathcal{M}_L)} = F^{-1}|_{\overline{F(\mathcal{M}_L)}} = F|_{\overline{\mathcal{M}_L}}^{-1}$ thereby showing bisimilarity. □

The condition of the previous result is in fact also a necessary one as we now show:

Proposition 2. *Let Φ_X be an abstract control system, Φ_Y a F-bisimulation of Φ_X and $\mathcal{M}_L$ a generalized submonoid of $\mathcal{M}_X$. If the restriction $\Phi_X|_{\mathcal{M}_L}$ is a $F|_{\mathcal{M}_L}$-bisimulation of $\Phi_Y|_{F(\mathcal{M}_L)}$ then $F^{-1}|_{\overline{F(\mathcal{M}_L)}} = (F|_{\overline{\mathcal{M}_L}})^{-1}$.*

Proof. The following commutative diagram is a consequence of bisimilarity:

$$\begin{tikzcd}
\Phi_Y|_{F(\mathcal{M}_L)} \arrow[r, "i_{F(\mathcal{M}_L)}"] \arrow[d, "F|_{\mathcal{M}_L}^{-1}"] & \Phi_Y \arrow[d, "F^{-1}"] \\
\Phi_X|_{\mathcal{M}_L} \arrow[r, "i_{\mathcal{M}_L}"] & \Phi_X
\end{tikzcd} \tag{18}$$

from which we get the following equality:

$$i_{\mathcal{M}_L} \circ F|_{\mathcal{M}_L}^{-1} = F^{-1} \circ i_{F(\mathcal{M}_L)} \tag{19}$$

from which follows the desired equality $F|_{\mathcal{M}_L}^{-1} = F^{-1}|_{\overline{F(\mathcal{M}_L)}}$. □

Theorem 2 is just a restatement of Lemma 1 and Propositions 1 and 2 and is therefore proved. □