Hybrid abstractions that preserve timed languages

Paulo Tabuada[1] and George J. Pappas[2]

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

Abstract. In this paper we consider the problem of extracting an abstraction from a hybrid control system while preserving timed languages. Such consistent abstractions are clearly useful as the abstracted, higher level model could be used for controller synthesis or verification of the more complicated lower level model. The class of abstracting maps we consider in this paper compress only the continuous states without aggregating any discrete states. Given such an abstracting map, we determine natural conditions that determine when trajectories of the original hybrid system can be generated by the abstracted hybrid system. Conversely, we determine conditions under which the two hybrid systems generate exactly the same timed language.

1 Introduction

The analysis and synthesis of hybrid control systems has received tremendous attention recently. The scale of the motivating applications, such as air traffic management systems [15] or automotive engine control systems [4], require that the resulting analysis and control methodologies scale up efficiently, in order to facilitate the realistic application of computational methods to real-scale examples.

One of the fundamental approaches to reducing the complexity of large scale system analysis and design is the process of *abstraction*. From an analysis perspective, given a model and a property of interest, one tries to extract a simpler model, an abstraction, that preserves the property of interest while ignoring irrelevant details. This approach has been used successfully in extracting discrete abstractions of hybrid systems while preserving many properties that can be expressed in various temporal logics [3].

From a design perspective, given a hybrid control system, one would like to extract an abstracted hybrid system, perform the design at the higher level abstraction, and then *refine* the design at the lower level. In this hierarchical setting, a methodology which extracts a hierarchy of hybrid system models at various levels of abstraction is critical.

Due to the complexity of combinatorial problems, the notion of abstraction is more mature in theoretical computer science than control theory. For purely discrete systems, the notions of language equivalence, simulation, and bisimulation are established [10]. For purely continuous systems, however, these concepts are only recently beginning to emerge. In particular, in [12], a notion of abstraction for continuous systems was formalized. In [11] reachability preserving abstractions of continuous linear systems were characterized, leading to hierarchical reachability algorithms for linear control systems. In [13], these results where generalized for nonlinear analytic systems. A general theory of abstraction for hybrid systems will clearly merge the continuous and discrete approaches.

In this paper, we address the problem of extracting a hybrid abstraction from a hybrid control model while preserving timed languages. Given a hybrid system, the timed language is simply the timed trajectory of the discrete states. Therefore, the timed language maintains the discrete state the system is in as well as relevant timing information.

This problem is important for a variety for reasons. For scheduling multiple physical processes (such as air traffic management systems), the higher level may be simply interested in which discrete mode each process is in (landing, holding, etc.) and when. Therefore the higher level (air traffic control) would like then to use the simplest possible model of an aircraft that is compatible with the original aircraft dynamics but also with the scheduling operation. Furthermore, the results of this paper can be easily adapted to properly extract hybrid abstractions from purely continuous systems [14]. Finally, the results of the paper are the fist steps towards a more general abstraction methodology for hybrid systems.

In order for the abstracted model to generate the same discrete symbols, we consider aggregating only the continuous dynamics. Abstracting the continuous dynamics while preserving the timed language requires the abstraction process to be done in manner that allows us to detect all the discrete transitions. This places a natural condition between the abstracting maps, guards and invariants of the discrete transitions. Assuming that our aggregating maps satisfy these conditions, we show that hybrid trajectories of the original model can be simulated by the abstracted model. Consequently, the abstracted model also generates the same timed language. In general, the abstracted system is *not* a timed automaton [2], as we may need to preserve richer continuous dynamics in order to properly detect the discrete transitions.

In order to ensure that timed trajectories of the abstracted model are feasible by the original hybrid model, we rely heavily on the abstraction results for continuous systems [13]. These results give us *constructive* methods for extracting hierarchies of nonlinear control systems while preserving exact time controllability. Exact time controllability allows us to preserve a form of *timed* reachability. Using these results, we can place additional conditions on our abstracting maps in order to ensure that in each discrete location, the ability to reach a certain guard at the same time can be done at both levels of abstraction. This allows us to show that the timed language generated at the high level can be implemented at the lower level.

This paper is organized as follows : In Section 2, we review the continuous abstraction methodology as presented in [11, 13]. In Section 3, we define hybrid systems, and determine conditions under which the hybrid abstraction and the original hybrid system model can generate the same timed language. Our constructions are briefly illustrated by a simple example in Section 4, but the reader is referred to a more detailed application in [14]. Section 5 contains interesting issues for further research.

2 Abstractions of Continuous Systems

Contrary to differential equations whose abstractions are characterized by very strict conditions, abstractions of control systems involve only moderate conditions due to the nondeterministic nature of control systems. In subsequent discussion, we assume the reader is familiar with differential geometric concepts at the level presented in [1].

