March 2002

Composing Abstractions of Hybrid Systems

Paulo Tabuada
Instituto Superior Técnico

George J. Pappas
University of Pennsylvania, pappasg@seas.upenn.edu

Pedro Lima
Instituto Superior Técnico

Postprint version. Published in *Lecture Notes in Computer Science*, Volume 2289, Hybrid Systems: Computation and Control: Proceedings of the 5th International Workshop, HSCC 2002, pages 436-450.
Publisher URL: http://springerlink.metapress.com/link.asp?id=105633

Composing Abstractions of Hybrid Systems

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.

Comments
Postprint version. Published in *Lecture Notes in Computer Science*, Volume 2289, Hybrid Systems: Computation and Control: Proceedings of the 5th International Workshop, HSCC 2002, pages 436-450.
Publisher URL: http://springerlink.metapress.com/link.asp?id=105633

Composing Abstractions of Hybrid Systems

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

[1] Instituto de Sistemas e Robótica, Instituto Superior Técnico
1049-001 Lisboa - Portugal, {tabuada,pal}@isr.ist.utl.pt
[2] Department of Electrical Engineering, University of Pennsylvania
Philadelphia, PA 19104, pappasg@ee.upenn.edu

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 system 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 details 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 [9], [17], and computer aided verification [8]. 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 [7], compositional refinements in a hierarchical setting are discussed in [2], and assume guarantee proof rules presented in [4].

For purely continuous systems, the notions of simulation, and bisimulation had not received much attention [16]. Recently, similar notions were introduced in [10, 11] which has resulted in constructions of abstractions for linear control systems [10], and nonlinear control systems [11] while characterizing abstracting maps that preserve properties of interest such as controllability. Based on these results, in [14], we took the first steps towards constructing abstractions of hybrid systems while preserving timed languages. This allowed us to introduce in [15] of 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 [15] towards compositionality. Following the approach described in [17], 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 [15] and introduce the notions of simulation and bisimulation. In Section 3 we introduce a composition operator based on [17], 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 [15], 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 [6, 13] by regarding systems as *control* systems. The framework is based on a notion of evolution, and the ability to control the evolution. The reader is refered to Appendix A for some mathematical preliminaries.

Definition 1 (Abstract Control System). *Let S be a set, $\mathcal{M}$ a monoid and A a fibering relation on $S \times \mathcal{M}$ with base space S such that A_s is a prefix closed subset of $\mathcal{M}$ containing the identity for every $s \in S$. An abstract control system over S is a map $\Phi : A \longrightarrow S$ respecting the monoid structure, that is $\Phi_s : A_s \longrightarrow S$ verifies:*

1. **Identity:** $\Phi_s(\varepsilon) = s$
2. **Semi-group:** $\Phi_{\Phi_s(a_s)}(a_{s'}) = \Phi_s(a_s a_{s'})$

Intuitively, we can think of the set S as the state space, and the fiber bundle A, also called in this work a fibering monoid, as the set of possible actions, that depend on the base point. The map Φ assigns to each point $s \in S$ a function

from A_s to S representing all the input choices that can be made at the point s. Given an input choice $a_s \in A_s$, $\Phi_S(a_s)$ returns the state reached under the action of the control input a_s. To get a better understanding of the above definition we will see how it applies to three classes of systems.

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^*. With concatenation as a monoid operation, Σ^* can be taken as the monoid $\mathcal{M}$. The state space is naturally $S = Q$. The transition function δ defines a *unique* partial map from $Q \times \Sigma^*$ to Q which is just an abstract control system $\Phi : (S \times \mathcal{M})|_R = A \longrightarrow S$, where R is the fibering monoid respecting relation given by $R = \{(s, m) \in S \times \mathcal{M} \ : \ \Phi(s, m) \text{ is defined}\}$.

