Effective Synthesis of Switching Controllers for Linear Systems

EUGENE ASARIN, OLIVIER BOURNEZ, THAO DANG, ODED MALER, AND AMIR PNUELI, MEMBER, IEEE

Invited Paper

In this paper, we suggest a novel methodology for synthesizing switching controllers for continuous and hybrid systems whose dynamics are defined by linear differential equations. We formulate the synthesis problem as finding the conditions upon which a controller should switch the behavior of the system from one "mode" to another in order to avoid a set of bad states and propose an abstract algorithm that solves the problem by an iterative computation of reachable states. We have implemented a concrete version of the algorithm, which uses a new approximation scheme for reachability analysis of linear systems.

Keywords*—Control synthesis, hybrid systems, switched systems.*

I. Introduction

"The purpose of control is to alter the dynamical behavior of a physical system in accordance with man's wishes" [1]. In other words, control theory is concerned with finding systematic ways to influence the behavior of systems by observing their state and injecting appropriate control signals (see, e.g., [2] for a modern exposition). Classically, both the system to be controlled ("plant" in the control terminology) and the controller were modeled as continuous dynamical[1] systems, and the control signal was "computed" continuously over time. This was a natural model when control was implemented using analog devices.

The introduction of digital control (and in particular, control by general-purpose computers) gave rise to models based on discrete-time and sampling [4]. In this setting, time is discretized into fixed intervals. During every interval the values of the state variables are measured and digitized, and the values to be delivered by the feedback function are computed and converted back to analog signals. The nature of the continuous process underlying the plant dictates which sampling rate is sufficient so that the sampled control can be viewed as a good approximation of the "ideal" continuous controller. If the computer is fast enough for computing the feedback function within the sampling period, sampled control can be treated by techniques similar to classical ones. This might explain why some control theorists do not always understand immediately what novelty there is in these hybrid "discrete-continuous" systems.

We argue that the major motivation for hybrid systems is not necessarily the introduction of computers into the feedback loop but rather the inadequacy of models based on continuous mathematics for describing certain classes of complex systems. Notwithstanding the mathematical beauty of the calculus, smooth functions, differential equations, etc., and their effectiveness in predicting stellar and missile motion and much more than that, they *do not* constitute a universal modeling language. A decision of a robot to go left or right, a command of a process control system to open a valve, shifting gears in a car—all are phenomena whose most natural and useful models contain discrete components, and attempts to express them exclusively using the tools of classical continuous mathematics are, perhaps, as adequate as epicycles (similar arguments can be found in [5]). Note that this is not an argument about the "real" nature of the world but rather about the utility of certain classes of mathematical models.

People designing real control systems have always known these facts, and control by switching (i.e., discrete change in the dynamics) has been used since the early days of control. This practice was sometimes accompanied by serious attempts of mathematical formalization, e.g., [6]. However, the

Manuscript received December 20, 1999; revised April 10, 2000.

E. Asarin, T. Dang and O. Maler are with VERIMAG, Gieres 38610, France (e-mail: Eugene.Asarin@imag.fr; Thao.Dang@imag.fr; Oded.Maler@imag.fr).
O. Bournez is with LORIA, Nancy 54602, France (e-mail: Olivier.Bournez@loria.fr).
A. Pnueli is with the Weizmann Institute of Science, Rehovot 76100, Israel (e-mail: amir@wisdom.weizmann.ac.il).

Publisher Item Identifier S 0018-9219(00)06461-6.

[1]Throughout this paper, the term "continuous dynamical system" is used in the narrow sense of a system whose behaviors are solutions to differential equations, e.g., [3].

feeling is that in all these attempts, the discrete dynamics was a second-class citizen,[2] a kind of hacking to glue together several respectable continuous models. The reason for this seems obvious: the existence of a rich and beautiful theory of continuous systems, a product of several centuries, and the lack of a similar theory in the discrete side.[3] There was some theoretical recognition of the existence of dynamical systems whose nature cannot be captured in a useful way by continuous models, and this led eventually to the theory of discrete-event dynamical systems (DESs), for which a controller synthesis methodology, essentially based on automata theory, has been developed by Ramadge and Wonham [8]. However, these models, (with some exceptions; see [9]) were kept apart from continuous models. It was only within computer science (in particular in the domains of verification and semantics) that discrete nonmetrical dynamical systems were studied as a primary object, deserving the attention of both (software and hardware) engineers and computer scientists.[4] Now the time is perhaps ripe for a real theory of hybrid control that can pick useful ingredients from both theories of dynamical systems and pass on the rest with silence (a primal sketch of a basis of such a theory can be found in [10]).

In addition to the theory of automata, an important potential contribution of computer science to control is in the emphasis on *computation* in both design and implementation of control systems. Computations during the design phase are those performed *off-line* by the control engineer in order to predict the behavior of the controlled system. Those might be analytical manipulations of formulas, "experiments" with simulations, etc. Like in any other engineering domain, computer-aided design systems are crucial for designing complex and reliable control systems, and the major contribution of this paper is in automating part of the design process of switching controllers. The outcome of the design stage includes the feedback map, a function to be computed by the controller *on-line* during its operation each time it samples the state of the plant. This computation falls into the category of "real-time" computation, a domain where there is still a lot to be done in terms of clean formalization and bridging the gap between the views of control and software engineers, but this is not the subject of this paper.

A feature of hybrid systems that makes controller design harder is that discrete transitions break down the possibility of finding nice analytical solutions of the control problem. Consider, for example, a linear system $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ for which one can, at least in an idealized setting, compute a stabilizing feedback function directly by looking at A and B. The introduction of nontrivial switching into the dynamics spoils all this beauty, and one has to resort, sometimes, to brute-force approaches of exploring the state space, inspired by algorithmic verification of automata where an analytic solution has never been a dominant option.

In this paper, we introduce a simple framework for studying control by switching, using the commonly accepted model of a hybrid automaton [11], [12]. In this model, a system can be in one of several "modes," in each of which its behavior is governed by a distinct continuous dynamical law. At certain parts of the continuous state space, the system can switch from one mode to another. We formulate the problem of controller synthesis as determining these switching surfaces so that all trajectories generated by the system satisfy some performance criteria. We present an abstract algorithm to solve this problem, which uses as a major component a procedure for computing sets of reachable states (sometimes called "flow pipes" or "tubes of trajectories") of continuous systems. In order to make this algorithm concrete, we use a novel technique of approximating such sets for linear differential equations [13] to give a switching controller with guaranteed correctness. This algorithm has been implemented. The bottom line of this work is that if you have a plant modeled as a piecewise-linear dynamical system and you design a controller capable of switching between modes, you can suggest potential switching regions in the state space and our (implemented) algorithm will synthesize safe switching surfaces in a *fully automatic* manner.

The rest of the paper is organized as follows: in Section II, we introduce the minimal necessary definitions concerning discrete and continuous behaviors and hybrid automata. A special effort is made to relate the definitions to those used in the control literature. In Section III, we formulate the controller synthesis problem and give the abstract algorithm that solves it. In Section IV, we discuss computability issues and our approximation scheme, and in Section V, we illustrate the approximation with two concrete examples. In the last section, we discuss the location of this work within the hybrid systems literature on controller synthesis and on approximate computation of reachable sets.

II. Switched Systems and Hybrid Automata

We start with a control-oriented presentation of the setting before introducing hybrid automata. The system depicted in Fig. 1 is defined over a continuous *time domain* $T = \mathbb{R}_+$ and a continuous state space $X \subseteq \mathbb{R}^n$, whose elements we write as $\boldsymbol{x} = (x_1, \cdots, x_n)$. The system can be in several modes, each with a distinct continuous dynamics. The origin of these modes can be of various sorts: they can be identified with different structural configurations of a continuous system such as gears in a car or combinations of open and closed valves in a liquid container; they can represent several continuous regulators, each used at a different range of operation; they can

[2]As exemplified by the term "differential equation with discontinuous right-hand side." See also an interesting recent survey of results on switched systems [7], which treats similar problems without mentioning an automaton.

[3]This does not mean that discrete mathematics did not exist before, but the development of a decent notion of a function over nonmetric domains was initiated only in the nineteenth century by Boole. The development of an explicit mathematical model of a discrete nonmetric dynamical system (the automaton) had to wait to the works of Turing, Kleene, and von Neumann.

[4]In computer science, due to various factors, the distance between engineers, applied, and pure theoreticians is much smaller than in other disciplines. It is not uncommon to find researchers of the foundations of computing who also have experience in building real systems. It is less likely to see, for example, a Lie algebraist actually involved in the control of a physical system of the type his mathematics is supposed to describe.

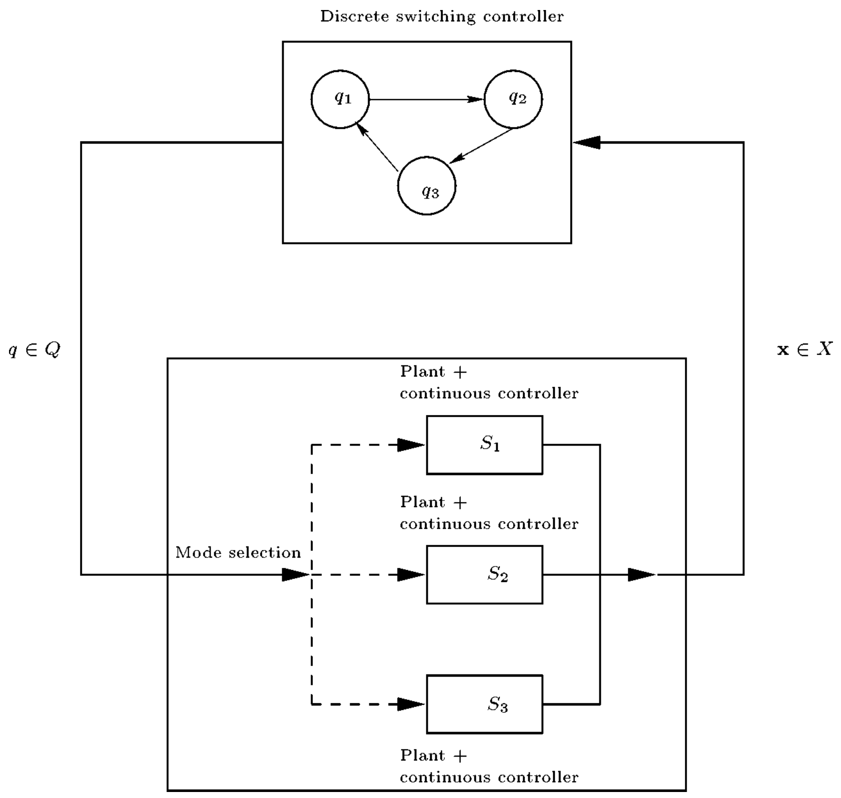

Discrete switching controller
q_1
q_2
q_3
$q \in Q$
$\mathbf{x} \in X$
Plant + continuous controller
S_1
Plant + continuous controller
Mode selection
S_2
S_3
Plant + continuous controller

Fig. 1. A switching controller.

be used to approximate continuous–valued control by a finite discretization or nonlinear systems by piecewise-linear ones. The choice between the modes is done by a discrete controller (supervisor, decision-maker), which observes continuously the state of the plant and decides continuously which mode to select out of a discrete (and usually finite) set.