2.1 Abstractions of Control Systems

We begin with an abstract definition of a control system:

Definition 1 (Control System). *A control system $S = (U, F)$ consists of a fiber bundle $\pi : U \longrightarrow M$ called the control bundle and a smooth map $F : U \longrightarrow TM$ which is fiber preserving, that is $\pi' \circ F = \pi$ where $\pi' : TM \longrightarrow M$ is the tangent bundle projection. Given a control system $S = (U, F)$, the control distribution $\mathcal{D}$ of control system S, is naturally defined pointwise by $\mathcal{D}(x) = F(\pi^{-1}(x))$ for all $x \in M$.*

The control space U is modeled as a fiber bundle since in general the control inputs available may depend on the current state of the system. On a local coordinate chart, Definition 1 can be read as $\frac{d}{dt}x = f(x, u)$ with $u \in \pi^{-1}(x)$, therefore recovering the traditional form of the control system. Before introducing the notion of abstraction for continuous control systems, the concept of trajectories of control systems is required:

Definition 2 (Trajectories of Control Systems). *A curve $c : I \longrightarrow M$, $I \subseteq \mathbb{R}_0^+$ is called a trajectory of control system $S = (U, F)$ if there exists a curve $c^U : I \longrightarrow U$ satisfying:*

$$\pi \circ c^U = c$$
$$\frac{d}{dt}c(t) = c_*(\frac{d}{dt}) = c_*(1) = F(c^U)$$

Again in local coordinates, the above definition simply says that $x(t)$ is a solution to a control system if there exists an input $u(t) \in U(x(t)) = \pi^{-1}(x(t))$ satisfying $\frac{d}{dt}x(t) = f(x(t), u(t))$. Our goal is to construct a map $\phi : M \longrightarrow N$, the *abstraction map* or *aggregation map*, that will induce a new control system (U_N, F_N) on the lower dimensional manifold N having as trajectories $\phi(c)$, where c are S trajectories. The concept of abstraction map for continuous control systems is defined as follows:

Definition 3 (Abstraction Map). *Let $S_M = (U_M, F_M)$ and $S_N = (U_N, F_N)$ be two control systems on manifolds M and N, respectively. A map $\phi : M \longrightarrow N$ is called an abstraction or aggregation map iff for every trajectory c^M of S_M, $\phi(c^M)$ is a trajectory of S_N. Control system S_N is called a ϕ-abstraction of S_M.*

The above definition is clearly inspired from the notions of language equivalence and simulation of transition systems [10]. From Definition 3, it is clear that an abstraction captures all the trajectories of the original system, but may also contain redundant trajectories. These redundant trajectories are not feasible by the original system and are therefore undesired.

Since Definition 3 defines abstractions at the level of trajectories, it is difficult to determine whether a control system is an abstraction of another one, since this would require integration of the control systems. One is then interested in a characterization of abstractions which is equivalent to Definition 3 but easily checkable. To pursue this, one needs to introduce the notion of ϕ-related control systems.

Definition 4 (ϕ-related control systems). *Let $S_M = (U_M, F_M)$ and $S_N = (B_N, F_N)$ be two control systems defined on manifolds M and N, respectively. Let $\phi : M \longrightarrow N$ be a smooth map. Then control systems S_M and S_N are ϕ-related iff for every $x \in M$*

$$\phi_* \left(F_M \left(\pi_M^{-1}(x) \right) \right) \subseteq F_N \left(\pi_N^{-1}(\phi(x)) \right) \tag{1}$$

The notion of ϕ-related control systems is a generalization of ϕ-related vector fields commonly found in differential geometry as explained in [11]. It is evident that given two systems that are ϕ-related to a control system their intersection is also ϕ-related. This immediately suggests that given a control system and a map ϕ, there is a *minimal* ϕ-related control system, in which case the inclusion (1) can be replaced by equality[1]. We can now provide the connection between abstractions and ϕ-related control systems:

Theorem 1 ([12, 11]). *Let S_M and S_N be control systems on manifolds M and N, respectively, and $\phi : M \longrightarrow N$ a smooth map. Then S_M and S_N are ϕ-related if and only if S_N is a ϕ-abstraction of S_M.*

The control system S_N is called the *minimal* ϕ-abstraction of a control system S_M iff S_N is the minimal system that is ϕ-related to S_M.

For analytic control systems there is a constructive method which given a control system S_M and a map $\phi : M \longrightarrow N$, generates a ϕ-abstraction S_N. This construction, which generalizes the construction for linear systems described in [11], is now briefly reviewed. The reader is referred to [13] for more details.

