Computational Techniques for Hybrid System Verification

Alongkrit Chutinan and Bruce H. Krogh, *Fellow, IEEE*

***Abstract*—This paper concerns computational methods for verifying properties of *polyhedral invariant hybrid automata* (PIHA), which are hybrid automata with discrete transitions governed by polyhedral guards. To verify properties of the state trajectories for PIHA, the planar switching surfaces are partitioned to define a finite set of discrete states in an *approximate quotient transition system* (AQTS). State transitions in the AQTS are determined by the reachable states, or *flow pipes*, emitting from the switching surfaces according to the continuous dynamics. This paper presents a method for computing polyhedral approximations to flow pipes. It is shown that the flow-pipe approximation error can be made arbitrarily small for general nonlinear dynamics and that the computations can be made more efficient for affine systems. The paper also describes *CheckMate*, a MATLAB-based tool for modeling, simulating and verifying properties of hybrid systems based on the computational methods previously described.**

***Index Terms*—Hybrid systems, model checking, reachability, verification.**

I. Introduction

THE growing use of computers in modern control systems results in complex dynamical systems called *hybrid systems*, which contain both discrete and continuous dynamics. This paper concerns formal verification of such systems. Given a desired property, called a *specification*, we would like to guarantee that all of the hybrid system behaviors satisfy the specification. This is a very important problem in the validation of the system design, especially for safety-critical applications.

This paper describes computational procedures implemented in *CheckMate*,[1] a MATLAB-based tool for verification of hybrid systems. *CheckMate* models are constructed as Simulink block diagrams, using the Stateflow Toolbox to represent the discrete-state transition logic. The verification procedure in *CheckMate* is based on the general theory of hybrid automata with transition system semantics [1], [2]. To apply this theory, *CheckMate* converts Simulink-Stateflow models into a class of hybrid automata called a *polyhedral-invariant hybrid automata (PIHA)*, which are hybrid automata with invariants and guards defined by linear inequalities (see Section III). As with hybrid systems in general, the PIHA transition system has an infinite (uncountable) state space. To apply standard model checking techniques for verification [3], [4], a finite-state conservative approximation to the hybrid system is constructed, called an *approximate quotient transition system (AQTS)* [5]. If the verification is inconclusive, the AQTS can be refined and the verification can be attempted again.

The main obstacle toward realizing the AQTS for hybrid systems is the lack of effective methods for computing *flow pipes*, that is, the set of continuous state trajectories emanating from a set of initial states [6]. We propose a procedure for computing conservative polyhedral approximations to flow pipes for continuous dynamic systems [7]. The procedure differs from most other approximation methods (e.g., [8]–[11]) in that it deals directly with the dynamics described by continuous state equations and the approximation error for a single flow pipe does not grow with simulation time. We also show that for general nonlinear dynamics, the flow-pipe approximation error can be made as small as desired, albeit at the expense of more computation time. We extend the results in [7] for efficient computation of flow-pipe approximations for *linear* systems to *affine* systems.

This paper is organized as follows. Section II presents the elements of *CheckMate* to provide a context for the formal models and computational procedures described in the rest of the paper. Section III defines the PIHA and the transition system semantics used for verification. This section also describes the AQTS and the role that reachability computations for continuous-state dynamic systems plays in building finite-state systems for verification. We then focus on the problem of approximating flow pipes for nonlinear and linear systems in Section IV. As an example, Section V describes the application of *CheckMate* to verify properties of a batch evaporator system [12]. The concluding section summarizes the contributions of this paper.

II. CheckMate

Recently, several tools have been introduced to perform formal verification of hybrid systems, including UPPAAL [13], HyTech [14], KRONOS [15], Veri-Shift [16], d/dt [17], and the MLD-verifier [18]. In terms of the types of continuous dynamics that can be handled be each of these tools, UPPAAL and KRONOS deal with timed systems, that is, the continuous dynamics are pure integrators; HyTech handles so-called linear hybrid automata, that is, the continuous state derivative vectors are constrained to be in given polyhedra (differential inclusions); d/dt and VeriShift deal with affine dynamical systems;

Manuscript received September 14, 2001. Recommended by Associate Editor S. Sreenivas. This work was supported in part by DARPA under Contract F33615-97-C-1012 and by the Ford Motor Company.