Some features distinguish our treatment of this setting from the way it is approached in some of the control literature.

1) Hybrid State Space: We are looking closely at the structure of the discrete switching controller and model it as an automaton with a set Q of states, where each state is identified with the value sent by the controller to the plant. Hence, the domain of the feedback map is $Q \times X$ and not only X, and the controller may react differently to the same $\boldsymbol{x} \in X$ according to its current mode. In other words, we consider $Q \times X$ as the state space of the combined system to which we apply all our analysis and synthesis techniques. In this context, the hybrid automaton model is most natural and phenomena such as certain types of hysteresis are easy to model.[5]

2) Nondeterminism: This feature might seem strange to some control theorists—we do not insist (at least not all the time) that the controller is deterministic, but are rather satisfied with a feedback map of the form c: $Q \times X \to 2^Q$. This means that at some parts of the state space there is more than one possibility for the continuous trajectory to evolve, either according to the current mode or by switching to one of several other modes. Unlike control, where differential inclusions [15] are not considered a mainstream topic, in computer science such a set-valued[6] nondeterminism is commonplace, and the archaic "uniqueness of solutions" is traded for higher expressive power. Nondeterminism can express uncertainty in models, sensors, actuators, and disturbances. It can also be used to specify the possible behaviors under all potential controllers before synthesis is done. Synthesis itself can then be viewed as replacing an underspecified controller c with a more restrictive one c' such that all the behaviors induced by c' satisfy the required properties. Hopefully, this will become clearer in the sequel.

3) Properties: In formal verification of discrete (software and hardware) systems, one is interested in showing that all possible behaviors of the system in question satisfy properties such as "the system will never reach a set of bad states" or "every occurrence of event a will be followed by an occurrence of event b," etc. These properties can be expressed in formalisms such as temporal logic [16]. Performance criteria used in control have similar yet somewhat different flavor, partly due to historical reasons, partly due to the different nature of the time domains and state spaces (in discrete spaces it is hard to talk about getting "closer" to a point). For example, convergence to an equilibrium is roughly the temporal property "eventually always P" where P is a small open ball around the equilibrium point. In this paper, we concentrate on the *safety* property, namely, the avoidance of a set of forbidden states ("avoid states where

[5]In fact, *differential automata* have been used to model hysteresis already in [14].

[6]By "set-valued" we mean nondeterminism, which specifies a set of possibilities but does *not* define probabilities on this set.

the altitude of the airplane is very low and its downward velocity is very high").

In order to speak about the behavior of a switched dynamical systems over time, we need a language to express both the evolution of the continuous variables as well as the evolution of the discrete state. A *temporal behavior* is the general concept that unifies them.

Definition 1 (Temporal Behavior): A temporal behavior over a set M is a partial function $\beta\colon T \to M$ whose domain of definition is an interval $[0, r)$ for some $r \in T \cup \{\infty\}$.

We call r the metric length of β, denote it by $|\beta|$, and say that β is infinite if $r = \infty$. Restrictions of β to points and intervals are denoted by $\beta[t]$ or $\beta[t_1, t_2)$.

Definition 2 (Piecewise-Constant Non-Zeno Behavior): A temporal behavior β of length r is piecewise-constant if it admits a strictly increasing sequence $\mathcal{J}(\beta) = 0, t_1, t_2, \cdots$ of time points such that for every k, β is constant on the interval $I_k = [t_k, t_{k+1})$. A behavior is non-Zeno if $\mathcal{J}(\beta) \cap [0, r]$ is finite for every $r < \infty$.

The definition of a piecewise-constant temporal behavior already excludes infinitely many switchings occurring at the *same* point in time (e.g., when a thermostat model has the same threshold for moving from ON to OFF and vice versa). The definition of a non-Zeno behavior excludes more sophisticated ways to "stop" the passage of time, by prohibiting infinitely many switchings from occurring in finite time, as do Achilles and the tortoise in the famous paradox attributed to Zeno of Elea.

We denote the sequence of intervals $I_0, I_1, \cdots$, by $\mathcal{I}(\beta)$. The *untimed abstraction* of a piecewise-constant behavior β is the partial function $\overline{\beta}\colon \mathbb{N} \to Q$ defined as $\overline{\beta}_k = q$ iff $\beta[t] = q$ for every $t \in I_k$. The number of intervals is called the *logical length* of β. Examples of continuous and piecewise-constant behaviors appear in Fig. 2.

Definition 3 (Hybrid Automaton): A hybrid automaton (HA) is a system $\mathcal{A} = (Q, X, H, G, f)$ where:

- $Q = \{1, \cdots, m\}$ is the discrete state space;
- X is the continuous state space, a bounded subset of $\mathbb{R}^n$;
- $H\colon Q \to 2^X$ are the staying conditions ("invariants");
- $G\colon Q \times Q \to 2^X$ are the switching conditions ("transition guards");
- $f\colon Q \to (X \to \mathbb{R}^n)$ assigns to every discrete state a continuous (and Lipschitz) vector field on X.

While the definition allows arbitrary subsets of the Euclidean space as staying and switching conditions, in practice we use much simpler subsets of X, *with limited topological complexity and some geometrical restrictions,* e.g., convex polyhedra. A hybrid automaton is depicted in Fig. 3. We will use the notation f_q for $f(q)$, $G_{qq'}$ for $G(q, q')$, H_q for $H(q)$ and H for $\bigcup_q H_q$. A pair $(q, \boldsymbol{x}) \in Q \times X$ is called a configuration of $\mathcal{A}$.

The set H_q is the subset of the continuous state space where the dynamics f_q can be applied. Such a restriction can come from physical modeling considerations (a plane cannot be in a flight mode in altitude zero) or from the decision of a particular controller (e.g., a good train controller will not

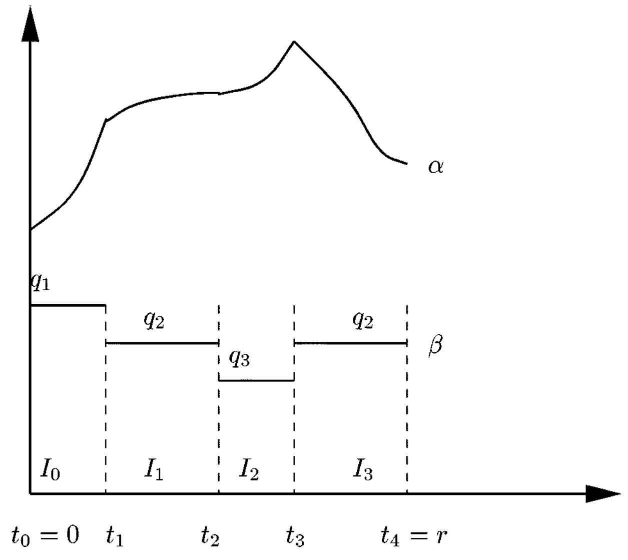


Fig. 2. A continuous behavior α and a piecewise-constant behavior β of logical length 4 with $\overline{\beta} = q_1, q_2, q_3, q_2$.

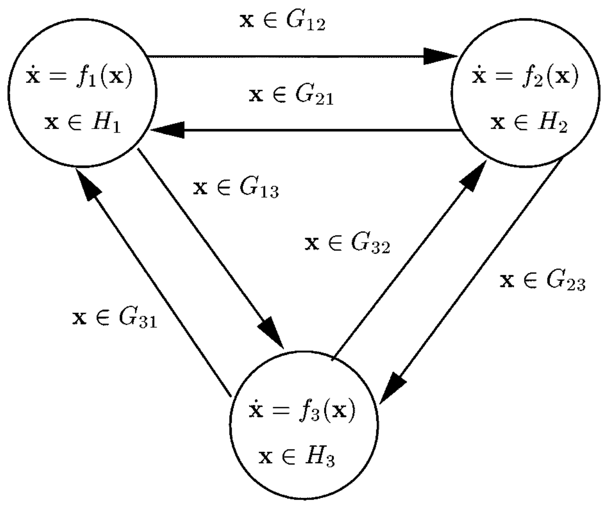


Fig. 3. A three-state hybrid automaton.

stay in an acceleration mode when the train is close to another train). Similarly, the set $G_{qq'}$ is the subset of the state space where the controller can switch from mode q to q'. These sets can be expressed in terms of the nondeterministic feedback map $c\colon Q \times X \to 2^Q$ where $H_q = \{\boldsymbol{x}\colon q \in c(q, \boldsymbol{x})\}$ and $G_{qq'} = \{\boldsymbol{x}\colon q' \in c(q, \boldsymbol{x})\}$.

The hybrid automata defined here are slightly more restricted than others appearing in this issue, e.g., [17] and [18], as there are no jumps or resets in the values of the continuous variables when transitions are taken. These more general models include also a reset map $R\colon Q \times Q \times X \to 2^X$, which decomposes into reset maps $R_{qq'}\colon X \to 2^X$ with $dom(R_{qq'}) = G_{qq'}$ for each $(q, q') \in Q \times Q$. Our models correspond to the special case where each $R_{qq'}$ is the identity map. For this reason, we may assume that $G_{qq} = \emptyset$ because a transition from a discrete state to itself without updating the continuous variables does not really change the configuration of the system. The results of this paper can be easily extended to systems with jumps at the price of additional notation.

The possible behaviors of such a hybrid automaton are those satisfying the following intuitive definition: when the HA is in a configuration $(q, \boldsymbol{x})$ such that $\boldsymbol{x} \in H_q$, the continuous state can evolve according to the differential equation $\dot{\boldsymbol{x}} = f_q(\boldsymbol{x})$, for which we assume existence and uniqueness of solution for every initial condition in H_q. Whenever a point $\boldsymbol{x}' \in G_{qq'}$ is reached, the automaton *can* make a transition to q' and switch the dynamics accordingly to $\dot{\boldsymbol{x}} = f_{q'}(\boldsymbol{x})$. We assume that transitions can be made only to states where continuous evolution can be continued, i.e., $G_{qq'} \subseteq H_{q'}$. Whenever this is not the case, G can be replaced by $G' = G \cap H$, i.e., $G'_{qq'} = G_{qq'} \cap H_{q'}$ for every $q, q' \in Q$. All this can be formalized as follows.