Given two distributions $\mathcal{A}$ and $\mathcal{B}$ on manifold M, define a distribution $[\mathcal{A}, \mathcal{B}]$ by declaring $[\mathcal{A}, \mathcal{B}](p)$ to be the subspace of T_pM generated by vectors of the form $[X, Y](p)$, where X,Y are any two analytic vector fields in $\mathcal{A}$ and $\mathcal{B}$ respectively,

[1] Note that this minimal element is unique up to a change of coordinates.

and $[X,Y]$ is their Lie bracket. By resorting to this constructive method, define the distribution $\overline{\mathcal{D}}_M$ as:

$$\overline{\mathcal{D}}_M = \mathcal{K} \cup \mathcal{D}_M \cup [\mathcal{K}, \mathcal{D}_M] \cup [\mathcal{K}, [\mathcal{K}, \mathcal{D}_M]] \cup \ldots \tag{2}$$

where $\mathcal{K}$ is the integrable distribution $Ker(\phi_*)$, ϕ_* is the push forward map of ϕ, and $\mathcal{D}_M$ the distribution associated with control system S_M. Distribution $\overline{\mathcal{D}}_M$ allows us to construct the minimal ϕ-abstraction on N as:

$$\mathcal{D}_N(y) = \phi_* \left(\overline{\mathcal{D}}_M(x) \right) \tag{3}$$

for any $x \in \phi^{-1}(y)$. If S_N is extracted from S_M using this canonical construction, then control system S_N will be referred to as *canonically ϕ-related* to S_M.

2.2 Controllability Equivalence

In general, since the abstracted system is less constrained, the abstracted model may allow evolutions that might not be implementable on the original system. However the original system and its abstraction can still be rendered equivalent regarding some properties of interest. In this paper, we will focus on exact time controllability which is defined using the reachable sets of control system S_M:

Definition 5 (Reachable set [7]). *For each $T > 0$, and each x in M, the set of points reachable from x at time T, denoted by $Reach(x,T)$, is equal to the set of terminal points $c^M(T)$ of S_M trajectories that originate at x.*

Definition 6 (Exact Time Controllability). *A control system is said to be exact time controllable if for any $T > 0$, $Reach(x,T) = M$ for any $x \in M$.*

Consider two systems S_M and S_N and a surjective map $\phi : M \longrightarrow N$. Control systems S_M and S_N are equivalent from an exact time controllability point of view if the following property holds: there exists an S_M trajectory connecting $x_1 \in M$ to $x_2 \in M$ in time T if and only if there exists a S_N trajectory connecting $\phi(x_1) \in N$ to $\phi(x_2) \in N$ also in time T. This property is clearly reminiscent of timed-bisimulations [10].

If we assume that the control system is affine in the control, that is, on local charts it can be written as:

$$F(x,u) = f(x) + \sum_{i=1}^{k} g_i(x) u_i \tag{4}$$

then we can characterize exact time controllability through the Lie algebra generated by $\{g_1(x), g_2(x), \ldots, g_k(x)\}$ and denoted by $Lie_g(S_M)$.

Theorem 2 ([7]). *An analytic control system S_M affine in control, as defined in (4), is exact time controllable if $Lie_g(S_M(x)) = T_x M$ for every $x \in M$.*

We defer the reader to [6, 7] for further details regarding the various notions and concepts of controllability. The main theorem regarding controllability equivalence of abstractions (see [13]) can now be restated as follows:

Theorem 3 (Exact Time Controllability Equivalence). *Let S_M and S_N be two analytic control systems on analytic manifolds M and N, respectively, and let N be an embedded submanifold of M. Let $\phi : M \longrightarrow N$ be an analytic surjective submersion. If S_N is canonically ϕ-related to S_M and*

$$Ker(\phi_*) \subseteq Lie_g(S_M) \tag{5}$$

then S_N is exact time controllable iff S_M is.

Equations (2,3) and Theorem 3 provide a constructive way of building continuous abstractions that propagate reachable sets, and in particular exact time controllability. When additional properties must be propagated, additional constraints must be imposed on the abstracting maps.

3 Hybrid Control Abstractions

Although hybrid abstractions follow the same conceptual ideas of discrete and continuous abstractions, their study is somewhat more involved due to the complicated nature of hybrid trajectories. We start with a hybrid system model that allows different continuous spaces in each discrete location.