A. Chutinan is with Shinawatra University, Pathumthani 12160, Thailand (e-mail: alongkrit@shinawatra.ac.th).

B. H. Krogh is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890 USA (e-mail: krogh@ece.cmu.edu).

Digital Object Identifier 10.1109/TAC.2002.806655

[1]The computational methods discussed in this paper have been implemented in *CheckMate*, a MATLAB-based verification tool for hybrid systems. Further information on *CheckMate* can be found at the web site http://www.ece.cmu.edu/~webk/CheckMate. Various efforts are underway to improve the computational efficiency and numerical robustness of *CheckMate*.

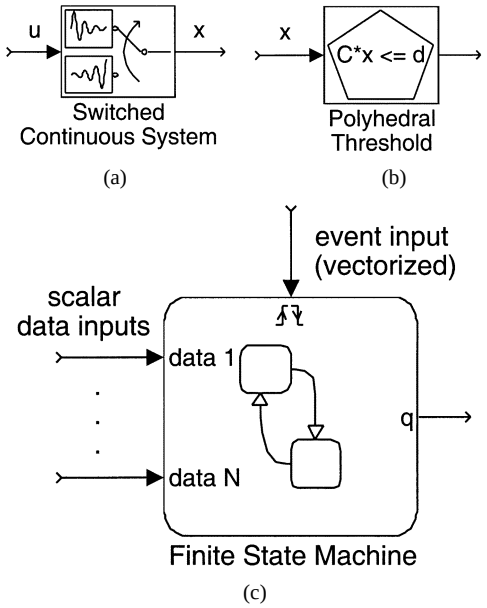

Fig. 1. Major block types in *CheckMate*.

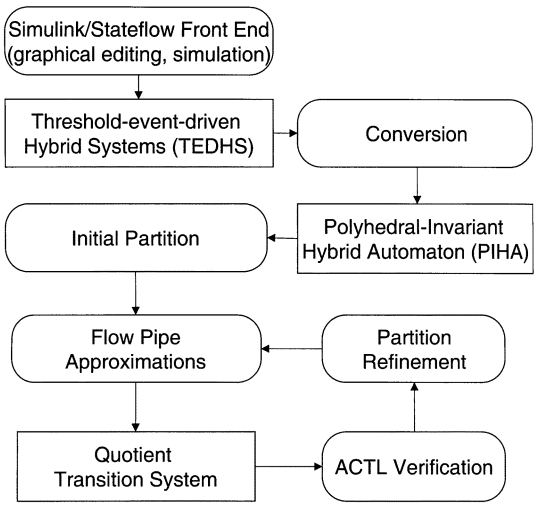

Fig. 2. Overview of *CheckMate* verification procedure.

and the MLD-verifier includes discrete-time linear dynamics. In contrast to these tools, the *CheckMate* tool for hybrid system verification will accept arbitrary nonlinear continuous state equations [19].

CheckMate is implemented in MATLAB, using the Simulink graphical user interface. Fig. 1 illustrates the three major blocks used to build a hybrid system model in *CheckMate*. The first two blocks are custom *CheckMate* blocks implemented with Simulink masking [20]. The third block is a standard Stateflow block. These three blocks are used with other standard Simulink blocks to build hybrid system models that can be simulated as well as verified. The three blocks in Fig. 1 are described as follows.

The *switched continuous system block (SCSB)* defines a continuous dynamics system whose dynamics depends on a discrete-valued input. Fig. 1(a) depicts an SCSB where u is the discrete input and the output x is the continuous state vector for the dynamics in the block. The continuous dynamics is selected by the value of u according to $\dot{x} = f_u(x)$. The discrete input u to an SCSB can only come from finite-state machine blocks (described later). The following three types of ordinary differential equations can be specified for each value of the discrete input u: `nonlinear`, where $\dot{x} = f(x)$ for an arbitrary continuous nonlinear function f; `linear`, where $\dot{x} = Ax + b$ for a constant matrix A and vector b; and `clock`, where $\dot{x} = c$ for a constant vector c. In each case, *CheckMate* uses a flow-pipe approximation procedure that is optimized for the specified type of dynamics. (The flow-pipe representation is exact in the case of `clock` dynamics.)

