Issues and Subtleties in Hybrid Control Systems

Michael S. Branicky

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139 U.S.A.
branicky@lids.mit.edu

Abstract

Here, we look at hybrid control systems as those combining continuous and discrete outputs, states, and controls. We first outline general hybrid control research areas. Then, we give a mathematical model of hybrid control systems. We end by elucidating issues in each of the research areas in light of previous results.

1. Introduction

A hybrid control system consists of a hybrid plant, combining differential equations and automata, governed by a hybrid controller that issues continuous-variable commands and makes logical decisions. See Figure 1. Some examples include computer disk drives, transmissions and stepper motors, constrained robotic systems, and automated highway systems (AHS). More generally, such systems arise whenever one mixes logical decision-making with the generation of continuous control laws. Thus, applications range from programmable logic controllers on our factory floors to flight vehicle management systems (FVMS) in our skies.

"Hybrid" systems, mixing continuous signals and discrete symbols, are certainly pervasive today. But they have been with us at least since the days of the relay. Traditionally, though, the hybrid nature of systems and controllers has been suppressed by converting them into either purely discrete or purely continuous entities. The reason is that science and engineering's formal modeling, analysis, and control "toolboxes" deal largely—and largely successfully—with these "pure" systems.

Engineers have pushed headlong into the application areas above. And the successes in flight control alone attest to the fact that it is possible to build highly complex, highly reliable systems. Yet ever more complex systems continue to arise (viz., FVMS and AHS). And the tools for analyzing and synthesizing such hybrid systems in a systematic—and economic—manner do not yet exist.

Motivated by such real-world problems and concerns, some control theorists and computer scientists have realized that the dichotomy of "pure" systems offers too restrictive a view for many of the complex control tasks we must undertake today, that it is time to focus on developing formal modeling, analysis, and control methodologies for "hybrid systems." In such an endeavor, successful toolboxes from control theory and computer science must not only merge, but be transformed towards dealing with such hybrid problems.

In this paper, we aim to start by posing some general control problems, whose solutions are goals of the hybrid systems community. We then examine some issues and subtleties that arise in the analysis and synthesis of hybrid control systems. In particular, we discuss the vagaries of the pure dichotomies mentioned above and the value of unified, state-space view. We also show the types of phenomena that arise in hybrid systems, which are not present in more conventional controllers, and how they can be handled. This will lead us into a discussion of an approach to control hybrid systems that incorporates digital and analog information and attempts to exploit the hierarchy of discrete symbols over analog signals.

For more details, related work, and references, see [12].

Figure 1: (a) Hybrid System. (b) Hybrid Control System.
2. General Problems

Hybrid systems research may be broken down into four broad categories:

- **Modeling**: formulating precise models that capture the rich behavior of hybrid systems. i.e., *How do we “fill in the boxes” in Figure 1? What is their dynamics? How can we classify their rich structure and behavior—and sort through the myriad hybrid systems models appearing?*

- **Analysis**: developing tools for the simulation, analysis, and verification of hybrid systems. i.e., *How do we analyze systems as in Figure 1(a)? What does continuity mean? What is their complexity? How do they differ from continuous dynamical systems? How do we test their stability? or analyze examples?*

- **Control**: synthesizing hybrid controllers — which issue continuous controls and make discrete decisions—that achieve certain prescribed safety and performance goals for hybrid systems. i.e., *How do we control a plant as in Figure 1(b) with a controller as in Figure 1(f)? How can we synthesize such hybrid controllers?*

- **Design**: conceiving new schemes and structures leading to easier modeling, verification, and control.

3. Paradigms for Hybrid Systems

I see four basic paradigms for the study of hybrid systems: aggregation, continuation, automatization, and systemization.

3.1. Pure Paradigms

The first two approaches deal with the different sides—analog and digital—of hybrid systems. They attempt to suppress the hybrid nature of the system by converting it into a purely discrete or purely continuous one, respectively.