Definition 7 (Hybrid Control System). *A hybrid control system is a tuple $H = (X, X_0, S, Inv, R)$ with the following components:*

- *X is the state space of the hybrid control system and is given by a family of smooth manifolds $X = \{M_q\}_{q \in Q}$ indexed[2] by a finite set Q. Each state thus has the form (x, q), where $x \in M_q$ is the continuous part of the state, and $q \in Q$ is the discrete part.*
- *$X_0 = \{M_q^0\}_{q \in Q_0} \subseteq X$ is the set of initial states.*
- *S: $Q \rightarrow \{(U_q, F_q) \ : \ (U_q, F_q)$ is a control system on $M_q\}$ assigns to each discrete state $q \in Q$ a control system (U_q, F_q) which governs the evolution of the continuous part of the state. Thus in discrete location q, the continuous part of the state satisfies $\frac{d}{dt}x = f(x, q, u)$ with $u \in \pi^{-1}(x, q)$.*
- *Inv: $Q \rightarrow 2^X$ assigns to each location $q \in Q$ an invariant set $Inv(q) \subseteq M_q$.*
- *$R \subseteq X \times X$ is a relation capturing the discrete jumps.*

Hybrid systems are typically represented as finite graphs with vertices Q, and edges E defined by

$$E = \{(q, q') \in Q \times Q \mid ((x, q), (x', q')) \in R \text{ for } x \in Inv(q) \text{ and } x' \in Inv(q')\}.$$

[2] When all the manifolds M_q are equal, then the state space X is $X = M \times Q$.

With each edge $e = (q, q') \in E$ we associate a *guard* set defined as

$$Guard(e) = \{x \in Inv(q) \mid ((x,q),(x',q')) \in R \text{ for some } x' \in Inv(q')\}$$

and a set-valued *reset* map

$$Reset(e,x) = \{x' \in Inv(q') \mid ((x,q),(x',q')) \in R\}.$$

Trajectories of the hybrid system H originate at any initial state $(x,q) \in X_0$ and consist of concatenations of *continuous flows* and *discrete jumps*. Continuous flows keep the discrete part of the state constant at q, and the continuous part evolves over time according to the control system $\frac{d}{dt}x = f(x,q,u)$, as long as x remains inside the invariant set $Inv(q)$. If during the continuous flow, it happens that $x \in Guard(e)$ for some $e = (q,q') \in E$, then the edge e becomes *enabled*. The state of the hybrid system may then instantaneously jump from (x,q) to any (x',q') with $x' \in Reset(e,x)$. Then the process repeats, and the continuous part of the state evolves according to the control system $\frac{d}{dt}x = f(x,q',u)$. We shall therefore assume that a trajectory of an hybrid control system is a map[3] ξ from a time set $\mathbb{T}$ to the state space $X = \{M_q\}_{q\in Q}$ of H, that is:

$$\begin{aligned} \xi : \mathbb{T} &\longrightarrow \{M_q\}_{q\in Q} \\ \tau &\mapsto (x(\tau), q(\tau)) \end{aligned} \tag{6}$$

An abstracting map for hybrid systems can now be defined in the same way it was defined for continuous systems.

Definition 8 (Abstraction Map). *Let $H_X = (X, X_0, S_X, Inv_X, R_X)$ and $H_Y = (Y, Y_0, S_Y, Inv_Y, R_Y)$ be two hybrid control systems with $X = \{M_q\}_{q\in Q}$ and $Y = \{N_p\}_{p\in P}$. A map $\phi : X \longrightarrow Y$ is called an abstraction or aggregation map iff for every trajectory c^{H_X} of H_X, $\phi(c^{H_X})$ is a trajectory of H_Y.*

Even though, we are interested in general abstracting maps, we now focus on a subclass of abstracting maps that are suitable for preserving timed languages.

3.1 Timed Language Generated by a Hybrid System

In this paper we shall focus on abstractions that render the original system and its abstraction equivalent regarding the timed language they can generate. The timed string corresponding to a trajectory $\xi(\tau) = (x(\tau), q(\tau))$ of an hybrid control system is simply given by $q(t)$. Naturally $q(t)$ can be regarded as a timed string[4] since it can be written in the more usual form $\{(t, q(t))\}_{t\in\mathbb{R}_0^+}$. The timed language generated by an hybrid control system is therefore defined as:

[3] When multiple discrete jumps in zero time are allowed, a more complex notion of time is required to regard an hybrid trajectory as a map, see for example [9].

[4] The string $s = q(t)$ can be transformed to retain only the discrete states, and the first instance of time at which the system has changed discrete state. The results presented in this paper are however independent of that transformation.