Definition 4 (Continuous Evolution): Let α: $T \to X$ be the solution of the differential equation $\dot{\boldsymbol{x}} = f_q(\boldsymbol{x})$ with $\alpha[0] = \boldsymbol{x}$, and let F be a subset of X. We say that:

- The dynamics q is enabled from $\boldsymbol{x}$ for time $t \geq 0$ if $\alpha[t'] \in H_q$ for every $t' < t$. This is denoted by $\boldsymbol{x} \xrightarrow{q,\, t}$. If, in addition, the trajectory stays within $F \subseteq X$ between 0 and t, we write it as $x \xrightarrow[F]{q,\, t}$.
- A point $\boldsymbol{x}'$ is q-reachable from $\boldsymbol{x}$ in time t if $\boldsymbol{x} \xrightarrow{q,\, t}$, $t < \infty$ and $\alpha[t] = \boldsymbol{x}'$. This is denoted by $\boldsymbol{x} \xrightarrow{q,\, t} \boldsymbol{x}'$. If, in addition, $\alpha[t'] \in F$ for every $t' < t$, we write it as $\boldsymbol{x} \xrightarrow[F]{q,\, t} \boldsymbol{x}'$.
- A set $G \subseteq X$ is q-reachable from $\boldsymbol{x}$ in time t if $\boldsymbol{x} \xrightarrow{q,\, t} \boldsymbol{x}'$ for some $\boldsymbol{x}' \in G$. This is denoted by $\boldsymbol{x} \xrightarrow{q,\, t} G$. If, in addition, $\alpha[t'] \in F$ for every $t' < t$, we write it as $\boldsymbol{x} \xrightarrow[F]{q,\, t} G$ (F until G in mode q).

Equipped with these definitions we can define the *semantics* of hybrid automata, i.e., the set of behaviors they can generate from any initial state.

Definition 5 (Semantics of HA): A trajectory of a hybrid automation $\mathcal{A}$, starting from a configuration $(q_0, \boldsymbol{x}_0)$ is a behavior ξ: $T \to (X \times Q)$, which can be written as a pair of behaviors α: $T \to X$ and β: $T \to Q$ such that α is continuous, β is piecewise-constant, and:

1) initiality: $\alpha[0] = \boldsymbol{x}_0$ and $\beta[0] = q_0$;
2) differential evolution: for every interval $I_k = [t_k, t_{k+1}) \in \mathcal{I}(\beta)$ such that $\overline{\beta}_k = q$, $\alpha[t_k] \xrightarrow{q,\, t} \alpha[t_k + t]$ for every $t \in [0, t_{k+1} - t_k)$;
3) transition conditions: for every $t_k \in \mathcal{J}(\beta)$, such that $\overline{\beta}_{k-1} = q$ and $\overline{\beta}_k = q'$, $\alpha[t_k] \in G_{qq'}$.

Note that our definitions of piecewise-constant behaviors imply strict monotonicity of the jump points and for any step k, the interval I_k has duration $t_{k+1} - t_k > 0$ and a nontrivial continuous evolution. Other models allow steps of zero duration and, hence, several jumps in one time point.

A behavior of a three-state HA is sketched in Fig. 4, which can be viewed as drawing the behavior in Fig. 2 on the state space. Recall that typically there is more than one possible trajectory starting from a given configuration $(q, \boldsymbol{x})$ because H_q and $G_{qq'}$ may overlap. Such "margins" can be useful for representing uncertainty both in physical modeling and in the delay between the initiation of a transition and its actual execution. Uniqueness of solutions is not part of the prerequisites of verification methodology. It can be restored by insisting on *deterministic* HA where for every q the sets $\{G_{qq'}\}_{q' \neq q}$ are mutually disjoint and have an empty intersection with the interior of H_q. The set of all trajectories starting from $(q_0, \boldsymbol{x}_0)$ is denoted by $L(\mathcal{A}, (q_0, \boldsymbol{x}_0))$ and the set of trajectories starting from any $(q, \boldsymbol{x})$ such that $\boldsymbol{x} \in H_q$ is denoted by $L(\mathcal{A})$.

Some anomalies may occur under these definitions. It may happen that a trajectory leaves H_q without entering any $G_{qq'}$. Such a trajectory becomes "blocked." One way to fix it is to "complete" the automaton by adding a new discrete state to which the systems enters when it goes out of the staying conditions of all dynamics. We will not do it because there will be an explicit notion of the "good" and "bad" states in the formulations of the synthesis problem.

The other anomaly is that of a Zeno behavior (a term first coined in [19]), namely, a trajectory that switches infinitely often between discrete states during a bounded time interval. Similar phenomena have been studied extensively under the title *sliding mode control* [6]. One should be very careful to prevent the synthesis algorithm from producing controllers that can avoid bad states only by generating such behaviors which "stop" Time. Our approach to this problem is to define HA that are non-Zeno by construction (as we did for timed automata in [20]) and, hence, any synthesized controller cannot generate Zeno behaviors.

Definition 6 (Non-Zeno Hybrid Automata): A Zeno cycle in a HA is a sequence $q_1, \cdots, q_s$ of states such that $cl(G_{12}) \cap cl(G_{23}) \cap \cdots \cap cl(G_{s1}) \neq \emptyset$ where cl is the closure operator. An HA is non-Zeno if it has no Zeno cycle.

Claim 1 (Trajectories of Non-Zeno Automata): All the behaviors of a non-Zeno HA are non-Zeno.

Proof: For any three states $q, q', q'' \in Q$ in a non-Zeno HA, the distance between the sets $G_{qq'}$ and $G_{q'q''}$ is bounded below by some $d > 0$, and, hence, there is a positive lower bound on the time duration of evolution within q'. Let Δ be the minimal such number over all triples of states. Then every trajectory ξ of logical length greater than or equal to m satisfies $|\xi| > m\Delta$. ■

The lack of Zeno cycles is a sufficient (but not necessary) condition for preventing Zeno behaviors, which is easy to check. Finding more precise conditions is an interesting topic that is irrelevant here.

Given two HA $\mathcal{A} = (Q, X, H, G, f)$ and $\mathcal{A}' = (Q, X, H', G', f)$, we say that $\mathcal{A}'$ is more restrictive than $\mathcal{A}$, denoted by $\mathcal{A}' \preceq \mathcal{A}$, if $H' \subseteq H$ and $G' \subseteq G$ in the natural sense of inclusion between such functions, i.e., $H'_q \subseteq H_q$ and $G'_{qq'} \subseteq G_{qq'}$ for every $q, q' \in Q$. Clearly $\mathcal{A}' \preceq \mathcal{A}$ implies $L(\mathcal{A}') \subseteq L(\mathcal{A})$, and if $\mathcal{A}$ is non-Zeno, so is $\mathcal{A}'$.

III. The Problem and the Solution

We formulate the simplest control problem of avoiding bad states in a non trivial way (a trivial solution would be to block completely the evolution of the system by letting $H = \emptyset$).

Definition 7 (Safety Synthesis for Hybrid Automata): Let $\mathcal{A} = (Q, X, H, G, f)$ be an HA, and let F be a subset of $Q \times X$. The safety controller synthesis problem is: find the maximal nonblocking HA $\mathcal{A}^* \preceq \mathcal{A}$ such that for every $\xi \in L(\mathcal{A}')$ and every $t \in T$, $\xi[t] \in F$.

In order to solve this problem, we make use of the following operator.

Definition 8 (Predecessors): The predecessor operator π: $2^{Q\times X} \rightarrow 2^{Q\times X}$ is defined for every set of configurations

$$F = (\{q_1\} \times F_1) \cup \cdots \cup (\{q_m\} \times F_m)$$

as

$$\pi(F) = \left\{(q, \boldsymbol{x}): \boldsymbol{x} \xrightarrow[F_q]{q,\,\infty} \vee \left(\exists t \in T\, \exists q' \in Q\, \exists \boldsymbol{x}' \in X \right.\right.$$
$$\left.\left. \boldsymbol{x} \xrightarrow[F_q]{q,\,t} \boldsymbol{x}' \wedge \boldsymbol{x}' \in G_{qq'} \wedge (q', \boldsymbol{x}') \in F\right)\right\}.$$

Essentially, $(q, \boldsymbol{x})$ is in $\pi(F)$ if either there is an infinite trajectory without switching starting from $(q, \boldsymbol{x})$ and always staying in F or it is possible to stay in F for some time and then make a transition to another configuration which is still in F. For those who do not feel comfortable with quantifiers, we give the following alternative definition of $\pi(F)$. Let A and B be two subsets of X. For every q, we define the following two operators.

- The unbounded time predecessor π_q^∞: $2^X \rightarrow 2^X$, defined as $\pi_q^\infty(A) = \{\boldsymbol{x}: \boldsymbol{x} \xrightarrow[A]{q,\,\infty}\}$, i.e., the points from which it is possible to continue indefinitely with dynamics q while staying in A.
- The until operator $\mathcal{U}_q$: $2^X \times 2^X \rightarrow 2^X$ defined as $\mathcal{U}_q(A, B) = \{\boldsymbol{x}: \exists t \boldsymbol{x} \xrightarrow[A]{q,\,t} B\}$, i.e., points from which it is possible to continue with dynamics q and stay in A until B is reached.

Then $\pi(F)$ can be written as

$$F' = (\{q_1\} \times F_1') \cup \cdots \cup (\{q_m\} \times F_m')$$

where for every q

$$F_q' = \pi_q^\infty(F_q) \cup \bigcup_{q' \neq q} \mathcal{U}_q(F_q, (G_{qq'} \cap F_{q'}))$$

as illustrated in Fig. 5.

The high-level algorithm for solving this problem is presented below. It works by computing the set P^* of "winning" states and can be viewed as a specialization of dynamic programming value iteration to 0–1 cost functions.

```
Algorithm 1
(Safety Controller Synthesis for HA)
$P^0 := F \cap H$
repeat
  $P^{k+1} := P^k \cap \pi(P^k)$
until $P^{k+1} = P^k$
$P^* := P^k$
```

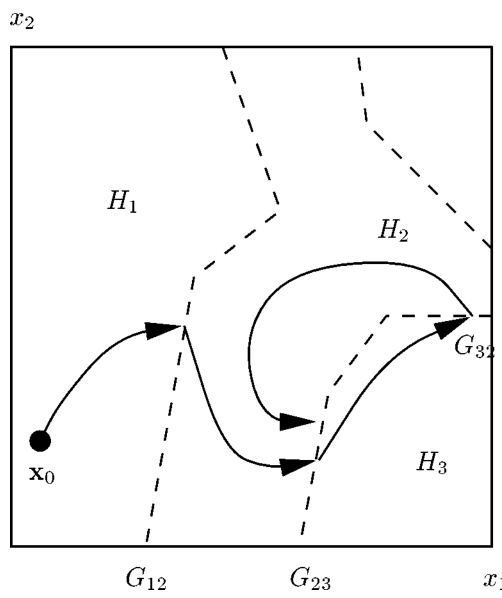


Fig. 4. A sketch of a behavior of the HA of Fig. 3 with $X = \mathbb{R}^2$.