**Aggregation.** That is, suppress the continuous dynamics so that the hybrid system is a finite automaton or discrete-event dynamical system (DEDS). This is the approach most often taken in the literature, e.g., [3]. The drawback of this approach is three-fold.

- **Nondeterminism**: one usually obtains nondeterministic automata [3, 23].
- **Nonexistence**, i.e., even if clever constructions are used, no finite automaton may exist that captures the combined behavior [20].
- **Partition Problem.** It appears a conceptually deep problem to determine when there exist partitions of just a continuous system such that its dynamics is captured by a meaningful finite automaton. “Meaningful,” since we note that every system is homomorphic to one with a single equilibrium point [27]. The answer thus depends on the dynamical behavior one is interested in capturing and the questions one is asking.

The aggregation program has been fully carried out so far only under strong assumptions on the hybrid system [1, 20].

**Continuation.** The complement of aggregation, that is, suppress the discrete dynamics so that the hybrid system becomes a differential equation. This original idea of Prof. Sanjoy Mitter and myself is to convert hybrid models into purely continuous ones—modeled by differential equations—using differential equations that simulate finite automata. In this familiar, unified realm one could answer questions of stability, controllability, and observability, converting them back to the original model by taking a “singular limit.” For instance, one would like tools that allow one to conclude the following: if a “sufficiently close” continuation of a system is stable, then the original system is stable. Such a program is possible in light of the existence of simple continuations of finite automata [13, 16] and pushdown automata and Turing machines [13]. The drawback of this approach is three-fold.

- **Arbitrariness**, i.e., how one accomplishes the continuation is largely arbitrary. For example, to interpolate or “simulate” the step-by-step behavior of a finite automaton Brockett used his double-bracket equations [17] and the author used stable linear equations [7, 13]. In certain cases this freedom is an advantage. However, care must be taken to insure that the dynamics used does not introduce spurious behavior (like unwanted equilibria) or that it itself is not hard to analyze or predict.

- **Hiding Complexity.** One cannot generally get rid of the underlying discrete dynamics, i.e., the complexity is merely hidden in the “right-hand side” of the continuation differential equations [9].

- **Artificiality.** It can lead to a possibly unnatural analytical loop of going from discrete to continuous and back to discrete. Cf. Chen’s recent
results in stochastic approximation vis-à-vis Kushner’s [16].

The combination of these points has been borne out by some experience: it can be easier to examine the mixed discrete-continuous system. Cf. my switched aircraft controller analysis [10] and Megretsky’s relay system analysis [24].

3.2. Hybrid Paradigms
The last two approaches are more general and potentially more powerful. Under them, a hybrid system is seen directly as an interacting set of automata or dynamical systems; they complement the input-output and state-space paradigms, respectively, of both control theory and computer science.

Automatization. Treat the constituent systems as a network of interacting automata [26, p. 325]. The focus is on the input-output or language behavior. This language view has been largely taken in the computer science literature in extending the dynamical behavior of finite automata incrementally toward full hybrid systems (see [1, 21] for background).

Automatization was pioneered in full generality by Nerode and Kohn [26]. The viewpoint is that systems, whether analog or digital, are automata. As long as there is compatibility between output and input alphabets, links between automata can be established. However, there is still the notion of “reconciling different time scales” [26, p. 325]. For instance, a finite automaton receives symbols in abstract time, whereas a differential equation receives inputs in “real time.” This reconciliation can take place by either of the following: forcing synchronization at regular sampling instants [26, p. 333], or synchronizing the digital automaton to advance at event times when its input symbols change [26]. For hybrid systems of interest, the latter mechanism appears more useful. It has been used in many hybrid systems models, e.g., [3, 17]. For a fruitful example of this approach see [19].

Systemization. Treat the constituent systems as interacting dynamical systems [27]. The focus is on the state-space. This state-space view has been taken most profitably in the work of Witsenhausen [30] and Tavarnini [28].