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

$$U^t = \{u : [0, t[\rightarrow U \quad | \quad [0, t[\subseteq \mathbb{R}_0^+ \} \tag{1}$$

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 $0 \leq t < \infty$:

$$U^* = \bigcup_{0 \leq t < \infty} U^t \tag{2}$$

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:

$$u^{t_1} u^{t_2}(t) = \begin{cases} u^{t_1}(t) & \text{if} \quad 0 \leq t < t_1 \\ u^{t_2}(t - t_1) & \text{if} \quad t_1 \leq t < t_1 + t_2 \end{cases} \tag{3}$$

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 $\gamma_x : [0, t[\rightarrow M$, such that $\gamma_x(0) = x$. We can then cast any smooth control system into our framework by defining:

$$\begin{aligned} \Phi : M \times U^* &\longrightarrow M \\ (x, u^t) &\mapsto \gamma_x(t) \end{aligned} \tag{4}$$

It is not difficult to see that Φ is in fact a well defined abstract control system since $\Phi(x, \varepsilon) = \gamma_x(0) = x$ and $\Phi(x, u^{t_1} u^{t_2}) = \gamma_x(t_1 + t_2) = \gamma_{\gamma_x(t_1)}(t_2) = \Phi(\Phi(x, u^{t_1}), u^{t_2})$. In general, the set of admissible control inputs may change with the point x so that the domain of Φ will be in fact a fiber bundle over M.

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). As action monoid we will use the set:

$$\mathcal{M} = \bigcup_{t \in \{1,2,\ldots,n\}} (U^* \cup \Sigma^*)^{\{1,2,\ldots,t\}} \tag{5}$$

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 $\sigma_1 u^{t_1} u^{t_2} \ldots u^{t_n} \ldots \sigma_2$ belongs to $\mathcal{M}$, where t_n 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 $u^{t_1} u^{t_2} \ldots u^{t_n} \ldots = u^{t'}$ as an element of U^* and consequently as an element of $\mathcal{M}$, where $t' = \lim_{n \to \infty} t_n$. This string is then regarded as the map $u : \{1, 2, 3\} \to \mathcal{M}$ defined by $u(1) = \sigma_1$, $u(2) = u^{t'}$ and $u(3) = \sigma_3$. 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* $H = (X, A_X, \Phi_X)$ *consists of:*

- *The state space* $X = \{X_q\}_{q \in Q}$.
- *A fibering relation* A_X *on* $X \times \mathcal{M}$ *defined by:*

$$A_X = \{((q, x), m) \in X \times \mathcal{M} \ : \ \Phi_X((q, x), m) \text{ is defined}\}$$

- *A partial map* $\Phi_X : X \times \mathcal{M} \to X$ *respecting the monoid structure 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 every prefix* $u^{t'}$ *of every* $u^t \in A_{(q,x)}$.

The semantics associated with the evolution from (q, x) governed by Φ and controlled by $a \in A_{(q,x)}$ 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.

Definition 3 (Simulations of Abstract Control Systems). *Let Φ_X and Φ_Y be two abstract control systems over X and Y with fibering monoids A_X and A_Y, respectively. Let $R \subseteq A_X \times A_Y$ be a fibering monoid respecting relation. Then Φ_Y is a simulation of Φ_X with respect to R or a R-simulation iff:*

$$\forall_{x \in X}(x, y) \in R_B \Rightarrow \forall_{(x,a_x) \in dom(R)} \exists_{(x,a_x,y,a_y) \in R} \qquad (\Phi_X(x, a_x), \Phi_Y(y, a_y)) \in R_B$$

The above definition slightly generalizes the usual notions of morphisms between transition systems in [17], since we allow the control inputs to depend on the state space and since we use relations instead of functions. 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 fibering monoids A_X and A_Y respectively. If $R \subseteq A_X \times A_Y$ is a fiber respecting relation we say that Φ_X is a R-bisimilar to Φ_Y iff Φ_Y is a R-simulation of Φ_X and Φ_X is a R^{-1}-simulation of Φ_Y.*

The approach taken to define bisimulation is similar in spirit to the one in [9], however instead of preserving labels between bisimulations, we relate them through the relation. Several other approaches to bisimulation are reported in the literature and we point the reader to the comparative study in [12] and the references therein.

If the simulation relation is surjective and defined for every $(x, a) \in A_X$ then in [15] we have presented an algorithm to perform the abstraction process for hybrid systems, based on the continuous constructions of [10] and [11].

3 Compositional Abstractions

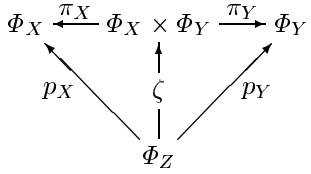
In this section, we follow the categorical description of composition of transition systems as described in [17]. A variety of composition operations can be modeled as the product operation followed by a restriction operation.