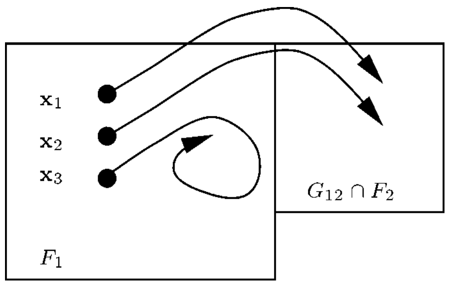


Fig. 5. Computation of F_1' from F_1 and F_2. Point $\boldsymbol{x}_3$ is in $\pi_1^\infty(F_1)$. Point $\boldsymbol{x}_2$ is in $\mathcal{U}_1(F_1, (G_{12} \cap F_2))$. Point $\boldsymbol{x}_1$ is in neither; hence, it is not in F_1'.

Claim 2 (Properties of the Algorithm): For every k, the state $(q, \boldsymbol{x})$ belongs to P^k iff $L(\mathcal{A}, (q, \boldsymbol{x}))$ contains a trajectory that remains invariantly in F, which is either of logical length smaller than k and infinite metric length, or else of logical length not less than k.

Proof: The proof concerning the length of trajectories is done by induction. For the base case, all points at P^0 admit empty runs of length zero and all points outside P^0 (and outside F) do not admit such runs. Going from k to $k+1$ is easy because if $(q, \boldsymbol{x}) \in \pi(P^k)$, it can take one transition to P^k and k transitions from there. On the other direction, if $(q, \boldsymbol{x}) \notin \pi(P^k)$, the automaton cannot take from it a transition to P^k nor an infinite run, and, hence, it can take at most k transitions. ∎

The algorithm produces a decreasing sequence $\{P^k\}$ of sets, and if the algorithm terminates, it returns the fixed point P^*.

Claim 3: The automaton $\mathcal{A}^* = (Q, X, H^*, G^*, f)$, where for every q, q' $H_q^* = \{\boldsymbol{x}: (q, \boldsymbol{x}) \in P^*\}$ and $G_{qq'}^* = G_{qq'} \cap H_q^* \cap H_{q'}^*$ is the solution of the safety controller synthesis problem.

Proof: The limit P^* is the set of all points that have either a run of finite logical length whose last interval is infinite or a run of infinite logical length, which implies (for non-Zeno HA) an infinite metric length. Hence, P^* is the maximal subset of $Q \times X$ from which all trajectories can be extended to infinity without leaving F. Any automaton larger than $\mathcal{A}^*$ will contain points outside P^*, which do not admit infinite trajectories inside F. ■

A feedback map $c: Q \times X \to 2^Q$ can be derived from $\mathcal{A}^*$ by letting

$$c(q, \boldsymbol{x}) = \{q': (q' = q \wedge \boldsymbol{x} \in H_q^*) \vee (q' \neq q \wedge \boldsymbol{x} \in {G^*}_{qq'})\}.$$

As mentioned above, this controller is not deterministic, similar to the "least restrictive supervisor" in the theory of discrete-event control [8]. A deterministic controller can be derived from it by reducing H^* and G^* so that the feedback map becomes a function $c: Q \times X \to Q$. In general, there is no "canonical" way to do this reduction, and we consider it an implementation issue.

Mathematicians might stop here: P^* is characterized as the maximal fixed point of the equation $P = F \cap \pi(P)$, and $\mathcal{A}^*$ is the solution of the synthesis problem. Those interested in actually *computing* the controller need to proceed further. While the notion of *effective computation*[7] is central in computer science, it is less familiar within control theory, so some introductory remarks are in order.

Algorithm 1 involves the computation of the following functions over *subsets of* X:

- intersection, $\cap: 2^X \times 2^X \to 2^X$;
- predecessors, $\pi: 2^X \to 2^X$;
- equivalence checking, $=: 2^X \times 2^X \to \{0, 1\}$.

The latter is needed to detect the termination of the algorithm when $P^{k+1} = P^k$ and can be reduced to checking emptiness of the set difference $P^k - P^{k+1}$.

In order for a function to be computable by a discrete device, the elements in its domain and its range must have a finite syntactic representation, and there must be an effective and terminating procedure, which takes as input a representation of an element of the domain and returns as output a representation of the value of the function applied to that element. For example, functions over the integers can be computed by applying well-known algorithms for addition and multiplication to unary, binary, or decimal representations of numbers. In this paper, we focus on the problem of computing functions over *subsets* of X. Subsets of the mathematical real numbers can be very weird objects and, unlike the finite sets encountered in discrete verification, they do not admit an explicit (enumerative) representation. Instead, they can be expressed symbolically by finite syntactical objects (e.g., formulas of some logic). To take a concrete example, the subclass of *polyhedral* sets consists of sets which can be represented by Boolean combinations of linear inequalities, and the membership of any point in a given set can be determined using a finite number of arithmetical and logical operations. Similarly, the *semi-algebraic* sets are those that can be written as combinations of polynomial inequalities. While these sets admit effective Boolean operations, easy membership testing, and more involved equivalence checking algorithms, they are usually not closed under flows of linear systems, and, hence, even if a set F belongs to a well-behaving class, $\pi(F)$ may not belong to that class.

We illustrate the problem using a HA with one discrete state and a simplified version of $\pi(F)$ defined as $\pi_0(F) = \{\boldsymbol{x}: \exists\, t \boldsymbol{x} \xrightarrow{t} F\}$, namely, the points from which it is possible to reach F some time in the future. Suppose that F itself is characterized by a formula $\varphi(\boldsymbol{x})$ whose truth value is 1 iff $\boldsymbol{x} \in F$. Suppose further that the equation $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ has a closed-form solution[8] of the form $\alpha(\boldsymbol{x}, t)$ for every initial condition $\boldsymbol{x}$. In this case, $\pi_0(F)$ can be characterized by the formula

$$\psi(\boldsymbol{x}) = \exists\, t \geq 0\, \varphi(\alpha(\boldsymbol{x}, t)).$$

The process of transforming ψ into an equivalent quantifier-free formula $\psi'(\boldsymbol{x})$ is called *quantifier elimination* by logicians. If φ and α are part of a theory that admits a quantifier elimination algorithm, then the π_0 operator is computable, i.e., there is an effective way to transform a formula $\varphi(\boldsymbol{x})$ for F into a formula $\psi'(\boldsymbol{x})$ for $\pi_0(F)$. If in the logic in question, the satisfiability problem for nonquantified formulas is decidable, questions such as the inclusion of $\pi_0(F)$ in another set are decidable as well (see [18] for a survey of logic notions).

In the simple case where f is constant, i.e., $f(\boldsymbol{x}) = \boldsymbol{c}$, and F is a polyhedral set written as a formula $\varphi(\boldsymbol{x})$, which is a combination of linear equalities, we get

$$\psi(\boldsymbol{x}) = \exists\, t \geq 0\, \varphi(\boldsymbol{x} + \boldsymbol{c}t)$$

and quantifier elimination can be performed using elementary linear algebra. This is the basis for reachability algorithms for timed automata and other hybrid systems with constant derivatives implemented in tools such as Kronos [21] and HyTech [22]. In the less trivial case of linear systems, the definition reduces to

$$\pi_0(F) = \{\boldsymbol{x}: \exists\, t\, e^{At}\boldsymbol{x} \in F\}.$$

In this case, quantifier-elimination techniques fail except for some special classes of matrices (recent results concerning the applicability of algebraic manipulation techniques for the reachability analysis of linear hybrid systems appear in [23], [24], and [17]). The geometry of $\pi_0(F)$ for constant and nonconstant dynamics is illustrated in Fig. 6(a) and (b).

Even if we were equipped with an effective precise procedure to compute $\pi(F)$, Algorithm 1, more often than not, will not terminate in a finite number of steps, even for the most trivial forms of continuous dynamics. In such cases, for every step k of the iteration, there will be some part of P^k that cannot stay within itself and the fixed point will not be reached in a finite number of iterations. This phenomenon is illustrated in Fig. 7 using a PCD system (a special class of deterministic HA where all vector fields are constant; see [25]).

[7]The theory of computability, also known as *recursion theory*, can be seen as a kind of pure mathematics of computer science, and it has deep connection with logic and set theory. The notions that we borrow from there are only the tip of the iceberg and the exposition is not meant by any means to be a serious introduction to the domain.

[8]An assumption which, by itself, restricts the scope of the approach significantly.

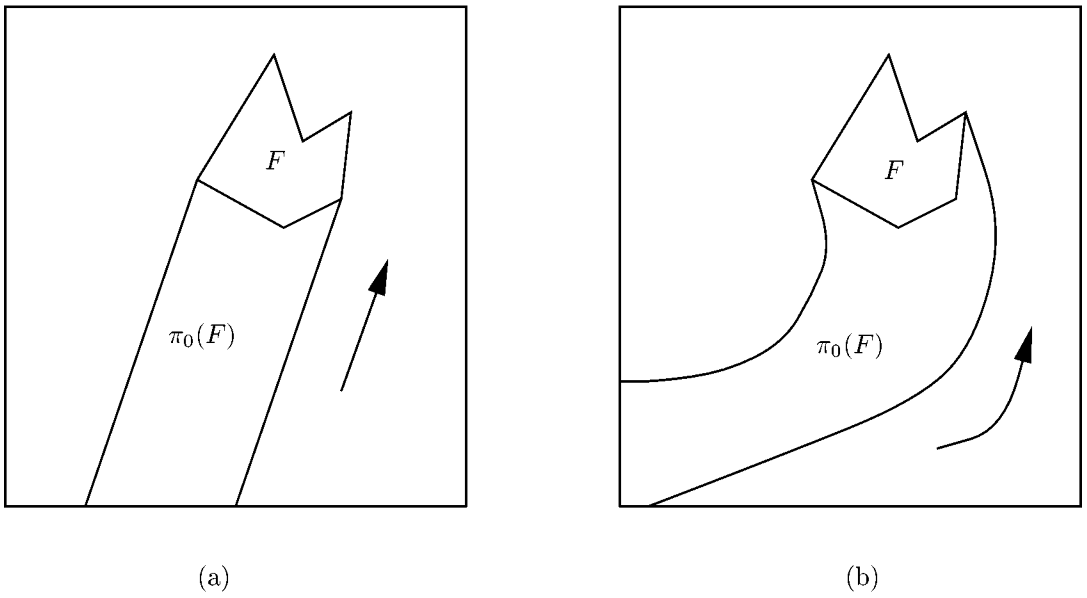


Fig. 6. (a) The predecessors of a polyhedral set F under constant derivative. (b) The predecessors of F under nonconstant derivative.

Given this state of affairs, we resort to a classical solution of continuous mathematicians and use numerical methods to compute approximately predecessors (and successors) in the continuous state space.