Systemization is developed in full generality in the thesis [12]. The viewpoint is that systems, whether analog or digital, are dynamical systems. As long as there is compatibility at switching times when the behavior of a system changes in response to a logical decision or event occurrence, links between these dynamical systems can be established. Again, there is still the notion of recon- ciling dynamical systems with different time scales (i.e., transition semigroups). For instance, a finite automaton abstractly evolves on the positive integers (or on the free monoid generated by its input alphabet), whereas a differential equation evolves on the reals. This reconciliation can take place by either or both of the following: sequentially synchronizing the dynamical systems at event times when their states enter prescribed sets, or forcing uniform semigroup structure via “timing maps.” Both approaches are introduced here, but the concentration is on the former.

Systemization is established in my formulation of hybrid dynamical systems above. It has been used in examining complexity and simulation capabilities of hybrid systems [13], analyzing their stability [10, 11], and in establishing the first comprehensive state-space paradigm for the control of hybrid systems [14].

4. A Mathematical Model of Hybrid Systems
Briefly, a hybrid dynamical system is an indexed collection of dynamical systems along with some map for “jumping” among them (switching dynamical system and/or resetting the state). This jumping occurs whenever the state satisfies certain conditions, given by its membership in a specified subset of the state space. Hence, the entire system can be thought of as a sequential patching together of dynamical systems with initial and final states, the jumps performing a reset to a (generally different) initial state of a (generally different) dynamical system whenever a final state is reached.

Formally, a controlled general hybrid dynamical system (CGHDS) is a system $H_c = (Q, \Sigma, A, G, V, C, F)$, with constituent parts as follows.

- $Q$ is the set of index states, or discrete states.
- $\Sigma = \{\Sigma_q\}_{q \in Q}$ is the collection of controlled dynamical systems, where each $\Sigma_q = (X_q, \Gamma_q, f_q, U_q)$ (or $\Sigma_q = (X_q, \Gamma_q, \phi_q, U_q)$) is a controlled dynamical system. Here, the $X_q$ are the continuous state spaces and $\phi_q$ (or $f_q$) are called the continuous dynamics.
- $A = \{A_q\}_{q \in Q}$, $A_q \subset X_q$ for each $q \in Q$, is the collection of autonomous jump sets.
- $G = \{G_q\}_{q \in Q}$, where $G_q : A_q \times V_q \to S$ is the autonomous jump transition map, parameterized by the transition control set $V_q$, a subset of the collection $V = \{V_q\}_{q \in Q}$; they are said to represent the discrete dynamics.
• $C = \{C_q\}_{q \in Q}$, $C_q \subseteq X_q$, is the collection of controlled jump sets.

• $F = \{F_q\}_{q \in Q}$, where $F_q : C_q \rightarrow 2^S$, is the collection of controlled jump destination maps.

Thus, $S = \bigcup_{q \in Q} X_q \times \{q\}$ is the hybrid state space of $H$. The case where the sets $U_q$ and $G$ through $F$ above are empty is simply a general hybrid dynamical system (GHDS): $H = [Q, \Sigma, A, G]$.

A CGHDS can be pictured as an automaton as in Figure 2. There, each node is a constituent dynamical system, with the index the name of the node. Each edge represents a possible transition between constituent systems, labeled by the appropriate condition for the transition's being "enabled" and the update of the continuous state (cf. [22]). The notation $!([\text{condition}]$ denotes that the transition must be taken when enabled. The notation $?[\text{condition}]$ denotes an enabled transition that may be taken on command; "$\in$" means reassignment to some value in the given set.

![Figure 2: Automaton Associated with CGHDS.](image)

Roughly, the dynamics of $H_0$ are as follows. The system is assumed to start in some hybrid state in $S \setminus A$, say $s_0 = (x_0, q_0)$. It evolves according to $\phi_{q_0}(\cdot, v)$ until the state enters—if ever—either $A_{q_0}$ or $C_{q_0}$ at the point $s^+_1 = (x^+_1, q_0)$. If it enters $A_{q_0}$, then it must be transferred according to transition map $G_{q_0}(x^+_1, v)$ for some chosen $v \in V_{q_0}$. If it enters $C_{q_0}$, then we may choose to jump and, if so, we may choose the destination to be any point in $F_{q_0}(x^+_1)$. Either way, we arrive at a point $s_1 = (x_1, q_1)$ from which the process continues. See Figure 3.