Definition 9 (Timed language of a hybrid system). *Let H be a hybrid control system. The timed language generated by H and denoted by Σ_H is given by all the strings $q(t)$, where $q(t)$ is the discrete part of an hybrid trajectory $\xi(\tau) = (x(\tau), q(\tau))$ of H.*

With this notion of timed language, timed language equivalence between two hybrid system requires the discrete behavior of the hybrid abstraction to be *equal* to the discrete behavior of the original system. Therefore aggregation can only happen on the continuous part of the hybrid system. We will therefore restrict the class of abstracting maps to the following form:

$$\begin{aligned} \phi : \{M_q\}_{q\in Q} &\longrightarrow \{N_q\}_{q\in Q} \\ \phi(x, q) &= (\phi(x), q) \end{aligned} \tag{7}$$

that is, if ϕ is written as $\phi = (\phi_M, \phi_Q)$, then ϕ_Q is the identity map on $Q = P$.

Even though for continuous systems we can always extract abstractions that preserve trajectories, for hybrid control systems additional constraints must be imposed on the abstracting map to ensure timed language equivalence. This is because the discrete dynamics rely heavily on certain sets, such as the guards and the invariants, and we have to ensure that these sets are abstracted correctly at the higher level.

3.2 Propagating Guards and Invariants

Let us zoom into a discrete state and consider the relevant sets which trigger the discrete dynamics, namely the guards and the invariants. Timed language equivalence requires that these sets must be aggregated in a consistent way.

Figure 1 represents the state space of the original system with the guard defined by a relation of the type $x_2 > const$. When performing an abstraction using the map $\phi(x_1, x_2) = x_2$, in the abstracted system it is still possible to determine if the continuous part of the trajectory belongs or not on the guard. No information required by the discrete dynamics was lost in the abstracting process. However if the abstracting map is $\phi(x_1, x_2) = x_1$ it is no longer possible to determine if the continuous part of the trajectory belongs or not to the guard, therefore it is not possible to generate the same timed language.

The essential property to be propagated is therefore the ability to distinguish between sets $\phi(A)$ and $\phi(B)$ in the abstracted system if and only if it is possible to distinguish between relevant sets A and B in the original system. The relevant sets can be encoded in a partition of the state space, where each equivalence class of the partition corresponds to a possible combination of guards and invariants. The required partition can be modeled as a map Ψ_M defined as:

$$\Psi_M : M \longrightarrow D \tag{8}$$

where D is a finite set. We assume that the map Ψ_M results in a topologically well behaved partition[5]. Partition propagation can now be defined as:

[5] For example, the partition can be a subanalytic stratification [8].

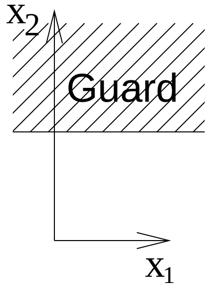

Fig. 1. Detecting a guard.

Definition 10 (Partition Propagation). *An abstracting map $\phi : M \longrightarrow N$ propagates a partition Ψ_M iff there exists a partition on N defined by a map $\Psi_N : N \longrightarrow D$ such that the following diagram commutes.*

$$\begin{tikzcd} M \arrow[rr, "\phi"] \arrow[dr, "\Psi_M"'] & & N \arrow[dl, "\Psi_N"] \\ & D & \end{tikzcd} \tag{9}$$

or equivalently iff $\Psi_M(x) = \Psi_N \circ \phi(x)$.

Note that propagating the partitions is stronger than preserving the partition which only requires that $\Psi_M(x_1) = \Psi_M(x_2) \Rightarrow \Psi_N \circ \phi(x_1) = \Psi_N \circ \phi(x_2)$ and allows, for example, merging two Ψ_M equivalence classes into a single equivalence class in Ψ_N. This is not a desirable situation since the ability to distinguish between the two equivalence classes is lost.

Although Definition 10 captures the fundamental property that the abstracting map should possess it does not characterize it directly. A characterization is given in the following proposition:

Proposition 1. *An abstracting map $\phi : M \longrightarrow N$ propagates a partition Ψ_M iff the preimage under ϕ of a point $y \in N$ is totally contained in a single Ψ_M equivalence class, equivalently, if for all $y \in N$ there exists one and only one $d \in D$ such that $\Psi_M \circ \phi^{-1}(y) = \{d\}$.*

Proof. (Sufficiency) We proceed by contradiction. Suppose that $\Psi_M(x) = \Psi_N \circ \phi(x)$ and there exist two different elements $a, b \in M$ that belong to two different Ψ_M equivalence classes, that is $\Psi_M(a) \neq \Psi_M(b)$. Admit further that they are mapped into the same point in N, $\phi(a) = \phi(b)$. We have that $\Psi_M(a) = \Psi_N \circ \phi(a)$, but since $\phi(a) = \phi(b)$, $\Psi_N \circ \phi(a) = \Psi_N \circ \phi(b) = \Psi_M(b)$. Therefore $\Psi_M(a) = \Psi_M(b)$, a contradiction.