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 $\Phi_X : A_X \longrightarrow X$ and $\Phi_Y : A_Y \longrightarrow Y$ we define their product to be the abstract control system $\Phi_X \times \Phi_Y : (A_X \times A_Y) \longrightarrow (X \times Y)$, $\Phi_X \times \Phi_Y(a_x, a_y) = (\Phi_X(a_x), \Phi_Y(a_y))$, where the fibers of $(A_X \times A_Y)$ 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 $\Phi_X : A_X \longrightarrow X$ and $\Phi_Y : A_Y \longrightarrow Y$ 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:*



(7)

The relations π_X and π_Y are in fact those induced by the canonical projection maps $\pi_X : X \times Y \longrightarrow X$, $\pi_Y : X \times Y \longrightarrow Y$ 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 an almost trivial task.

Example 1. Consider the transition systems inspired from [17] 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.

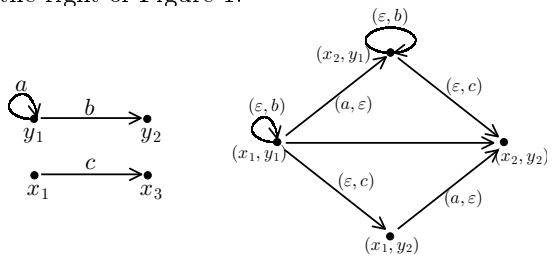


Fig. 1. Two transition systems on the left and the corresponding product transition system on the right.

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 fibering submonoid[1] $A_L \subseteq A_W$ we define the restriction of control system $\Phi_W : A_W \longrightarrow W$ to A_L as a new control system $\Phi_W|_{A_L} : A_L \longrightarrow L$ which is given by $\Phi_W|_{A_L}(x, a) = \Phi_W(x, a)$ iff $(x, a) \in A_L$ and $\Phi_W(x, a')$ belongs to L for any prefix a' of a. If the fibering submonoid A_L has the same base space

[1] A fibering submonoid A of a fibering monoid B is understood as a fibering monoid such that the inclusion relation $i : A \hookrightarrow B$ is fibering monoid preserving.

as A_W but "smaller" fibers, then restriction is modeling synchronization of both systems on the control inputs. If on the other hand the fibers are equal but the base space of A_L is "smaller" then the base space of A_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 fibering submonoid with "smaller" fibers and base space. This operation also admits a categorical characterization.

Definition 6 (Restriction of abstract control systems). *Let $\Phi_W : A_W \rightarrow W$ be an abstract control system and let A_L be a fibering submonoid of A_W. The restriction of Φ_W to A_L is a pair $(\Phi_W|_{A_L}, i_L)$ where $\Phi_W|_{A_L}$ is an abstract control system and $i_L \subseteq L \times W$ is an inclusion relation such that Φ_W is a i_L-simulation of $\Phi_W|_{A_L}$, and for any other pair (Φ_Z, i_Z) of this type with $i_Z(A_Z) = i_L(A_L)$ there is one and only one relation η such that $\Phi_W|_L$ is a η-simulation of Φ_Z, and the following diagram commutes:*

$$\begin{array}{ccc} \Phi_W|_{A_L} & \xrightarrow{i_L} & \Phi_W \\ \eta \uparrow & \nearrow_{i_Z} & \\ \Phi_Z & & \end{array} \tag{8}$$

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

Definition 7 (Parallel Composition with synchronization). *Let $\Phi_X : A_X \rightarrow X$ and $\Phi_Y : A_Y \rightarrow Y$ be two abstract control systems and consider a fibering submonoid $A_L \subseteq A_X \times A_Y$. The parallel composition of Φ_X and Φ_Y with synchronization over A_L is the abstract control system defined as:*

$$\Phi_X \parallel_{A_L} \Phi_Y = (\Phi_X \times \Phi_Y)|_{A_L} \tag{9}$$

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

$$\begin{aligned} A_L = \{&((x_1, y_1), (a, b)), ((x_1, y_1), (\varepsilon, \varepsilon)), ((x_2, y_1), (\varepsilon, c)), \\ &((x_2, y_1), (\varepsilon, \varepsilon)), ((x_2, y_2), (\varepsilon, \varepsilon)), ((x_1, y_2), (\varepsilon, \varepsilon))\} \end{aligned} \tag{10}$$

it is possible to synchronize the event a with the event b on the parallel composition of these systems. The resulting transition system is displayed in Figure 2.