To overcome the effectiveness and termination problems we propose an approximate variant of Algorithm 1, specialized for piecewise-linear systems, i.e., $\dot{x} = A_q\boldsymbol{x}$ for every $q \in Q$ (similar systems have been introduced in [26] for discrete time), and which uses a restricted class of subsets of X, orthogonal "griddy" polyhedra (unions of unit boxes defined by a fixed grid [27]). The approximation technique, described in the next section (and in more detail in [13]), when plugged into Algorithm 1, yields a sequence $\tilde{P}^k$ of polyhedra, which converges after finitely many steps and which satisfies $\tilde{P}^k \subseteq P^k$ for every k. The obtained solution $\tilde{P}^*$ is included in P^*, and, hence, the resulting HA satisfies the same required properties of $\mathcal{A}^*$ except, of course, being maximal.

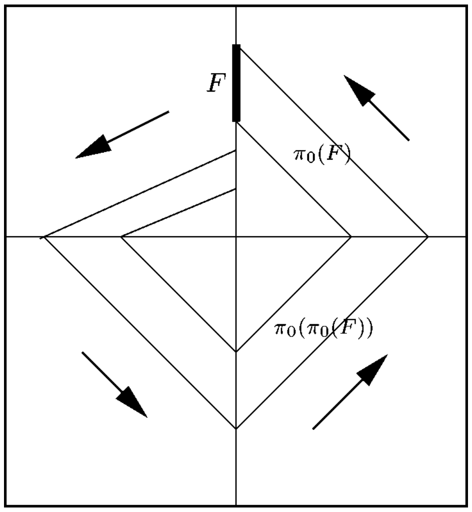


Fig. 7. A simple hybrid system with piecewise-constant derivatives for which the computation of $P*$ does not terminate.

IV. The Approximation Technique

The approximation technique developed in [13] treats various other problems related to the automatic analysis of hybrid systems. While the synthesis algorithm requires *under-approximation* of the *backward* reachability operator π, other tasks such as *verification* of reachability properties (computing all states reachable from a given set F of initial conditions under all admissible inputs) require *over-approximation* of the *forward* analysis operator δ, which is a bit more intuitive to explain.

Definition 9 (Successors): Let $\mathcal{A}$ be a dynamical system defined by $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and let $I \subseteq T$ be a time interval. The successor operator $\delta_I\colon 2^X \to 2^X$ is defined for any subset F of X as

$$\delta_I(F) = \{\boldsymbol{x}'\colon \exists\, \boldsymbol{x} \in F\ \exists\, t \in I\ \boldsymbol{x} \xrightarrow{t} \boldsymbol{x}'\}.$$

We use the notation δ_r for $\delta_{[r,\,r]}$ (states reachable after *exactly* r time), δ for $\delta_{[0,\,\infty)}$ (states reachable after any nonnegative amount of time), and $\delta_I(\boldsymbol{x})$ for $\delta_I(\{\boldsymbol{x}\})$. Note that δ satisfies the semigroup property, i.e., $\delta_{I_2}(\delta_{I_1}(F)) = \delta_{I_1 \ominus I_2}(F)$ where $I_1 \oplus I_2 = \{t_1 + t_2\colon t_1 \in I_1 \wedge t_2 \in I_2\}$ is the Minkowski sum of two intervals, and that, in particular, $\delta_{[0,\,r_2]}(\delta_{[0,\,r_1]}(F)) = \delta_{[0,\,r_1+r_2]}(F)$. Hence, the computation of $\delta(F)$, which is the basic operation in verification algorithms [as is $\pi(F)$ for algorithm 1], can be reduced to the following iterative numerical algorithm for some time step r.

```
Algorithm 2
(Exact Computation of $\delta(F)$ Using Time
step $r$):
$P^0 := F$
repeat
```

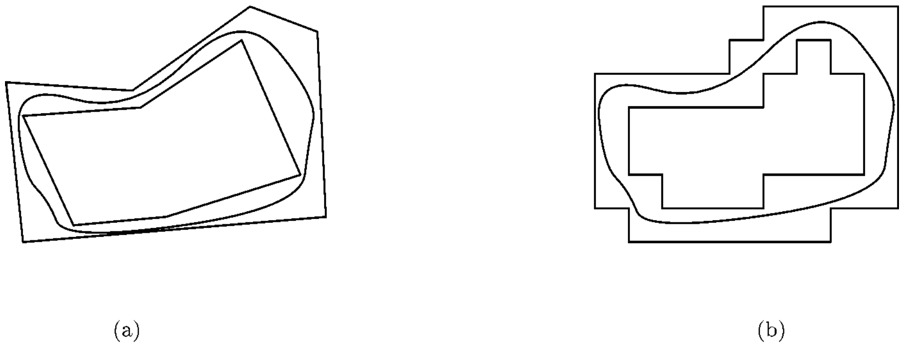


Fig. 8. (a) A set F over- and underapproximated by polyhedra. (b) The same set approximated by griddy polyhedra.

```
    $P^{k+1} := P^k \cup \delta_{[0,\,r]}(P^k)$
until $P^{k+1} = P^k$
```

The exact application of Algorithm 2 suffers from two problems. The computation of $\delta_{[0,\,r]}$ is not more feasible than the computation of the whole δ, and even if $\delta_{[0,\,r]}$ was computable, the algorithm usually does not terminate after a finite number of steps. The first problem can be resolved by approximating subsets of X by polyhedral sets. Any open or closed set $F \subseteq X$ can be over- or underapproximated arbitrarily closely by a set F' consisting of a finite union of convex polyhedra with rational vertices [see Fig. 8(a)]. An effective approximation of Algorithm 2 can thus be implemented by replacing all the operations (Boolean operations, equivalence testing, and computation of $\delta_{[0,\,r]}$) by their approximated versions. Note that if the class is closed under Boolean operations, only $\delta_{[0,\,r]}(F)$ needs to be approximated (this is true for arbitrary polyhedra but neither for convex polyhedra nor for ellipsoids). If the approximate algorithm terminates, the result is an overapproximation of $\delta(F)$.

The termination of the procedure, however, cannot be guaranteed since there are infinitely many polyhedral sets. Moreover, the implementation is very complicated because the sets P^k can be very complex nonconvex polyhedra for which there is no useful canonical form and the test $P^{k+1} = P^k$ is very expensive. To overcome this problem, we restrict further the class of sets which are used to approximate P^k to be what we call *griddy polyhedra*, i.e., sets that can be written as unions of closed unit hypercubes with integer vertices. When the continuous state space X is bounded, there are only finitely many griddy polyhedral subsets and Algorithm 2 is guaranteed to terminate. Moreover, the restriction to griddy polyhedra allows us to benefit from a relatively efficient canonical representation for both convex and nonconvex sets [27], supported by an experimental software package. The price, however, for using griddy polyhedra is that the quality of the approximation they provide in terms of Hausdorff distance per vertex is poorer than that of arbitrary polyhedra[9] but such a compromise seems unavoidable.

[9]For an m-vertex approximation of a figure with piecewise-smooth boundaries in $\mathbb{R}^n$, the worst case error is $O(m^{-1/(n-1)})$ for griddy and $O(m^{-2/(n-1)})$ for arbitrary approximating polyhedra.

Some aspects of the technique take advantage of special properties of linear systems. Let $conv(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\})$ be the convex hull of a set of points, i.e., $\{\boldsymbol{x}\colon \boldsymbol{x} = l_1\boldsymbol{x}_1 + \cdots, l_m\boldsymbol{x}_m\}$ for nonnegative l_i whose sum is 1. For linear systems, we have $\delta_t(\boldsymbol{x}) = e^{At}\boldsymbol{x}$ and the matrix exponential, as a linear operator, preserves convexity, i.e.,

$$\delta_t(conv(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\})) = conv(\{\delta_t(\boldsymbol{x}_1), \cdots, \delta_t(\boldsymbol{x}_m)\}).$$

This means that for a convex set $F = conv(\boldsymbol{V})$ where $\boldsymbol{V} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\}$, and for every t, the states reachable from F can be determined by the states reachable from $\boldsymbol{V}$ [see Fig. 9(a)]. We exploit this property to approximate $\delta_{[0,\,r]}(conv(\boldsymbol{V}))$ based on the set of points $\boldsymbol{V} \cup \delta_r(\boldsymbol{V})$ where $\delta_r(\boldsymbol{V})$ is computed from $\boldsymbol{V}$ by a finite number of matrix exponentiations or numerical integration steps. Our approximation scheme consists of three steps.

1. Compute $G = conv(\boldsymbol{V} \cup \delta_r(\boldsymbol{V}))$ [see Fig. 9(b)]. This set is an approximation of $\delta_{[0,\,r]}(conv(\boldsymbol{V}))$ but neither an overapproximation nor an underapproximation. The convex-hull algorithm provides us with information concerning the orientation of the faces, which is used in the next step.
2. Push the faces of G outward to obtain a bloated convex polyhedron G' that is guaranteed to contain the required set [Fig. 9(c)]. The amount of pushing is determined by the time step r and the matrix A (see the analysis in [13]). Pushing inward will result in an underapproximation.
3. Overapproximate G' by a griddy polyhedron $\delta'_{[0,\,r]}(F)$ [Fig. 9(d)].

The approximate algorithm for computing $\delta(F)$ for $F = conv(\boldsymbol{V})$ is defined below.

```
Algorithm 3
(Approximate Computation of $\delta(F)$ for
Linear Systems)
$P^0\colon = F$; $\boldsymbol{V}^0 := \boldsymbol{V}$;
repeat $k = 1, 2, \cdots$
    $\boldsymbol{V}^k := \delta_r(\boldsymbol{V}^{k-1})$;
    $G^k := conv(\boldsymbol{V}^{k-1} \cup \boldsymbol{V}^k)$;
    $G^k := bloat(G^k)$;
    $G^k := griddy(G^k)$;
    $P^k := P^{k-1} \cup G^k$
until $P^{k+1} = P^k$
```