(Necessity) We define explicitly the map Ψ_N as $\Psi_N(y) = \Psi_M(x)$ for all $x \in \phi^{-1}(y)$ which is well defined since $\phi^{-1}(y)$ is contained in a single Ψ_M equivalence class. $\square$

Proposition 1 states partition propagation conditions explicity on the abstracting map ϕ, but they are very difficult to check in general. However it is rather intuitive that a sufficient condition for partition propagation is symmetry, as expressed in the next proposition.

Proposition 2. *Suppose that the partition Ψ_M on manifold M is invariant under the action of a group G, then the abstracting map ϕ defined as the projection from the manifold M to the orbit space M/G propagates the partition Ψ_M.*

Proof. If the Ψ_M equivalence classes are invariant under G action, then the orbit through the point x_0, namely $O_{x_0} = \{x \in M \ : \ x = gx_0 \ \forall_{g \in G}\}$ is contained in a Ψ_M equivalence class. Since the preimages under ϕ are precisely the sets O_{x_0} the conditions of Proposition 1 are satisfied. $\square$

In fact, symmetry is also a necessary condition when more structure is imposed on the set M and the map ϕ. To study general nonlinear abstracting maps we consider that M and N are smooth manifolds and that the abstracting map ϕ is a smooth surjective submersion. Resorting to this differentiable structure, Proposition 1 specializes to:

Proposition 3. *A smooth surjective submersion $\phi : M \longrightarrow N$ between smooth manifolds propagates a partition Ψ_M if and only if the partition equivalence classes are invariant under $Ker(\phi_*)$.*

Proof. (Sufficiency) The vectors in $Ker(\phi_*)$ span an involuntive distribution which has constant rank at every $x \in M$ since the map ϕ is a submersion. By Frobenius theorem [1] there exists an integrating manifold that can be described as the action of $\mathbb{R}^p$, with $p = dim(\mathcal{K})$, on M given by $\gamma = \phi_1(t_1) \circ \phi_2(t_2) \circ \ldots \circ \phi_p(t_p)$. Each $\phi_i(t_i)$ is the flow of the vector field Z^i from the generators of $\mathcal{K}$, that is $\mathcal{K} = Span\{Z^1, Z^2, \ldots, Z^p\}$. The partition equivalence classes are therefore invariant under this action and by Proposition 2 the partition is propagated.

(Necessity) The preimage of a point $y \in N$ by ϕ is a smooth submanifold of M when the derivative of ϕ, is surjective, which is the case since ϕ is an submersion. The tangent space of the submanifold $\phi^{-1}(y)$ is given by the vectors $X \in TM$ that belong to $Ker(\phi_*)$. Since the partition is propagated the preimage of a point $y \in N$ by ϕ is totally contained inside a partition equivalence class and therefore the partition equivalence classes are invariant under $Ker(\phi_*)$. $\square$

The above characterizations of the abstracting maps are critical in order to propagate discrete trajectories from the original hybrid control system to the abstracted one while ensuring timed language equivalence.

3.3 Hybrid Abstractions

Given a hybrid system, H_X and an abstracting map ϕ, we now present a construction that generates an hybrid abstraction H_Y. The abstraction process depends on the observation that the continuous dynamics in a particular discrete state is essentially decoupled from the continuous dynamics in the other discrete state, the only link being given by the *Reset* map. It is therefore possible to use a different abstracting map ϕ_q in each discrete state $q \in Q$ of the hybrid system H_X. More formally:

Definition 11 (Construction of hybrid abstractions). *Consider hybrid control system $H_X = (X, X_0, S_X, Inv_X, R_X)$ with $X = \{M_q\}_{q \in Q}$ and consider the collection of maps $\Phi = \{\phi_q\}_{q\in Q}$, $\phi_q : M_q \longrightarrow N_q$. The resulting hybrid abstraction $H_Y = (Y, Y_0, S_Y, Inv_Y, R_Y)$ is a tuple consisting of:*

- *For all $q \in Q$, $N_q = \phi_q(M_q)$, therefore the state space is $Y = \{N_q\}_{q\in Q}$.*
- *$Y_0 = \{N_q^0\}_{q \in Q_0}$ where $N_q^0 = \phi_q(M_q^0)$.*
- *S_Y is a function that maps each $q \in Q$ to the minimal ϕ_q-abstraction of the corresponding control system $S_X(q)$ using the canonical construction (2,3).*
- *$Inv_Y(q) = \phi_q(Inv_X(q))$.*
- *$R_Y = \{\big((y,q),(y',q')\big) \in Y \times Y \ : \ (y,q) = \phi_q(x,q) \ \wedge \ (y',q') = \phi_{q'}(x',q') \ \wedge \ ((x,q),(x',q')) \in R_X\}$. More specifically we have*
  - *$Guard_Y(e) = \phi_{q_i}(Guard_X(e))$*
  - *$Reset_Y(e, x_i) = \phi_{q_j} \circ Reset_X(e, \phi_{q_i}^{-1}(x_i))$ for all $e = (q_i, q_j) \in E$, $x \in M$.*