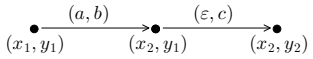


Fig. 2. Parallel composition with synchronization of the transition systems displayed on the left of Figure 1.

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 R_X-simulation of Φ_X), Φ_Y, Φ_W (which is a R_Y-simulation of Φ_Y) and the fibering submonoid $A_L \subseteq A_X \times A_Y$, the parallel composition of the simulations Φ_Z and Φ_W with synchronization over $R_{X\times Y}(A_L)$ is a $R_{X\times Y}|_{A_L}$-simulation of the parallel composition of Φ_X with Φ_Y with synchronization over A_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

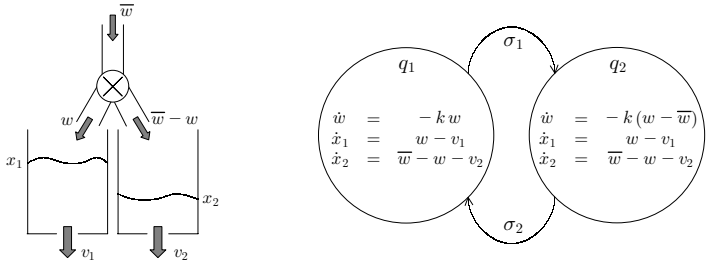


Fig. 3. Water tank system: Physical setup on the left and hybrid model on the right.

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 pump in the hybrid automaton describing this hybrid control system, displayed on the right of Figure 3. We now seek to abstract away the pump dynamics to obtain the usual model that considers the commutation of the inflow from one tank to the other instantaneous[2]. Instead of computing an

[2] We remark that considering the water commutation instantaneous leads to zeno trajectories [5], however this problem falls beyond the scope of the current paper.

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

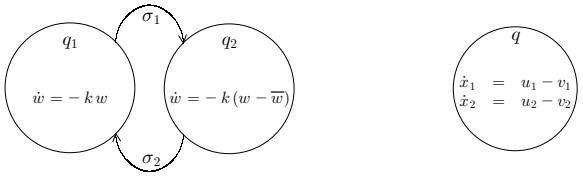


Fig. 4. Hybrid model of the pipe and water tanks on the left and right, respectively.

composition is synchronized on the fibering submonoid $A_L \subseteq A_X \times A_Y$ defined by the points of the form $\big(((q_1, w), (x_1, x_2)), (\varepsilon, u^t)\big)$, $\big(((q_1, w), (x_1, x_2)), (\sigma_1, \varepsilon)\big)$, $\big(((q_2, w), (x_1, x_2)), (\varepsilon, u^t)\big)$ and $\big(((q_2, w), (x_1, x_2)), (\sigma_2, \varepsilon)\big)$, where the continuous inputs satisfy $u^t = (w(t), \overline{w} - w(t))$. 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 $H_X \parallel_{A_L} H_Y$. The new synchronizing fibering monoid is obtained from A_L by replacing w by 0 on the continuous inputs in state q_1, replacing w by $\overline{w}$ in the continuous inputs at discrete state q_2 and identifying (q_1, w) and (q_2, w) with q_1 and q_2, respectively. The resulting hybrid control system is displayed in Figure 5. This example il-

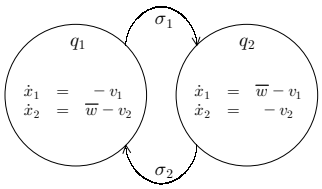


Fig. 5. Abstracted hybrid model of the water tank system.

lustrates 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 $H_X \parallel_{A_L} H_Y$ without exploring the compositional structure, one would have computed continuous abstractions of the three-dimensional continuous control systems at each discrete location.