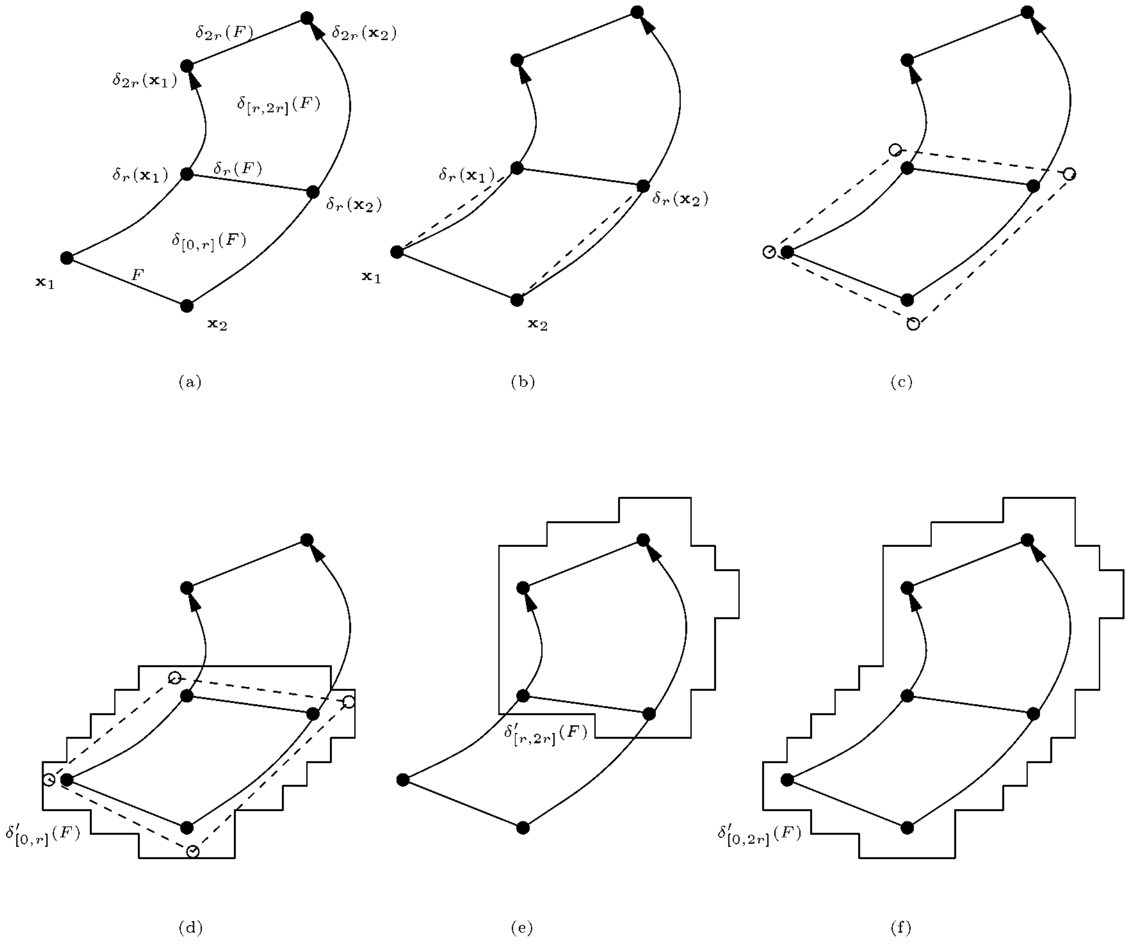


Fig. 9. (a) A set $F = conv(\{\boldsymbol{x}_1, \boldsymbol{x}_2\})$ and its exact successors for time intervals $[0, r]$ and $[r, 2r]$. (b) Approximating $\delta_{[0,r]}(F)$ by convex hull. (c) Bloating the convex polyhedron to obtain a polyhedral overapproximation. (d) Rectangulating the polyhedron into $\delta'_{[0,r]}(F)$. (e) Repeating the same procedure in the next time step to obtain $\delta'_{[r,2r]}(F)$. (f) The accumulated states $\delta'_{[0,2r]}(F) = \delta'_{[0,r]}(F) \cup \delta'_{[r,2r]}(F)$.

The algorithm is guaranteed to terminate because $\{P^k\}$ is a monotone increasing sequence over the finite set of griddy polyhedra. There are two types of errors accumulated in the process of computing P^k: from the actual set to its bloated convex hull and from there to the griddy polyhedron. However, these errors do not propagate to the next step, which computes P^{k+1} based on $\boldsymbol{V}^k \cup \boldsymbol{V}^{k+1}$ and not on P^k [Fig. 9(e)]. Note that our orthogonal polyhedra package [27] maintains $\delta'_{[0,2r]}(F)$ as a *single* canonical object and *not* as a union of convex polyhedra or ellipsoids [Fig. 9(f)]. The algorithm can be fine-tuned by changing the time step r and the size of the hypercubes.

Result 1 (Computation of Reachable States for Linear Systems): There exists an implemented algorithm for over-approximating the reachable sets of systems defined by linear differential equations.

This result is not a theorem due to the following facts.

1. There is always a trivial overapproximation of any subset F of X: X itself.
2. The smallest polyhedral or griddy set that contains $\delta(F)$ is as impossible to compute as $\delta(F)$.
3. The best upper bounds that can be easily proved on the approximation error are much larger than what happens in practice.

Doing underapproximation is almost symmetric, and for computing backwards, one needs to invert the system. With some additional modification, one can underapproximate the π operator and, hence, Result 2 follows.

Result 2 (Effective Controller Synthesis for Linear Systems): There exists an implemented algorithm for underapproximating the least restrictive safety controller for piecewise-linear systems.

In the framework described so far, we assumed no adversary (disturbances) and that all the transitions are controllable, i.e., initiated by the discrete controller. These assumptions can be relaxed but a detailed description is beyond the scope of this paper, so only a sketch is given below. Uncontrolled transitions, i.e., transitions initiated by the plant, can serve several modeling purposes. They can model a discrete transition of a physical system (such as a collision), an adversary that is modeled using a switching controller (a human operator that pushes a button) or, when a nonlinear system is approximated by a piecewise-linear one, a passage from one region of the state space to another. There is a standard adaptation of the π operator for such situations (see, e.g., [28]). For continuous disturbances, we have extended our system to treat dynamics of the form $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$ where $\boldsymbol{u}$ ranges over a convex set. Using a modification of the procedure proposed in [29], based on the maximum principle, we can compute an appropriate variant of π and solve the synthesis problem in the presence of such disturbances, as Example 1 will show. Our techniques can be adapted, via discretization of control values, to construct strategies for linear differential games of the form $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u} + C\boldsymbol{v}$, and it remains to see whether this approach has advantages over known techniques.

V. Examples

We illustrate the behavior of the algorithm on several examples. Recall that the results are obtained in a fully automatic manner once the model has been written.

Example 1 (Thermostat with Delay and Disturbances): Consider a thermostat having two states q_1 (OFF) and q_2 (ON). The corresponding dynamics are

$$\dot{x}_1 = -x_1 + u \qquad u \in [-0.5,\ 0.5]$$

and

$$\dot{x}_1 = -x_1 + u \qquad u \in [3.5,\ 4.5].$$

Note that these are equivalent to differential inclusions due to uncontrolled (but bounded) disturbances. Similarly to [30], we augment the system with an additional "clock" variable x_2, such that in every state $\dot{x}_2 = 1$, every transition is guarded by the condition $x_2 \geq 0.5$ and x_2 is reset to zero after a transition is taken. This construction guarantees that the guards are separated and the automaton is non-Zeno. The price is having an additional dimension and a slight modification of π to take clock resetting into account.

Our goal is to keep x_1 within the interval $[1.5, 3.6]$; hence, $F = [1.5,\ 3.6] \times [0.0,\ 0.5]$. The synthesis is done in the two-dimensional space and converges after three iterations into the safe sets shown in Fig. 10. After removing the fictitious clock variable by intersection with $x_2 = 0$, we obtain $H_1^* = [2.48365,\ 3.5]$ and $H_2^* = [1.5,\ 3.15736]$. From this we can derive a deterministic switching controller that starts heating when $x_1 = \theta_{12}$ and stops heating when $x_2 = \theta_{21}$ for any θ_{12} and θ_{21} satisfying $2.48365 < \theta_{12} < \theta_{21} < 3.15736$.

Example 2 (Two Spirals): Consider a system with two discrete states q_1 and q_2 where the dynamics is defined by

$$A_1 = \begin{pmatrix} 0.05 & -0.5 \\ 2.0 & 0.05 \end{pmatrix} \quad A_2 = \begin{pmatrix} 0.05 & -2.0 \\ 0.5 & 0.05 \end{pmatrix}$$

so that in each state there is an expanding spiral (see a sketch of the phase-portrait on top of Fig. 11). In order to stay within $F = \{q_1,\ q_2\} \times [-0.65,\ 0.35] \times [-0.35,\ 0.68]$, the controller must switch between the orthogonal spirals. The initial transition guards are

$$G_{12} = [-0.2,\ -0.01] \times [-0.2,\ 0.01]$$
$$G_{21} = [0.01,\ 0.32] \times [-0.01,\ 0.1].$$

The algorithm starts with F as the safe set and terminates after three iterations (Fig. 11).

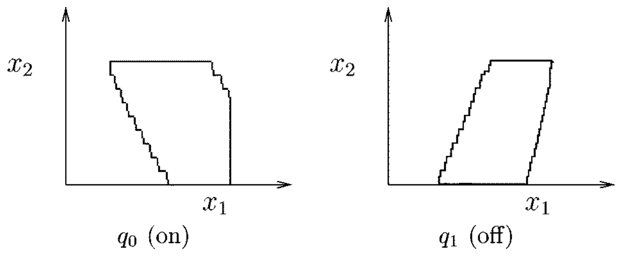


Fig. 10. The safe sets for the thermostat in the x_1–x_2 plane. The safe sets for x_1 are those obtained by intersection with $x_2 = 0$.

VI. Related Work

In this section, we discuss the relationship between our work and some other results on controller synthesis for hybrid systems. This is by no means an exhaustive survey; the reader is also referred to [31], [9], and [18] in this special issue.

Albeit under different names, many verification and controller synthesis problems can be viewed as instances of the more general concept of finding or evaluating strategies in games. For example, the typical question of verification, "Will the system behave correctly in the face of all behaviors of the environment?" can be rephrased as asking whether a particular given strategy is winning. Similarly, many problems in control can be viewed as simple instances of finding winning strategies in differential games [32]. For finite-state discrete systems, all variants of these problems are algorithmically solvable, and we survey various attempts to extend such results to deal with continuous and hybrid dynamics. These works can be classified according to the following interdependent criteria.

1) What is the complexity of the discrete and continuous dynamics considered? Different degrees of continuous complexity exist, starting from clocks, via constant derivatives, linear differential equations up to arbitrary continuous dynamics. Of course, the more complex the dynamics, the weaker the computational content of the results.
2) Direct versus indirect approach: does the synthesis procedure work directly on the continuous state space or is the system reduced first, via abstraction, into a finite-state automaton?
3) Computational content and generality: does the approach attempt to solve the problem for a *class* of systems or does it focus on a particular system originating from an application? Is the solution really effective or is there an implicit notion of an "oracle" that solves the hard computational problems?

