HYBRID ABSTRACTIONS : A SEARCH AND RESCUE CASE STUDY

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

[1]Instituto de Sistemas e Robótica
Instituto Superior Técnico
1049-001 Lisboa - Portugal
Fax: (351) 21 841 8291
{tabuada,pal}@isr.ist.utl.pt

[2]Department of EE
University of Pennsylvania
Philadelphia, PA 19104
Fax: (251) 573 2068
pappasg@seas.upenn.edu

Keywords: Hybrid systems, abstractions, timed languages, scheduling.

Abstract

Large-scale, multi-agent systems are becoming extremely complex due to the rapid advances in computation and communication. A natural approach to deal with the increased complexity of such systems is the use of *abstractions*: given a complicated model and some properties of interest, extract simpler models of the original system that propagate the desired properties to the abstracted model, while hiding details that are of no interest. In this paper, we review our methodology for extracting hybrid systems out of continuous control systems while preserving timed languages. This allows us to extract high level models that can be used for real time scheduling while ensuring that high level plans have feasible implementations at the lower level model. Our methodology, is then fully illustrated by a search and rescue case study.

1 Introduction

Complexity reduction in large scale systems is typically reduced by imposing a hierarchical structure on the system architecture. In hierarchical structures, systems of higher functionality utilize coarser models as they are unaware of unnecessary lower level details. Extracting a hierarchy of models at various levels of abstraction is critical for constructing hierarchies in a principled way.

The notions of *abstraction* or *aggregation* refer to grouping the system states into equivalence classes. Depending on the cardinality of the resulting quotient space we may have *discrete* or *continuous* abstractions. With this notion of abstraction, the abstracted system is defined as the induced quotient dynamics. Discrete abstractions of continuous systems have been considered in [3] as well as [4, 12].

In hierarchical systems, one would also like to ensure that certain properties propagate from the abstracted model to the original system. This would ensure that the higher level model is a *consistent abstraction* of the lower level. Different properties may require placing different conditions on the quotient maps.

In previous work, we have focused on extracting *continuous abstractions* from continuous systems. In particular, in [10] a hierarchical framework for continuous control systems was formally proposed, and easily checkable characterizations were obtained for constructing reachability preserving abstractions of linear control systems. In [11], we extended our hierarchical approach to a significant class of nonlinear control systems that consists of analytic control systems on analytic manifolds.

In [13], we presented a methodology for extracting hybrid abstractions from hybrid systems while preserving timed languages. This is a very important problem in various domains. Consider scheduling air traffic near an airport. For the perspective of air traffic control, the only aspect of aircraft dynamics that may be of interest is if the aircraft is at one of finite number of checkpoints as well as when it will reach them. More generally, from a scheduling perspective, the desired property at a higher level can be described as a timed language. Having a methodology that abstracts systems while preserving time languages is critical for the real time scheduling of continuous processes.

In [13], our approach focused on compressing the continuous component of the system while being able to generate exactly the same timed language. Furthermore, compatibility conditions between the abstracting map, and the finite partition of the state space needed to be imposed. Note that, in general, the abstracted system may not be a timed automaton ([2]) as we may need a richer dynamical system in each discrete location to preserve the timed language. In this paper, we review our framework and illustrate its complexity reduction properties on a detailed search and rescue case study.

2 Abstraction Methodology

In this section we will review the hybrid abstraction framework of [13] which builds on top of the continuous abstraction framework of [10, 11]. We assume that the reader is familiar with differential geometric concepts at the level of [1].

2.1 Problem Formulation

We start with a continuous control system F_M, affine in control and defined on a analytic manifold M:

$$\begin{aligned} F_M : M \times U &\rightarrow TM \\ (x,u) &\mapsto f(x) + \sum_{i=1}^{k} g_i(x)u_i \end{aligned} \quad (1)$$

where U is the control space. With this control system we associate a map Ψ defined as:

$$\Psi : M \rightarrow Q \quad (2)$$

where Q is a finite set of symbols. Thus Ψ results in a *finite* partition of the state space and each discrete symbol $q_i \in Q$ is associated with a block of the partition. This map defines a qualitative behavior of control system (1) if regarded as a quantizing or discretizing map of its trajectories. We shall assume that the partition is topologically *nice* without being specific about what nice means[1].