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 R_X-bisimulation of Φ_X), Φ_Y, Φ_W (a R_Y-bisimulation of Φ_Y) and a fibering submonoid $A_L \subseteq A_X \times A_Y$ we have that the parallel composition of the bisimulations Φ_Z and Φ_W with synchronization over $R_{X\times Y}(A_L)$ is a $R_{X\times Y}|_{A_L}$-bisimulation of the parallel composition of Φ_X with Φ_Y with synchronization over A_L iff $R^{-1}_{X\times Y} \circ R_{X\times Y}|_{A_L} = id_{A_L}$ and $R_{X\times Y} \circ (R_{X\times Y}|_{A_L})^{-1} = id_{R_{X\times Y}(A_L)}$.*

From the previous result we conclude that if we have a mean of computing bisimulations and if we choose the synchronization fibering submonoid carefully we can compute bisimulations by exploring the interconnecting structure of large-scale systems. A constructive algorithm to compute abstractions of hybrid control systems was proposed in [15]. It was also shown that if certain assumptions hold, then the algorithm also computes bisimulations.

4 Conclusions

In this paper we addressed the compositional abstractions 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. Furthermore, we presented necessary and sufficient conditions for these operator to be also compatible with bisimulations. Clearly, future research should focus on classes of hybrid systems and composition operators where the abstraction process can be fully automated.

Acknowledgments: The authors would like to thank Esfandiar Haghverdi for extremely stimulating discussions on category theory, and its use for hybrid systems. The work of George J. Pappas is 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. Karl Henrick Johansson, Magnus Egersted, John Lygeros, and S. Sastry. On the regularization of hybrid automata. *Systems and Control Letters*, 38:141–150, 1999.
6. E.A. Lee and A. Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on Computer Aided Design*, 17(12), December 1998.
7. 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.
8. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer Verlag, New York, 1995.
9. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
10. George J. Pappas, Gerardo Lafferriere, and Shankar Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, June 2000.
11. George J. Pappas and Slobodan Simic. Consistent abstractions of affine control systems. *IEEE Transactions on Automatic Control*, 2001. To appear.
12. Markus Roggenbach and Mila Majster-Cederbaum. Towards a unified view of bisimulation: a comparative study. *Theoretical Computer Science*, (238):81–130, 2000.
13. J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.
14. 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*. Springer Verlag, 2001.
15. 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.
16. 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.
17. 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 R between the sets S_1 and S_2 is simply a subset of their Cartesian product, that is $R \subseteq S_1 \times S_2$. The domain of a relation is the set $dom(R) = \{s_1 \in S_1 \ : \ \exists s_2 \in S_2 \ \ (s_1, s_2) \in R\}$. Given two relations $R \subseteq S_1 \times S_2$ and $R' \subseteq S_2 \times S_3$ we can define their composition to be the relation $R' \circ R \subseteq S_1 \times S_3$ defined by $R' \circ R = \{(s_1, s_3) \in S_1 \times S_3 \ : \ \exists s_2 \in S_2 \ \ (s_1, s_2) \in R \wedge (s_2, s_3) \in R'\}$. Given a relation $R \subseteq S_1 \times S_2$ we call $R^{-1} \subseteq S_2 \times S_1$ given by $R^{-1} = \{(s_2, s_1) \in$

$S_2 \times S_1 \ : \ (s_1, s_2) \in R\}$ the inverse relation. An object that we will use frequently is the set valued map $R : S_1 \rightarrow 2^{S_2}$ induced by a relation R and defined by $R(s_1) = \{s_2 \in S_2 \ : \ (s_1, s_2) \in R\}$.

We also introduce some notation for later use. Given relations $R_1 \subseteq S_1 \times S_2$, $R_2 \subseteq S_3 \times S_4$ and a subset $L \subseteq S_1 \times S_3$ we define the new relations $R_{1\times 2}$ and $R_{1\times 2}|_L$ as $R_{1\times 2} = R_1 \times R_2 = \{((s_1, s_3), (s_2, s_4)) \in (S_1 \times S_3) \times (S_2 \times S_4) : (s_1, s_2) \in R_1 \ \wedge \ (s_3, s_4) \in R_2\}$ and $R_{1\times 2}|_L = \{((s_1, s_3), (s_2, s_4)) \in R_{1\times 2} \ : \ (s_1, s_3) \in L\}$.

The product $S_1 \times S_2$ comes equipped with two projection maps $\pi_{S_1} : S_1 \times S_2 \rightarrow S_1$ and $\pi_{S_2} : S_1 \times S_2 \rightarrow S_2$. If we now chose a subset R of the product such that $\pi_{S_1}(R) = S_1$ we can regard this subset R as a fiber bundle over the base space S_1 and we call R a fibering relation. The fiber over $s \in S_1$, denoted by $R_s = \pi_{S_1}^{-1}(s)$ is given by all the elements $r \in R$ such that $\pi_{S_1}(r) = s$. We also denote an element $r = (a, b) \in R$ by b_a when we whish to emphasize the fiber part of r.