The first and simplest class of hybrid systems for which controller synthesis has been applied is the class of timed automata [33] where all continuous variables are clocks following the same dynamics, $\dot{\boldsymbol{x}} = 1$, in all discrete states. Wong-Toi and Hoffmann [34] were the first to consider the application of supervisory control methodology to this class. In [35] (see also [20] and [28]), we have defined an algorithm similar to Algorithm 1 and have shown that the *exact* computation of P^* can be performed over the set of zones

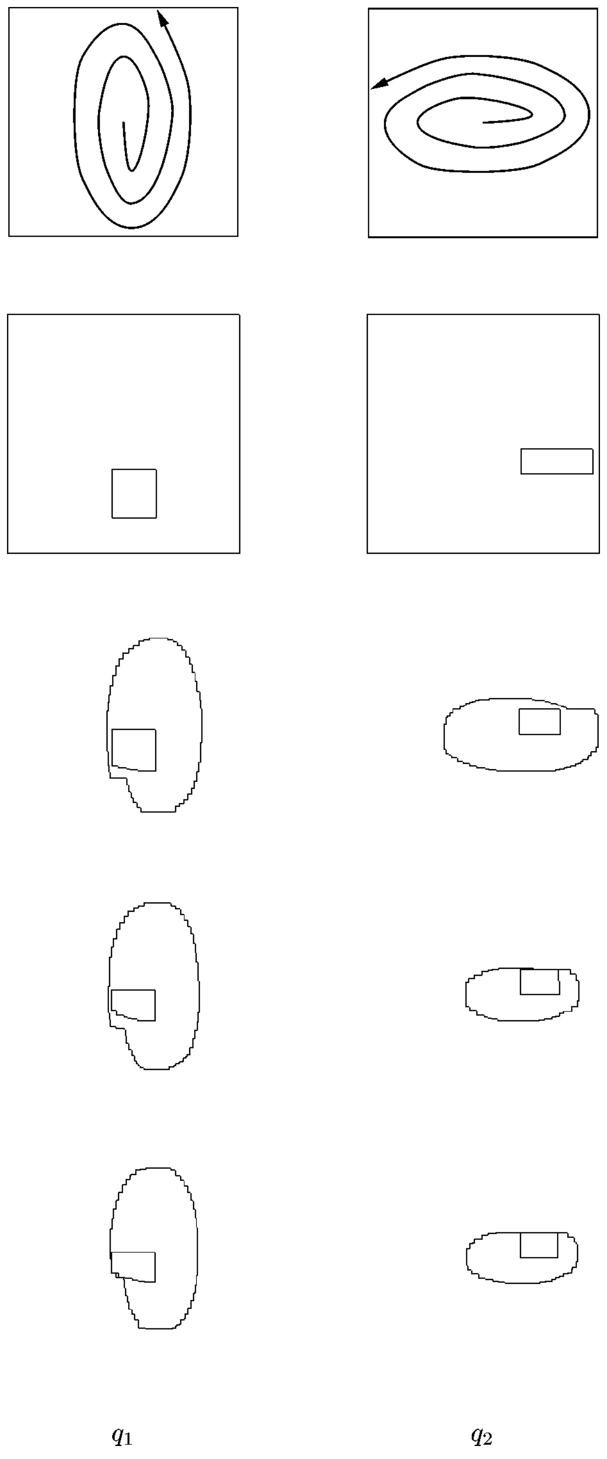


Fig. 11. The phase portrait of the two spiral system and the results of the algorithm. The evolution of F_1 and $G_{12} \cap F_1$ is shown on the left and that of F_2 and $G_{21} \cap F_2$ on the right. The final results show for each state the safe region where the system can spiral and then make a transition to the safe region of the other state.

(a restricted class of polyhedra underlying the verification of timed automata). Another class of systems for which a controller synthesis procedure always terminates are the initialized *rectangular* hybrid automata, systems where the dynamics in each state is of the form $c \leq \dot{x}_i \leq d$ for every variable x_i and for which controller synthesis was proved to be solvable for discrete [36] and continuous [37] time by Henzinger *et al.* In the rest of the constant slope world, exact verification and synthesis problems are undecidable, algorithms such as Algorithm 1 are not guaranteed to terminate, yet operators such as π and δ can be computed exactly at every step using linear algebra (polyhedra are closed under these operations). A controller synthesis procedure for fixed slope hybrid automata, which also treats the problem of Zeno controllers, was given by Wong–Toi in [38] and implemented in HyTech. Earlier work concerning controller synthesis for such systems was reported in [39].

Moving to nontrivial continuous dynamics, one faces the problems mentioned in this paper, and the class of systems for which results on synthesis can be given an exact computational content is extremely limited. The works of Lygeros *et al.* using a game-theoretic approach [40], [31] are very close in spirit to ours. They attempt to solve the controller synthesis problem using an abstract algorithm similar to Algorithm 1, for arbitrary continuous dynamics with time-varying piecewise-continuous control and disturbance inputs. The computational burden of computing the π operator is delegated, in the spirit of differential games, to the solution of a Hamilton–Jacobi–Bellman–Isaacs partial differential equation. In fact, all problems of tracking the evolution of a subset of X under differential flows can be rephrased as an initial-value problem for PDE [41], but no evidence has been given so far of the computational advantages of this point of view. In [24], it has been shown that for the subclass of linear systems where the matrices are either diagonal or nilpotent, the synthesis problem is solvable, in principle, using computer algebra. Beyond this limit, we believe there is not much hope for exact answers.

Another distinguishing feature of computational approaches to these problems is whether they work *directly* on subsets of the continuous state space (as in this paper) or *indirectly* by using a finite-state abstraction of the original system. Without getting into technical details (see [17] for exact definitions), this approach consists in finding a dynamics-preserving homomorphism (such as the one known as *bisimulations*) from the continuous or hybrid system into a finite-state automaton. These homomorphisms result in a finite partition of the state space such that the continuous reachability between partition blocks is faithfully reflected in the finite-state automaton transition relation.

The indirect approach is very tempting to use because once the finite abstraction has been constructed, we can apply standard discrete algorithms, some of which are already implemented in tools. The danger of this approach is that it may lead to sweeping the hard parts under the carpet and proving results that assume already the existence of a well-behaving partition or prove the *existence* of such a partition for certain systems, without explaining how to actually compute it or use it in practice (such partitions might have very complex boundaries, whose crossing cannot be detected by any realistic controller).

The work of [34] is an example of the rigorous application of the indirect approach to timed automata by using the finite quotient, also known as the "region graph," on which a supervisory control problem a la Ramadge–Wonham is solved.

Our work in [35] solves the same problem in a direct manner. In verification of systems with constant slopes, the indirect approach is represented by papers like [42] and [17], which prove that for certain classes of systems such reductions are possible. The verification procedure in [12] and in [25] (the latter proves decidability for a class of systems *not* having a finite quotient) and the synthesis algorithm of [38] are examples of the direct approach. More complex continuous dynamics were treated by [43] and [44], which tried to compute approximate discrete abstractions for electrical circuits modeled at the transistor level.

We believe that in practice, except for timed automata, very few interesting hybrid systems admit a finite quotient, and that in any case, trying to compute or approximate such a partition is at least as hard as solving any verification problem. For example, in order to show that a given partition is a bisimulation, one needs to compute successors for every block of the partition. Moreover, the size of the quotient might be prohibitively large for discrete verification and synthesis. A direct algorithm, which explores the state space "on-the-fly," might, in the worst case, do all this exploration, but in many practical cases the exploration will terminate without traversing the whole state space. Nevertheless, proving the existence of a finite bisimulation for a class of systems is an important step in tackling its verification and synthesis problems. For example, the termination of Algorithm 1 for timed automata is implied by the existence of the region graph [33]. If the system is to be subject to many different queries, it might be worthwhile to compute its finite-state abstraction as a preprocessing step.

Other works based on a mixture of direct and indirect approaches have been proposed by various authors from the control and DES communities (see the tutorial [45] and the survey in [9]). Some of these works try to relate continuous and hybrid models to the supervisory control framework. Since the dynamics treated by these authors is nontrivial, they do not look for exact finite quotient but rather for *approximating* automata (e.g., [46]). In more recent works [47], [48], backward methods similar to ours are used for computing and refining finite partitions of such systems. In [9], the problem of *interface design*, i.e., defining a mapping from the continuous state space to a finite observation alphabet that can serve as a basis for feedback control, has been investigated. In fact, modern DES methods for hybrid systems can be viewed as combining preliminary direct analysis with indirect synthesis of the supervisory controller.

Our approximation scheme is among a class of new techniques for computing reachable states of continuous systems, e.g., [44], [49]–[51], [29], [47], [52], [30], [53], and [54]. Among these, the work of Chutinan and Krogh [47], also centered on linear differential equations, is the closest to ours. Their goal is to find discrete abstractions, and they use the approximate "flow pipes" as means to achieve this goal. Their approach differs from ours in some technical aspects, most notably their different way of computing δ and our use of griddy polyhedra for storing the reachable states.

One cultural difference between the different communities is manifested in the computer science tendency toward generality: a controller synthesis algorithm works (if complexity is ignored) for *all* finite-state automata or for *all* timed automata. On the other hand, in the study of continuous systems, it is sometimes hard enough to treat one instance of a problem. For example, the work of Zhao *et al.* in [55] combines knowledge of dynamics with computation-geometrical algorithms in order to synthesize controllers that navigate in the phase space of one particular nonlinear system. In our work, we have tried to follow the more general (and, perhaps, naive) approach, offering an approximate solution for the whole class of piecewise-linear dynamical systems. To be fair, our tool is not yet as general-purpose as we would like and some user intelligence is required in order to tune the system parameters and adapt them to each problem instance. We hope that further experience in applying this technique to real-life case studies will inspire more development and will determine whether it has a place among the useful techniques for computer-aided control system design.

Acknowledgment

The authors would like to thank an anonymous referee for a thorough reading of the manuscript and for comments which improved significantly the quality of the presentation. Additional useful comments were contributed by J. H. van Schuppen and B. Krogh.

References

[1] R. E. Kalman, P. L. Falb, and M. A. Arbib, *Topics in Mathematical System Theory*. New York: McGraw-Hill, 1968.
[2] E. D. Sontag, *Mathematical Control Theory—Deterministic Finite Dimensional Systems*. Berlin, Germany: Springer, 1990.
[3] M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra*. New York: Academic, 1974.
[4] K. J. Astrom and B. Wittenmark, *Computer Controlled Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[5] E. D. Sontag, "Interconnected automata and linear systems: A theoretical framework in discrete-time," in *Hybrid Systems III: Verification and Control*, T. A. Henzinger, R. Alur, and D. Sontag, Eds. Berlin, Germany: Springer-Verlag, 1996, no. 1066, Lecture Notes in Computer Science, pp. 436–448.
[6] A. F. Filippov, *Differential Equations with Discontinuous Right-Hand Sides*. Norwell, MA: Kluwer, 1988.
[7] D. Liberzon and A. S. Morse, "Basic problems in stability and design of switched systems," *IEEE Contr. Syst. Mag.*, vol. 19, 1999.
[8] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, 1989.
[9] X. D. Koutsoukos, P. J. Antsaklis, M. D. Lemmon, and J. A. Stiver, "Supervisory control of hybrid systems," *Proc. IEEE*, vol. 88, pp. 1026–1049.
[10] O. Maler, "A unified approach for studying discrete and continuous dynamical systems," in *Proc. CDC'98*, 1998.
[11] O. Maler, Z. Manna, and A. Pnueli, "From timed to hybrid systems," in *Real-Time: Theory in Practice*, W. P. de Roever, J. W. de Bakker, C. Huizing, and G. Rozenberg, Eds. Berlin, Germany: Springer-Verlag, 1992, no. 600, Lecture Notes in Computer Science, pp. 447–484.
[12] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoret. Comput. Sci.*, vol. 138, pp. 3–34, 1995.
[13] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science, pp. 20–31.