Instead of retaining only the sequence of discrete symbols we want to retain also the temporal properties associated with each symbol, that is we want to propagate the image of control system (1) trajectories under the map Ψ. These trajectories can be seen as maps $q : \mathbb{R}_0^+ \rightarrow Q$ or as a family of pairs $\{(t,q)\}_{t\in\mathbb{R}_0^+} \in \Sigma_{(F_M,\Psi)}$ defined as follows:

$$\begin{aligned} \Sigma_{(F_M,\Psi)} = \{ &(t',q) \in \mathbb{R}_0^+ \times Q : \exists u : \mathbb{R}_0^* \rightarrow U \\ &\frac{d}{dt}x(t) = F_M(x(t),u(t)) \,,\ q = \Psi(x(t')) \\ &\wedge \lim_{t\rightarrow t'^+} \Psi(x(t)) \neq \lim_{t\rightarrow t'^-} \Psi(x(t)) \} \end{aligned} \quad (3)$$

The set $\Sigma_{(F_M,\Psi)}$ is called the *timed language* generated by the pair (F_M, Ψ). The timed language abstracts away unnecessary details regarding the particular value of the solutions, retaining only the time and a qualitative location given by the partition blocks. This type of trajectory description is sufficient for many applications where only timing information and a coarse knowledge of the trajectories is necessary.

Our objective is therefore to abstract the continuous control (1) into a simpler *hybrid system* that is able to generate the same timed language $\Sigma_{(F_M,\Psi)}$. Furthermore, this *hybrid abstraction* should be a *consistent* abstraction of (1), in the sense that any trajectory of the hybrid system can also be generated by (1).

The main idea is to abstract the continuous dynamics in each element of the partition differently. Depending on how the partition interacts with the continuous system, different abstractions could be used in different elements of the partition. The natural way of patching together these new abstracted systems is by defining an hybrid system with several states, each one modeling the behavior of the original system inside a partition element. Before we proceed to hybrid abstractions, we first review the existing theory for continuous abstractions.

[1]For example, subanalytic stratifications [7] or partitions definable in o-minimal theories [8] would suffice.

2.2 Continuous Abstractions

A general methodology for abstracting continuous systems has been introduced in [10] for linear systems and in [11] for nonlinear systems. We will review some results necessary for our hybrid abstraction problem. Given a control system F_M on a analytic manifold M as in (1) we define a surjective abstracting map:

$$\Phi : M \rightarrow N \quad (4)$$

where N is a analytic submanifold of M. The map Φ, which is also assumed to be a submersion, is the *quotient map* which performs the state aggregation[2]. The quotient map will induce a control system F_N on N referred as the *abstraction*. In [11], conditions on the abstracting map were derived to ensure propagation of local reachability ([5], [9]). It is easy to extend these results to exact time controllability [6]. In order to do so we need to define some objects. Let $\mathcal{D}_M$ denote the distribution associated with control system F_M:

$$\mathcal{D}_M = \bigcup_{x\in M} \bigcup_{u\in U} F(x,u) \quad (5)$$

If $\mathcal{A}$ and $\mathcal{B}$ are two distributions on 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. By resorting to this constructive method we define also:

$$\overline{\mathcal{D}}_M = \mathcal{K} \cup \mathcal{D}_M \cup [\mathcal{K},\mathcal{D}_M] \cup [\mathcal{K},[\mathcal{K},\mathcal{D}_M]] \cup \ldots \quad (6)$$

where $\mathcal{K}$ is the distribution $Ker(T\Phi)$. Distribution $\overline{\mathcal{D}}_K$ allows us to specify the abstracted control system on N by:

$$\mathcal{D}_N(q) = T\Phi\overline{\mathcal{D}}_M(p) \quad (7)$$

for any $p \in \Phi^{-1}(q)$. Any control system F_N on N with control distribution $\mathcal{D}_N$ is said to be *canonically* Φ-related to F_M. For constructive cases, the reader is referred to [10, 11].

Let $Lie_g(F_M)$ to be the lie algebra generated by $\{g_1(x), g_2(x), \ldots, g_k(x)\}$ the main result of [11] can now be extended to the following:

Theorem 2.1 *Let F_N be canonically Φ-related to F_M and assume that $\mathcal{K} \subseteq Lie_g(F_M)$ then F_N is exact time controllable iff F_M is.*

[2]Note that any map Φ gives rise to an equivalence relation by defining states x and y equivalent iff $\Phi(x) = \Phi(y)$. In order for the resulting quotient space to have a manifold structure, the equivalence relation must be regular [1].