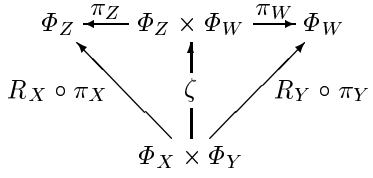
A monoid is a triple $(\mathcal{M}, \cdot, \varepsilon)$ where $\mathcal{M}$ is a set closed under the associative operation $\cdot : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$ and ε is a special element of $\mathcal{M}$ called identity. This element satisfies $\varepsilon \cdot m = m \cdot \varepsilon = m$ 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 $m_1 m = m_2$. Suppose now that we have a fibering relation $R \subseteq S \times \mathcal{M}$ with base space S. If $\pi_S^{-1}(s)$ contains ε and is prefix closed for every $s \in S$ then we call R a fibering monoid.

We now relate relations with fiber bundles and monoids. Suppose that the sets S_1 and S_2 are in fact fiber bundles. Then a relation $R \subseteq S_1 \times S_2$ induces a relation $R_B \subseteq B_1 \times B_2$ on the base spaces B_1 and B_2 of S_1 and S_2, respectively, by declaring $(b_1, b_2) \in R_B$ iff $(s_1, s_2) \in R$, and $\pi_{S_1}(s_1) = b_1$ and $\pi_{S_2}(s_2) = b_2$. If the fiber bundles have a richer structure such as fibering monoids we need the relation to also respect that structure. We then say that a relation $R \subseteq S_1 \times S_2$ between two fibering monoids is fibering monoid respecting iff satisfies:

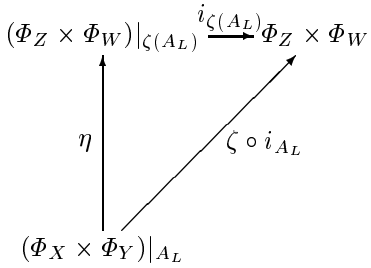
- **Identity:** $(s_1, s_2) \in R_B \ \Rightarrow \ ((s_1, \varepsilon), (s_2, \varepsilon)) \in R$
- **Semi-group:** $((s_1, m_1), (s_2, m_2)), ((s_1', m_1'), (s_2', m_2')) \in R$ and $(s_1, m_1 m_1') \in S_1 \Rightarrow ((s_1, m_1 m_1'), (s_2, m_2 m_2')) \in R$.

B Proofs

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



commutes and this relation is given by $\zeta = (R_X, R_Y) = R_{X\times Y}$, meaning that $\Phi_Z \times \Phi_W$ is a $R_{X\times Y}$-simulation of $\Phi_X \times \Phi_Y$. Consider now the following diagram:



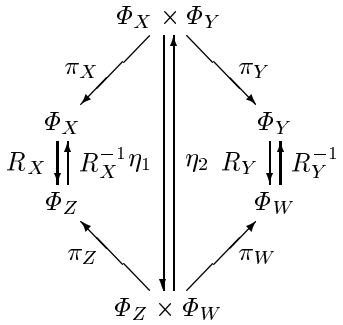
(12)

Once sees that the unique relation η is given by $\eta = \zeta \circ i_{A_L} = R_{X\times Y} \circ i_{A_L}$, that is, η is the relation $R_{X\times Y}$ restricted to the fibering submonoid A_L. From this we conclude that $\Phi_Z \parallel_{R_{X\times Y}(A_L)} \Phi_W$ is a $R_{X\times Y}|_{A_L}$-simulation of $\Phi_X \parallel_{A_L} \Phi_Y$ as desired.