[14] L. Tavernini, “Differential automata and their simulators,” *Non-Linear Anal., Theory, Methods and Applicat.*, vol. 11, pp. 665–683, 1987.

[15] J. P. Aubin and A. Cellina, *Differential Inclusions: Set-Valued Maps and Viability Theory*. Berlin, Germany: Springer, 1984.

[16] Z. Manna and A. Pnueli, *Temporal Verification of Reactive Systems: Safety*. Berlin, Germany: Springer, 1995.

[17] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, “Discrete abstractions of hybrid systems,” *Proc. IEEE*, vol. 88, pp. 971–984.

[18] J. M. Davoren and A. Nerode, “Logics for hybrid systems,” *Proc. IEEE*, vol. 88, pp. 985–1010.

[19] M. Abadi and L. Lamport, “An old-fashioned recipe for real time,” in *Real-Time: Theory in Practice*, W. P. de Roever, J. W. de Bakker, C. Huizing, and G. Rozenberg, Eds. Berlin, Germany: Springer-Verlag, 1992, no. 600, Lecture Notes in Computer Science, pp. 1–27.

[20] E. Asarin, O. Maler, and A. Pnueli, “Symbolic controller synthesis for discrete and timed systems,” in *Hybrid Systems II*, P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1995, no. 999, Lecture Notes in Computer Science, pp. 1–20.

[21] S. Yovine, “Kronos: A verification tool for real-time systems,” *Software Tools Technol. Transfer*, vol. 1, pp. 123–133, 1987.

[22] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, “Hytech: A model checker for hybrid systems,” *Software Tools Technol. Transfer*, vol. 1, pp. 110–122, 1997.

[23] G. Pappas, G. Lafferriere, and S. Yovine, “A new class of decidable hybrid systems,” in *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1790, Lecture Notes in Computer Science, pp. 29–31.

[24] O. Shakernia, G. J. Pappas, and S. Sastry, “Decidable controller synthesis for classes of linear systems,” in *Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science, pp. 407–420.

[25] E. Asarin, O. Maler, and A. Pnueli, “Reachability analysis of dynamical systems having piecewise-constant derivatives,” *Theoret. Comput. Sci.*, vol. 138, pp. 35–66, 1995.

[26] E. D. Sontag, “Nonlinear regulation: The piecewise linear approach,” *IEEE Trans. Automat. Contr.*, vol. 26, 1981.

[27] O. Bournez, O. Maler, and A. Pnueli, “Orthogonal polyhedra: Representation and computation,” in *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1569, Lecture Notes in Computer Science, pp. 46–60.

[28] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis, “Controller synthesis for timed automata,” in *Proc. IFAC Symp. System Structure and Control*, 1998, pp. 469–474.

[29] P. Varaiya, “Reach set computation using optimal control,” in *Proc. KIT Workshop*, Verimag, Grenoble, 1998, pp. 377–383.

[30] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, “Beyond hytech: Hybrid system analysis using interval numerical methods,” in *Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science, pp. 130–144.

[31] C. J. Tomlin, J. Lygeros, and S. S. Sastry, “A game-theoretic approach to controller design for hybrid systems,” *Proc. IEEE*, vol. 88, pp. 949–970.

[32] R. Isaacs, *Differential Games: A Mathematical Theory With Applications to Warfare and Pursuit, Control and Optimization*. New York: Wiley, 1965.

[33] R. Alur and D. L. Dill, “A theory of timed automata,” *Theoret. Comput. Sci.*, vol. 126, pp. 183–235, 1994.

[34] H. Wong-Toi and G. Hoffmann, “The control of dense real-time discrete event systems,” Stanford University, Tech. Rep. STAN-CS-92-1411, 1992.

[35] O. Maler, A. Pnueli, and J. Sifakis, “On the synthesis of discrete controllers for timed systems,” in *Proc. STACS’95*, E. W. Mayr and C. Puech, Eds. Berlin, Germany: Springer-Verlag, 1995, no. 900, Lecture Notes in Computer Science, pp. 229–242.

[36] T. A. Henzinger and P. W. Kopke, “Discrete-time control for rectangular hybrid automata,” *Theoret. Comput. Sci.*, vol. 221, pp. 369–392, 1999.

[37] T. A. Henzinger, B. Horowitz, and R. Majumdar, “Rectangular hybrid games,” in *Proc. CONCUR’99*, J. C. M. Baeten and S. Mauw, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1664, Lecture Notes in Computer Science, pp. 320–335.

[38] H. Wong-Toi, “The synthesis of controllers for linear hybrid automata,” in *Proc. CDC’97*, 1997.

[39] M. Tittus and B. Egardt, “Controllability and control-law synthesis of linear hybrid systems,” in *Proc. Int. Conf. on Analysis and Optimization of Systems*, G. Cohen and J.-P. Quadrat, Eds. Berlin, Germany: Springer-Verlag, 1994, no. 199, Lecture Notes in Computer Science, pp. 377–383.

[40] C. Tomlin, J. Lygeros, and S. Sastry, “Controllers for reachability specifications for hybrid systems,” *Automatica*, vol. 35, 1999.

[41] P. Caspi, “Global simulation via partial differential equations,” Verimag, unpublished note, 1993.

[42] T. A. Henzinger, “Hybrid automata with finite bisimulations,” in *Proc. ICALP’95*, Z. F. Fülöp and F. Gécseg, Eds. Berlin, Germany: Springer-Verlag, 1995, no. 944, Lecture Notes in Computer Science, pp. 324–335.

[43] R. P. Kurshan and K. L. McMillan, “Analysis of digital circuits through symbolic reduction,” *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1350–1371, 1991.

[44] M. R. Greenstreet, “Verifying safety properties of differential equations,” in *Proc. CAV’96*, R. Alur and T. A. Henzinger, Eds. Berlin, Germany: Springer-Verlag, 1996, no. 1102, Lecture Notes in Computer Science, pp. 277–287.

[45] M. D. Lemmon, K. X. He, and I. Markovsky, “Supervisory hybrid system,” *IEEE Contr. Syst. Mag.*, vol. 19, 1999.

[46] J. E. R. Cury, B. H. Krogh, and T. Niinomi, “Supervisory controllers for hybrid systems based on approximating automata,” *IEEE Trans. Automat. Contr.*, vol. 43, pp. 564–568, 1998.

[47] A. Chutinan and B. H. Krogh, “Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations,” in *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1569, Lecture Notes in Computer Science, pp. 76–90.

[48] T. Moor and J. Raisch, “Discrete control of switched linear systems,” in *Proc. ECC’99*, 1999.

[49] M. R. Greenstreet and I. Mitchell, “Reachability analysis using polygonal projections,” in *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1569, Lecture Notes in Computer Science, pp. 76–90.

[50] T. Dang and O. Maler, “Reachability analysis via face lifting,” in *Hybrid Systems: Computation and Control*, T. A. Henzinger and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1998, no. 1386, Lecture Notes in Computer Science, pp. 96–109.

[51] A. Kurzhanski and I. Valyi, *Ellipsoidal Calculus for Estimation and Control*. Boston, MA: Birkhauser, 1997.

[52] J. Preussig, O. Stursberg, and S. Kowalewski, “Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis,” in *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1569, Lecture Notes in Computer Science, pp. 210–222.

[53] O. Botchkarev and S. Tripakis, “Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations,” in *Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science, pp. 73–88.

[54] A. Bemporad, F. D. Torrisi, and M. Morari, “Optimization-based verification and stability characterization of piecewise affine and hybrid systems,” in *Hybrid Systems: Computation and Control*, B. Krogh and N. Lynch, Eds. Berlin, Germany: Springer-Verlag, 2000, no. 1790, Lecture Notes in Computer Science, pp. 45–58.

[55] F. Zhao, S. C. Loh, and J. A. May, “Phase-space nonlinear control toolbox: The maglev experience,” in *Hybrid Systems V*, P. J. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, Eds. Berlin, Germany: Springer-Verlag, 1999, no. 1567, Lecture Notes in Computer Science, pp. 429–444.

Eugene Asarin received the degree in mathematics from Moscow State University, Moscow, Russia, in 1984, and the Ph.D. degree in system analysis from the Institute for Control Science, Moscow, in 1988.

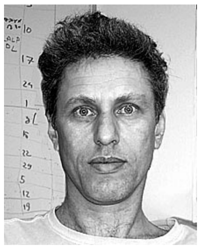
He was a Senior Researcher at the same institute and at the Institute for Information Transmission Problems, Moscow. Since 1999, he has been a Professor of Computer Science at the University of Grenoble, France, and a Member of VERIMAG laboratory, Gieres, France. His research interests include the theory of hybrid and timed systems.

Olivier Bournez received the "Magistère" degree in computer science and the Ph.D. degree in computer science from Ecole Normale Superieure of Lyon, France, in 1995 and 1999, respectively.

Since 1999, he has been an INRIA Researcher at the LORIA laboratory, Nancy, France. His research interests include modeling and verification of hybrid systems.

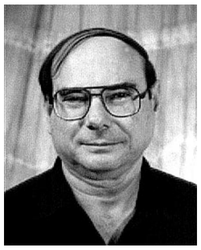
Thao Dang received the French graduate engineer degree in electrical engineering and the DEA (M.Sc.) degree in automatic control from the National College of Electrical Engineering, Grenoble, France, in June and October 1996, respectively. She is currently pursuing the Ph.D. degree from VERIMAG, Gieres, France.

Her research interests are the modeling, verification, and synthesis of hybrid systems.



Oded Maler received the B.A. degree in computer science from the Technion, Haifa, Israel, in 1979, the M.Sc. degree in management science from Tel-Aviv University, Israel, in 1984, and the Ph.D. degree in computer science from the Weizmann Institute of Science, Israel, in 1991.

Since 1994, he has been a CNRS Researcher at the VERIMAG laboratory, Gieres, France, where he leads the hybrid systems group. His research interests include the modeling and verification of discrete, timed, and hybrid systems, as well as the theoretical foundations of these topics.



Amir Pnueli (Member, IEEE) received the B.Sc. degree in applied mathematics from the Technion, Haifa, Israel, in 1962, and the Ph.D. degree in applied mathematics from the Weizmann Institute, Rehovot, Israel, in 1967.

He founded the Department of Computer Science, Tel Aviv University, Israel, where he was a Professor from 1973 to 1980. He has been a Professor of Computer Science at the Weizmann Institute since 1980. His research interests include formal verification and synthesis of reactive, real-time, and hybrid systems, using temporal logic and similar formalisms.

Prof. Pnueli is the recipient of the ACM Turing award for 1996 and the Israel prize in exact sciences for 2000. He is a foreign member of the (American) National Academy of Engineering and holds honorary doctorates from Universite Joseph Fourier, Grenoble, France, and the University of Uppsala.