Proof: Similar to [11] ∎

We have all the tools we need to abstract the continuous dynamics while maintaining exact time controllability, in the sense that if it is possible to go from point $a \in N$ to point $b \in N$ in T units of time in the abstracted model (F_N), then it is also possible to go from $\Phi^{-1}(a)$ to $\Phi^{-1}(b)$ in T units of time in the original model (F_M).

2.3 Partition Propagation

In order to be able to preserve the timed language, the abstracted system needs to be able to determine when the trajectories cross the partition blocks defined by Ψ. To ensure that the partition blocks propagate from M to N, further conditions must be imposed on the abstracting map Φ. A *partition propagating* abstraction map is defined as follows:

Definition 2.2 *An abstracting map $\Phi : M \rightarrow N$ propagates a partition Ψ iff there exists a partition on N defined by a map Ψ' such that the following diagram commutes.*

$$\begin{array}{ccc} N & & \\ \uparrow{\scriptstyle \Phi} & \searrow{\scriptstyle \Psi'} & \\ M & \xrightarrow[\Psi]{} & Q \end{array} \tag{8}$$

or equivalently iff $\Psi(x) = \Psi' \circ \Phi(x)$.

Note that propagating the partitions is a stronger than simply preserving the partition, since this only requires that $\Psi(x_1) = \Psi(x_2) \Leftrightarrow \Psi' \circ \Phi(x_1) = \Psi' \circ \Phi(x_2)$ and allows for example merging two Ψ blocks into a single block in Ψ'. This is not a desirable situation since the knowledge of the qualitative position of the initial system is lost.

Although Definition 2.2 expresses the fundamental property that the abstracting map should possess it does not characterize it directly. This characterization is presented in the next propositions for both linear an nonlinear systems:

Proposition 2.3 *A smooth map $\Phi : M \rightarrow N$ between smooth manifolds preserves a partition Ψ iff the partition blocks are invariant under the action of the lie algebra $Ker(T\Phi)$.*

Proof: See [13]. ∎

2.4 Consistent Hybrid Abstractions

Merging all the conditions that must be imposed on the abstracting map we can characterize timed-language propagating abstraction maps as follows:

Theorem 2.4 *An abstraction map $\Phi : M \rightarrow N$ propagates the timed language $\Sigma_{(F_M,\Psi)}$ generated by a control system F_M iff $Ker(T\Phi) \subseteq Lie_g(F_M)$ and the partition Ψ is invariant under the action of the lie algebra $Ker(T\Phi)$.*

Proof: A direct application of Theorem (2.1) and Proposition (2.3). ∎

Equipped with Theorem 2.4 a general methodology to extract hybrid control system generating the timed language $\Sigma_{(F_M,\Psi)}$ can be described as follows:

For each partition block q_i find the largest lie algebra $\mathcal{L}$ that leaves the partition block invariant. Reduce the control system through an abstracting map Φ_i such that $Ker(\Phi_{i*}) \subseteq \mathcal{L} \cap Lie_g(F_M)$. We shall use the usual hybrid automata conventions to define a hybrid control system state q_i by the following data:

- Abstracted control system: F_{N_i}.
- Invariant: $Inv(q_i) = \Phi_i(\Psi^{-1}(q_i))$.
- Guards: $Guard_j(q_i) = \Phi_i\left(\Psi^{-1}(q_j) \cup \left(\cup_{k \notin J} \Psi^{-1}(q_k)\right)\right)$ $\forall_{j \in J}$ $J = \{j \in \mathbb{N} : q_j \text{ is neighbor of } q_i\}$[3].
- Transitions: Place an arrow from q_i to q_j for all $j \in J$.
- Resets: For each arrow linking q_i to q_j add the corresponding reset map $Reset_{ij} = \Phi_j \circ \Phi_i^{-1}$.

Note that the hybrid control system built according to the above prescription is nondeterministic since the reset maps can be set valued. However all possible trajectories are equivalent with respect to exact time controllability. Equivalently, if it is possible to go from $a \in Reset_{ij}(x)$ to some point b in T units of time then it is also possible to go from any other point $c \in Reset_{ij}(x)$ to b in T units of time. The reader is deferred to [13] for further details.