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 R_X-bisimulation of Φ_X), Φ_Y and Φ_W (a R_Y-bisimulation of Φ_Y) the product abstract control system $\Phi_Z \times \Phi_W$ is a $R_{X\times Y}$-bisimulation of $\Phi_X \times \Phi_Y$.*

Proof. Consider the following diagram:



By definition of product there exists one and only one relation η_1 and one and only one relation η_2 such that the diagram commutes. In fact, η_1 is the relation $\eta_1 = (R_X \circ \pi_X, R_Y \circ \pi_Y) = R_{X\times Y}$ and $\eta_2 = (R_X^{-1} \circ \pi_Z, R_Y^{-1} \circ \pi_W) = R_{X\times Y}^{-1}$ meaning that $\Phi_X \times \Phi_Y$ is $R_{X\times Y}$-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 R-bisimulation of Φ_X and A_L a fibering submonoid of A_X such that $R^{-1} \circ R|_{A_L} = id_{A_L}$ and $R \circ R|_{A_L}^{-1} = id_{R(A_L)}$. The restriction $\Phi_X|_{A_L}$ is a $R|_{A_L}$-bisimulation of $\Phi_Y|_{R(A_L)}$.*

Proof. Consider the restrictions (Φ_X, i_L) and $(\Phi_Y|_{R(A_L)}, R^{-1} \circ i_{R(A_L)})$ with $i_L : A_L \rightarrow A_X$ and $i_{R(A_L)} : R(A_L) \rightarrow A_Y$ the inclusion morphisms. Since $i_{A_L}(A_L) = A_L = R^{-1} \circ R(A_L) = R^{-1} \circ i_{R(A_L)}(R(A_L))$ and by definition of restriction there is one and only one morphism η_1 making:

$$\begin{array}{ccc} \Phi_Y|_{R(A_L)} & & \\ \eta_1 \downarrow & \searrow^{R^{-1} \circ i_{R(A_L)}} & \\ \Phi_X|_{A_L} & \xrightarrow[i_{A_L}]{} & \Phi_X \end{array} \tag{14}$$

commutative. A similar argument for the pairs $(\Phi_Y|_{R(A_L)}, i_{R(A_L)})$ and $(\Phi_X|_{A_L}, R \circ i_{A_L})$ assures the existence of a single morphism η_2 such that:

$$\begin{array}{ccc} \Phi_Y|_{R(A_L)} & \xrightarrow{i_{R(A_L)}} & \Phi_Y \\ \eta_2 \uparrow & \nearrow_{R \circ i_{A_L}} & \\ \Phi_X|_{A_L} & & \end{array} \tag{15}$$

commutes. Since $R^{-1} \circ R|_{A_L} = id_{A_L}$ and $R \circ R|_{A_L}^{-1} = id_{R(A_L)}$ we conclude that $\eta_1 = R^{-1} \circ i_{R(A_L)} = \{((y, a_y), (x, a_x)) \in A_Y \times A_X : ((x, a_x), (y, a_y)) \in R \ \wedge \ (x, a_x) \in A_L\}$. Noticing that $\eta_2 = R \circ i_{A_L} = \{((x, a_x), (y, a_y)) \in R : (x, a_x) \in A_L\}$ we clearly have $\eta_1 = \eta_2^{-1}$ meaning that $\Phi_X|_{A_L}$ and $\Phi_Y|_{R(A_L)}$ are bisimilar as desired.

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 R-bisimulation of Φ_X and A_L a fibering submonoid of A_X. If the restriction $\Phi_X|_{A_L}$ is a $R|_{A_L}$-bisimulation of $\Phi_Y|_{R(A_L)}$ then $R^{-1} \circ R|_{A_L} = id_{A_L}$ and $R \circ R|_{A_L}^{-1} = id_{R(A_L)}$.*

Proof. Consider the following commutative diagram:

$$\begin{array}{ccc} \Phi_Y|_{R(A_L)} & \xrightarrow{\quad i_{R(A_L)} \quad} & \Phi_Y \\ R|_{A_L} \Big\uparrow \Big\downarrow {R|_{A_L}}^{-1} & & R \Big\uparrow \Big\downarrow R^{-1} \\ \Phi_X|_{A_L} & \xrightarrow{\quad i_{A_L} \quad} & \Phi_X \end{array} \tag{16}$$

from which we get the following equality:

$$i_{A_L} = R^{-1} \circ i_{R(A_L)} \circ R|_{A_L} = R^{-1} \circ R|_{A_L} \tag{17}$$

that implies $R^{-1} \circ R|_{A_L} = id_{A_L}$. From the diagram we also extract the equality $i_{R(A_L)} = R \circ i_{A_L} \circ R|_{A_L}^{-1}$ which gives us the remaining condition $R \circ R|_{A_L}^{-1} = id_{A_L}$.

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