The following are some important notes about CGHDS:

Dynamical Systems. GHDS with $|Q| = 1$ and $A$ empty recover all these.

Hybrid Systems. The case of GHDS with $|Q|$ finite, each $X_q$ a subset of $\mathbb{R}^n$, and each $I_q = \mathbb{R}$ largely corresponds to the usual notion of a hybrid system, viz. a coupling of finite automata and differential equations [13, 14, 21]. Herein, a hybrid system is a GHDS with $Q$ countable, and with $I_q \equiv \mathbb{R}$ (or $\mathbb{R}_+$) and $X_q \subset \mathbb{R}^d$, $d_q \in \mathbb{Z}_+$, for all $q \in Q$: $\mathcal{H} = [Q, \{X_q\}_{q \in Q}, \mathbb{R}_+, \{I_q\}_{q \in Q}, \mathbf{A}, G]$.

Changing State Space. The state space may change. This is useful in modeling component failures or changes in dynamical description based on autonomous—and later, controlled—events which change it. Examples include the collision of two inelastic particles or an aircraft mode transition that changes variables to be controlled [25]. We also allow the $X_q$ to overlap and the inclusion of multiple copies of the same space. This may be used, for example, to take into account overlapping local coordinate systems on a manifold [4].

Refinements. We may refine the concept of, say, GHDS $H$ by adding:

- inputs, including control inputs, disturbances, or parameters (see controlled HDS below).

- outputs, including state-output for each constituent system as for dynamical systems [12] and edge-output: $H = [Q, \Sigma, A, G, O, \eta]$, where $\eta : A \rightarrow O$ produces an output at each jump time.

- $\Delta : A \rightarrow \mathbb{R}_+$, the jump delay map, which can be used to account for the time which abstracted-away, lower-level transition dynamics actually take.\(^1\)

- Marked states (including initial or final states), timing, or input and output for any constituent system.

Other Notes. (1) Nondeterminism in transitions may be taken care of by partitioning $[\text{condition}]$ into those which are controlled and uncontrolled (cf. DEDS) Disturbances (and other nondeterminism) may be modeled by partitioning $U, V$, and $C$ into portions that are under the influence of the

\(^1\)Here, we may take the view that the system evolves on the state space $\mathbb{R}^n \times Q$, where $\mathbb{R}^n$ denotes the set of finite, but variable-length real-valued vectors. For example, $Q$ may be the set of labels of a computer program and $x \in \mathbb{R}^n$ the values of all currently allocated variables. This then includes Simins' state machines [6].

\(^2\)Think of modeling the closure time of a discretely-controlled hydraulic valve or trade mechanism imperfections in economic markets.

339
controller or nature respectively. Systems with state-output, edge-output, and autonomous and controlled jump delay maps (ΔΔ and ΔΔ, respectively) may be added as above. (2) The model includes the "unified" model posed by Branicky, Borkar, and Mitter (BBM; [14]) and thus several other previously posed hybrid systems models [3, 4, 17, 26, 28, 30]. It also includes systems with impulse effect [5] and hybrid automata [19]. (3) In particular, our unified BBM model is, briefly, a controlled hybrid system, with the form $[Z_+; \{(X^A)_{t \in \mathbb{Z}_0}, X^+, \{J^A\}_{t \in \mathbb{Z}_0}, U\}; A, V, G, C, F]$. The admissible control actions available in this model are the continuous controls $\alpha \in U$, exercised in each constituent regime, the discrete controls $\nu \in V$, exercised at the autonomous jump times (i.e., on hitting set $A$); and the intervention times and destinations of controlled jumps (when the state is in $C$). Control results for this model are given in [14, 15].