The above concepts are illustrated in the following example.

3 A search and rescue example

In case of serious natural catastrophes such as earth-quakes, tornados or other equally devastating natural phenomena it is of special importance to perform search and rescue missions. In such a scenario there are several different locations where the necessary intervention could be conducted by specially developed robots. However these search and rescue robotic teams are in limited number and should be distributed through the different emergency areas in the most efficient way possible. This kind of resource allocation must be carefully coordinated by a tactical planner that should take into account the specific kinematic and dynamic constraints of each individual robot. This planner will use a simplified model of the differential equations governing the different robots that preserves the physical constraints of the robots. Such a model will now be developed using the abstracting framework presented in the previous section.

[3] Considering that the partition Ψ is *nice* the informal description q_j is a neighbor of q_i can be replaced by $\overline{\Psi^{-1}(q_j)} \cap \overline{\Psi^{-1}(q_i)} \neq \varnothing$, where the overbar represents the topological closure of a set.

Consider the two-dimensional version of the search and rescue problem for simplicity of presentation. We will consider that the robotic teams are initially at the head quarters modeled as a disk, and that there are $n = 3$ emergency areas where emergency intervention is required. These areas are also modeled by $n = 3$ disks on the plane as can be seen in figure 1. Each disk i is located at (x_i, y_i) and has a radius of r_i.

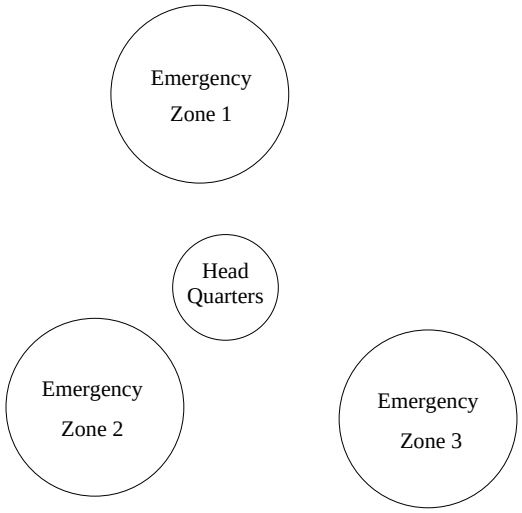

Figure 1: Graphical model of the catastrophe scenario.

By associating a different symbol with each disk and another symbol with the complement of the union of the disks we naturally define the partition Ψ. The tactical planner is interested in retaining the timing information associated with crossing of the disks borders by the robots. This timing information is essential for an effective planning and coordination of the several teams.

We shall illustrate the abstracting methodology for a single robot, since the process is similar for the others. Consider the model of a simple nonholonomic robot given by:

$$\begin{aligned} m\dot{v} &= F \\ I\dot{\omega} &= N \\ \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{9}$$

The x and y coordinates represent robot position, while the angle θ represents its heading angle. The control inputs are given by the total force F acting on the robot and the total torque N, both induced by the torques applied on the robot wheels. By inverting the the mass m and the inertia I we rewrite (9) as:

$$\begin{aligned} F_M &= f(x) + g^1(x)u_1 + g^2(x)u_2 \\ f(x) &= v\cos\theta\frac{\partial}{\partial x} + v\sin\theta\frac{\partial}{\partial y} + \omega\frac{\partial}{\partial\theta} \\ g^1(x) &= \frac{1}{m}\frac{\partial}{\partial v} \\ g^2(x) &= \frac{1}{I}\frac{\partial}{\partial\omega} \end{aligned} \tag{10}$$

The partition Ψ does not depend on the coordinates v or ω, therefore the partition blocks are invariant under the action of the vectors $\frac{\partial}{\partial v}$ and $\frac{\partial}{\partial\omega}$. Since these vectors are also contained in $Lie_g(F_M)$ we can define an abstracting map as:

$$\Phi(v,\omega,x,y,\theta) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{11}$$