Therefore the discrete state space remains unaltered and only the continuous state space is aggregated from M_q to N_q is each discrete location $q \in Q$, and similarly for the set of initial conditions. The continuous control system $S_X(q)$ is replaced by its minimal ϕ_q-abstraction. The new invariant on each location $q \in Q$ is the image of the initial invariant under ϕ_q, that is $\phi_q(Inv_X(q))$. The reset relation R_Y is the image of the reset relation R_X by the abstracting map resulting in the new guards being the image of the initial guards by the abstracting map. The reset maps $Reset_Y$ are given by the image under ϕ_{q_j} of the reset maps $Reset_X$ evaluated at every point of the set valued map $\phi_{q_i}^{-1}$. The main result relating hybrid abstraction constructed through Definition 11 and timed language equivalence can now be stated as follows:

Theorem 4 (Timed language equivalent hybrid abstractions). *Let H_X and H_Y be hybrid control systems and suppose H_Y is obtained from H_X using Definition 11. If the family of maps $\Phi = \{\phi_q\}_{q\in Q}$ is such that the invariants and guards in each discrete location $q \in Q$ are invariant under $Ker(\phi_{q*})$ then H_Y is a Φ-abstraction of H_X.*

If furthermore $Ker(\phi_{q*}) \subseteq Lie_g(S_M(q))$ for each $q \in Q$ then H_X and H_Y generate the same timed language.

Proof. To show that H_Y is a Φ-abstraction of H_X we need to show that for every trajectory $c^{H_X} = (x(\tau), q(\tau))$, $\Phi(c^{H_X})$ is a trajectory of H_Y. For any trajectory $(x(\tau), q(\tau))$ of H_X, $(x(0), q(0)) \in X_0$, therefore $\Phi(x(0), q(0)) = (\phi_{q(0)}, q(0)) \in Y_0$

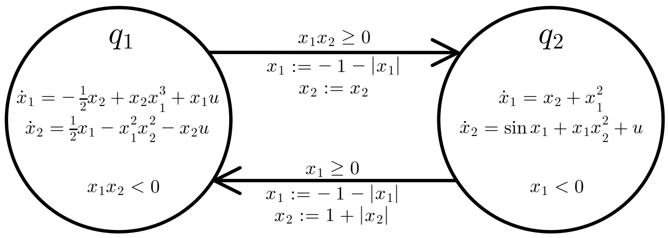

Fig. 2. Hybrid control system H_X.

since $N_q^0 = \phi_q(M_q^0)$. As long as the trajectory c^{H_X} flows continuously on a state $q \in Q$, $x(\tau)$ is a trajectory of $S_X(q)$, therefore $y(\tau)$ is a trajectory of $S_Y(q)$ since $S_Y(p)$ is ϕ_q-related to $S_X(q)$ and $x(\tau) \in Inv_X(q)$ implies $y(\tau) \in Inv_Y(q)$ by construction and partition propagation. When $x(\tau)$ enters a guard $Guard_X(e)$, $y(\tau)$ enters $Guard_Y(e)$ by construction and partition propagation. If the hybrid control system H_X jumps from location q_i to location q_j then H_Y can also take the same transition since the finite graphs of H_Y and H_X are equal and the corresponding transitions become enabled at the same time. After the jump $x(\tau) \in Reset_X(e, x')$ and therefore $y(\tau) \in Reset_Y(e, y')$ by construction of $Reset_Y$. Since the trajectory c^{H_X} is composed of continuous flows and jumps and H_Y simulates both, a finite induction argument on the number of jumps concludes the proof.