5. Issues

Modeling. There are a myriad of hybrid systems models (see [12] and the collections [21, 2] for references). Of less prevalence are results proven about these models. I mention a few here. Tavernini [28], for instance, has given a model plus assumptions and proven properties of trajectories and simulations of trajectories of hybrid systems. In early work, Wittenhausen [30] gave a model and derived some optimal control results. Deshpane [19] has taken an automata approach to modeling and derived results for viable control of their hybrid systems. Above, and in [14, 12] I have a model which (1) captures the phenomena associated with hybrid systems, (2) subsumes many of the models presented in the literature, (3) is amenable to the posing and solution of optimal control problems (see the discussion below).

One needs to explore the plethora of modeling choices available in hybrid systems. Since hybrid systems include dynamical systems as a subset, subclasses which permit efficient simulation, analysis, and verification should be explored. I believe that such a program is indeed being carried out by the computer scientists. Control theorists should do the same in their field in examining the hybrid control of hybrid systems.

Analysis. In general, hybrid systems analysis is theoretically intractable. The reason is that even simple hybrid systems possess the power of universal computation [1, 12]. Thus, any purported general solution of reachability or stability would lead to the impossible situation of a solution to the halting problem. There are, however, at least two ways to proceed. The first is to limit the hybrid systems one wants to consider. The second is to use general conditions for, say, stability of hybrid systems as sufficient conditions for design, i.e., design controls so that the system is easy to analyze.

The first approach has been taken by the computer scientists (as mentioned above). The second approach has had success in control theory. For example, even though it is in general, notoriously hard to find Lyapunov functions to prove stability. Yet, most stability proofs in adaptive control treat Lyapunov stability as a sufficient condition and design stable adaptive controllers by first choosing a Lyapunov function and then controls that enforce its decay.

In the case of hybrid systems, the analysis tools I have developed [10, 11] in general deal not only with the behavior of the constituent systems, but also impose conditions on the behavior of the system at switching times. Thus, control design arising from the constraints of [11] is a topic of further research.

There are also theoretical issues to be explored. Some examples include the stability of systems with multiple equilibrium points, the stability of switched systems, relations between fixed-point theory and Lyapunov stability, and the stability and dynamics of ordinary differential equations driven by Markov chains whose transition probabilities are a function of the continuous state. The latter may provide a link to the large literature on jump systems.

Another important topic of further research is to incorporate developed analysis tools into software engi-
neering tools. This will allow application of these tools to complicated examples in a timely manner.

Control. Specific open theoretical issues were discussed in [14]. Another has to do with the robustness of our hybrid controls with respect to state. Here, the transversality assumptions made in [14] should combine with Taunton's result on continuity with respect initial condition to yield continuity of control laws on an open dense set.

Algorithms for synthesizing optimal hybrid controllers have also been given [12, 15], along with some academic examples. An important area of current research is to develop good computational schemes to compute near-optimal controls in realistic cases. Analysis of rates of convergence of discretized algorithms should be explored. Later, the development of software tools to design such controllers automatically will become an important area of research.

Design. Finally, from modeling, through analysis and control, we come to design of complex, hybrid systems. Here, some of the interaction between levels is under our jurisdiction. What would we do with such freedom, coupled with our new-found analysis and control techniques? For example, we might design a flexible manufacturing system that not only allows quick changes between different product lines, but allows manufacture of new ones with relative ease.

Consider the so-called reflex controllers of [29], which constitute a dynamically consistent interface between low-level servo control and higher-level planning algorithms that ensures obstacle avoidance. Thus as a step in the direction of hierarchical design, the reflex control concept is an example of how to incorporate a new control module to allow rapid, dynamically transparent design of higher-level programs. Thus, these designs incorporate structures which allow engineers to separate the continuous and logical worlds. Ultimately, it is hoped they will lead to truly intelligent engineering systems.

References