This map abstracts away the dynamics retaining only information concerning the kinematics of the vehicle. To determine the abstracted control system we compute:

$$\begin{aligned} \mathcal{K} &= Span\{\frac{\partial}{\partial v}, \frac{\partial}{\partial\omega}\} \\ [\frac{\partial}{\partial v}, \mathcal{D}_M] &= [\frac{\partial}{\partial v}, f(x)] = \cos\theta\frac{\partial}{\partial x} + \sin\theta\frac{\partial}{\partial y} \\ [\frac{\partial}{\partial\omega}, \mathcal{D}_M] &= [\frac{\partial}{\partial\omega}, f(x)] = \frac{\partial}{\partial\theta} \end{aligned} \tag{12}$$

therefore $\overline{\mathcal{D}}_M$ and $\Phi_*\overline{\mathcal{D}}_M$ are given by:

$$\begin{aligned} \overline{\mathcal{D}}_M = Span\{ &\quad \tfrac{\partial}{\partial v}, \tfrac{\partial}{\partial\omega}, v\cos\theta\tfrac{\partial}{\partial x} + v\sin\theta\tfrac{\partial}{\partial y} + \omega\tfrac{\partial}{\partial\theta}, \\ &\qquad \cos\theta\tfrac{\partial}{\partial x} + \sin\theta\tfrac{\partial}{\partial y}, \tfrac{\partial}{\partial\theta}\} \\ \Phi_*\overline{\mathcal{D}}_M = Span\{ &\qquad v\cos\theta\tfrac{\partial}{\partial x} + v\sin\theta\tfrac{\partial}{\partial y} + \omega\tfrac{\partial}{\partial\theta}, \\ &\qquad \cos\theta\tfrac{\partial}{\partial x} + \sin\theta\tfrac{\partial}{\partial y}, \tfrac{\partial}{\partial\theta}\} \end{aligned} \tag{13}$$

and the new control system is given by:

$$\begin{aligned} \dot{x} &= v\cos\theta + \cos\theta \\ \dot{y} &= v\sin\theta + \sin\theta \\ \dot{\theta} &= \omega + 1 \end{aligned} \tag{14}$$

The variables v and ω are now regarded as control inputs and by defining new inputs as $u_1 = v + 1$ and $v_2 = \omega + 1$ we get the usual kinematic model for the nonholonomic robot. Further simplifications are possible by performing another abstraction defined as:

$$\Phi(x,y,\theta) = \begin{bmatrix} x \\ y \end{bmatrix} \tag{15}$$

Q	Description
q_0	Head-Quarters
q_1	Emergency Zone 1
q_2	Emergency Zone 2
q_3	Emergency Zone 3
q_4	Remaining territory

Table 1: Labelings associated with the partition Ψ.

and the resulting control system is:

$$\begin{aligned} \dot{x} &= u_1(\cos\theta - \sin\theta) \\ \dot{y} &= u_1(\cos\theta + \sin\theta) \end{aligned} \tag{16}$$

The variable θ is now also a control input so the system can be rewritten as:

$$\begin{aligned} \dot{x} &= \overline{u}_1 \\ \dot{y} &= \overline{u}_2 \end{aligned} \tag{17}$$

by identifying $\overline{u}_1$ with $u_1(\cos\theta - \sin\theta)$ and $\overline{u}_2$ with $u_1(\cos\theta + \sin\theta)$. When the robot is inside one of the disks its dynamics can be even further reduced by realizing that the disks are invariant under the action of the algebra $Span\{(y-y_i)\frac{\partial}{\partial x} - (x-x_i)\frac{\partial}{\partial y}\}$ which suggests the abstracting map:

$$\Phi(x,y) = z = (x-x_i)^2 + (y-y_i)^2 \tag{18}$$

resulting in the abstracted control system $\dot{z} = u$, which is valid only inside the disks.

To complete the hybrid control system the invariants, guards and resets are computed according the prescription of Section 2 and table 3. For any disk $i = 1, 2, 3$ the invariant is given by:

$$\begin{aligned} Inv(q_i) &= \Phi_i\left(\{(x,y) \in \mathbb{R}^2 \,:\, (x-x_i)^2 + (y-y_i)^2 < r_i^2\}\right) \\ &= \{z \in \mathbb{R} \,:\, z < r\} \end{aligned} \tag{19}$$

and the guard by:

$$\begin{aligned} Guard_4(q_i) &= \Phi_i\left(\Psi^{-1}(q_4) \cup \left(\cup_{k\neq 4}\Psi^{-1}(q_k)\right)\right) \\ &= \Phi_i\left(M \setminus \Psi^{-1}(q_i)\right) \\ &= \{z \in \mathbb{R} \,:\, z \geq r\} \end{aligned} \tag{20}$$