To show timed language equivalence it suffices to show that hybrid control system H_X is capable of simulating the continuous part of every H_Y trajectory since both systems have the same finite graph. This is now a direct consequence of using the minimal control abstraction $S_N(q)$ of control system $S_M(q)$ in each discrete location $q \in Q$ as Theorem 3 asserts that both control systems are exact time controllability equivalent. $\square$

4 Example

We illustrate our results by a simple example. Consider the hybrid control system H_X displayed in Figure 2. Using as abstracting maps $\phi_{q_1} = x_1 x_2$ and $\phi_{q_2} = x_1$ we extract the timed language equivalent abstraction presented in Figure 3. Due to space restrictions, we shall present the details regarding state q_2. We start by noting that $Inv(q_2)$ is invariant under $Ker(\phi_{q_2 *}) = K = \frac{\partial}{\partial x_2}$ since K is everywhere tangent to the surfaces $x_1 = const$. The guard is given by the complement of the invariant and is, therefore, also invariant under K. The next step is to determine if ϕ_{q_2} satisfies Theorem 3 conditions, but this is automatically true since $K = g_1(x)$, and therefore $K \in Lie_g\{g_1(x)\} = \{g_1(x)\}$. The new dynamics in each location can be determined through the construction (2, 3). Writing the dynamics as $\dot{x} = f(x) + g(x)u$ we compute $[K, f] = X_1 = \frac{\partial}{\partial x_1} + 2x_1 x_2 \frac{\partial}{\partial x_2}$ and

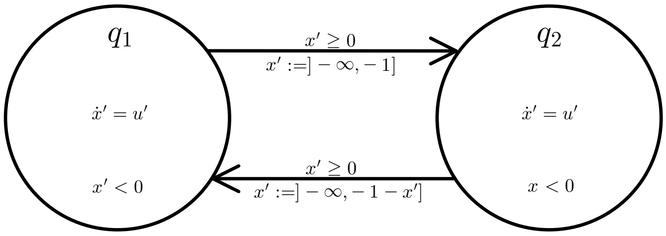

Fig. 3. Hybrid abstraction H_Y of the hybrid control system H_X.

$[K, X_1] = X_2 = 2x_1 \frac{\partial}{\partial x_2}$. However X_2 in linearly dependent on g so that:

$$\overline{\mathcal{D}}_M = \{f, g, X_1\} \tag{10}$$

Computing the pushforward by ϕ_{q_2} of $\overline{\mathcal{D}}_M(x)$ we get:

$$\phi_{q_2 *}\big(\overline{\mathcal{D}}_M(x)\big) = \{(x_2 + x_1^2)\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_1}\} \tag{11}$$

In N coordinates, (given by x'), x_1 equals x' and x_2 is now regarded as a control input v. The new dynamics is then given by $\dot{x}' = 1 + {x'}^2 + v$ and after introducing a new control input given by $u' = v + 1 + {x'}^2$ we get finally $\dot{x}' = u'$. The invariant on N becames $x_1 = x' < 0$ and the guard reads $x' \geq 0$. To determine the new reset map one computes $\phi_{q_2}^{-1}(x') = \{(x_1, x_2) \in M \, : \, x_1 = x' \wedge x_2 \in \mathbb{R}\}$. Using this data the reset map of the hybrid automaton H_X is $x_1 := -1 - |x'| = -1 - x'$ (since the guard is only enable for $x_1 \geq 0$) and $x_2 := 1 + [0, +\infty[= [1, +\infty[$. Aplying ϕ_{q_1} to this reset maps gives the new reset map $x' := (-1 - x')([1, +\infty[) =]-\infty, -1 - x']$.

Note how in this case the nonlinear dynamics could be simplified in such a way that HyTech [5] or other similar tool can be used to analyze the resulting abstraction. For a more complicated example which extracts a hybrid abstraction from a purely continuous system, the reader is referred to [14].

5 Conclusions

In this paper, we have considered the problem of extracting hybrid abstractions from hybrid control systems while preserving timed languages. Generalizing the results of this paper to more general abstracting maps and more general properties is clearly important. Different properties may require different conditions on the abstracting maps, as well as different compatibility conditions between the abstracting maps and the guards, invariants, and continuous dynamics.

Acknowledgment: This work was performed while the first author was visiting the University of Pennsylvania. This research is partially supported by DARPA MoBIES grant F33615-00-C-1707, DARPA JFACC 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.

References

1. R. Abraham, J. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis and Applications.* Applied Mathematical Sciences. Springer-Verlag, 1988.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. 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.
4. A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vicentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proceedings of the IEEE*, 88(7):888–912, July 2000.
5. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science 1019*, pages 41–71. Springer-Verlag, 1995.
6. A. Isidori. *Nonlinear Control Systems.* Springer-Verlag, second edition, 1989.
7. Velimir Jurdjevic. *Geometric Control Theory.* Cambridge University Press, 1997.
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. John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.
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 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.
13. 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.
14. Paulo Tabuada, George J. Pappas, and Pedro Lima. Hybrid abstractions: A search and rescue case study. In *Proceedings of the 2001 European Control Conference*, Porto, September 2001. Submitted.
15. Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management : A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.