Finally the reset map is given by:

$$\begin{aligned} Reset_{i4}(q_i) &= \Phi_4 \circ \Phi_i^{-1}(z) \\ &= \{(x,y) \in \mathbb{R}^2 \,:\, (x-x_i)^2 + (y-y_i)^2 = z^2\} \end{aligned} \tag{21}$$

Using the same process we get the following data for state q_4:

$$\begin{aligned} Inv(q_4) &= \mathbb{R}^2 \setminus \left(\cup_{i=0,1,2,3}\ \Phi_4 \circ \Psi^{-1}(q_i)\right) \\ Guard_i(q_4) &= \Phi_4 \circ \Psi^{-1}(q_i) \quad i = 0,1,2,3 \\ Reset_{4i}(q_4) &= (x-x_i)^2 + (y-y_i)^2 \quad i = 0,1,2,3 \end{aligned} \tag{22}$$

The resulting hybrid control system is displayed in figure 2.

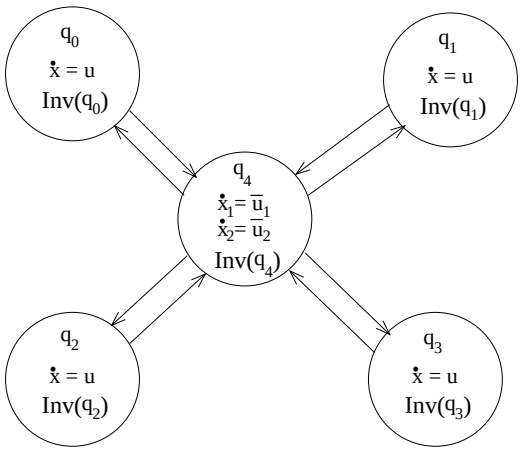

Figure 2: Simplified model of the hybrid control automaton.

The abstracting process reduced the initial control system (9) incorporating dynamics and nonholonomic restrictions to the trivial control systems $\dot{x} = \overline{u}_1$, $\dot{y} = \overline{u}_2$ and $\dot{z} = u$ on a hybrid control system. The resulting abstraction has therefore reduced the complexity of the original system to the minimum amount necessary to generate the same timed language.

4 Conclusions

In this paper a methodology for abstracting control systems to simpler hybrid control systems was discussed. The proposed abstraction framework allows the hybrid control system to generate the same timed language defined by the original control system and a finite partition of the state space. More important is that the language generated in the hybrid abstraction can also be generated by the low level model. A detailed example of application was considered showing the complexity reduction achieved through the abstracting process. The tight conditions on the abstracting map suggest that more general methodologies should be considered for complexity reduction, allowing for example merging several symbols into macro-symbols. This would allow for complexity reduction in the discrete level as well as in the continuous level.

5 Acknowledgments

This work was performed while the first author was visiting the University of Pennsylvania. This research is partially supported by DARPA under grant N66001-99-C-8510, DARPA Grant F33615-00-C-1707, the University of Pennsylvania Research Foundation, and by Fundação para a Ciência e Tecnologia under grant PRAXIS XXI/BD/18149/98 and Project SAPIENS (FCT 33293/99) "RESCUE - Coopertaive Navigation for Rescue Robots".

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] P. E. Caines and Y.J. Wei. Hierarchical hybrid control systems. In S. Morse, editor, *Control Using Logic Based Switching*, volume 222 of *Lecture Notes in Control and Information Sciences*, pages 39–48. Springer Verlag, 1996.

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

[5] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, second edition, 1989.

[6] Velimir Jurdjevic. *Geometric Control Theory*. Cambridge University Press, 1997.

[7] 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.

[8] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.

[9] H. Nijmeijer and A.J. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer-Verlag, 1990.

[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 hierarchies of nonlinear abstractions. In *Proceedings of the 39th IEEE Conference in Decision and Control*. Sydney, Australia, December 2000.

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

[13] Paulo Tabuada and George J. Pappas. Hybrid abstractions of hybrid systems. In *Hybrid Systems : Computation and Control*, Lecture Notes in Computer Science. Springer Verlag, 2001.