The *polyhedral threshold block (PTHB)* in Fig. 1(b) defines a convex polyhedron parameterized by a matrix-vector pair (C, d). The input is a continuous state vector x and the output is a Boolean signal indicating whether or not x lies within the convex polyhedron defined by $Cx \leq d$. The input must be constructed from the outputs of SCSBs.

The *finite-state machine block (FSMB)* in Fig. 1(c) is implemented by a *Stateflow* [21] block with the following restrictions. Each *event* input must be a logical function of the outputs of PTHBs. Each data input must be a logical function of the outputs of PTHBs or FSMBs. Only one discrete-valued output signal is allowed. The Stateflow diagram must contain no hierarchy and each state must assign a unique value to the data output in its *entry* action. No other action is permitted on any state or transition label string.

The *CheckMate* verification procedure, shown in Fig. 2, starts with the conversion of the Simulink model into an equivalent PIHA (defined in Section III). The analysis of the PIHA is limited to a user-specified polyhedral region called the *analysis region* in the continuous state space. A finite-state AQTS is constructed to verify properties of the PIHA state trajectories [5] (see Section III-C). The discrete states in an AQTS are defined by an initial partition of the switching surfaces (i.e., the boundaries of the polyhedra defined in the threshold blocks) constructed according to parameters specified by the user. Transitions between states in the AQTS are computed using the flow-pipe approximations (see Section IV). The AQTS is then verified against a given specification using standard model checking techniques for finite-state transition systems. The user defines specifications to be verified using ACTL, a restricted class of *computation tree logic* (CTL) [3]. If the verification fails due to the coarseness of the discretization in the AQTS, the partition for the AQTS is refined to give a tighter approximation. The process can be repeated until the AQTS satisfies the specification or the user terminates the verification.

Remark: When the ACTL expression is found to be true, it can be concluded that the specification is true for the given PIHA. Since PIHA verification problems are undecidable in general [22], however, it is impossible to determine *a priori* whether or not the procedure described above will terminate. That is, if the verification is inconclusive for a given AQTS, the user usually cannot determine if further refinement will help. In our experience, the verification is often successful after two or three refinements, and even when the verification fails, the user typically gains valuable knowledge about the hybrid system behaviors from the construction and simulation of the *CheckMate* model.

Section III presents the formal model of hybrid systems used in *Checkmate*. We then focus on the theory behind the flow-pipe approximation in Section IV. Section V presents an application of *CheckMate* to a batch reactor.

III. Polyhedral-Invariant Hybrid Automata (PIHA)

To develop effective computational tools for verification, we focus on a particular class of hybrid systems called PIHA.

A. PIHA

We define a PIHA using the formalism from [23] (with some restrictions).

Definition 1: A PIHA is a tuple $H = (X, X_0, F, E, I, G)$ where

- $X = X_C \times X_D$, where $X_C \subseteq R^n$ is the continuous state–space and X_D is a finite set of discrete locations;
- F is a function that assigns to each discrete location $u \in X_D$ a vector field $f_u(\cdot)$ on X_C;
- $I : X_D \to 2^{X_C}$ assigns to $u \in X_D$ an *invariant* set of the form $I(u) \subseteq X_C$ where $I(u)$ is a nondegenerate convex polyhedron;
- $E \subseteq X_D \times X_D$ is a set of discrete transitions;
- $G : E \to 2^{X_C}$ assigns to $e = (u, u') \in E$ a guard set that is a union of faces of $I(u)$;
- $X_0 \subseteq X$ is the set of initial states of the form $X_0 = \bigcup_i (P_i, u_i)$ where each $P_i \subseteq I(u_i)$ is a polytope and $u_i \in U$; here, the notation (P, u) means the set $\{(x, u) \in X | x \in P\}$;
- I, G, and E must satisfy the following *coverage requirements*: 1) for each u, $\partial I(u) = \cup_{e \in E \,|\, e=(u,u') \text{for some} u' \in X_D} G(e)$, that is, the guards for u cover the faces of the invariant for u; and 2) for all $e = (u, u') \in E$, $G(e) \subseteq I(u')$, that is, events do not lead to transitions that violate invariants.

As previously defined, the PIHA differs from general hybrid automata [2] in the following respects: 1) there are no so-called reset mappings associated with the discrete transitions, which means there are no discontinuities in the continuous-state trajectories; 2) the invariants are defined by linear inequalities (hence the name "polyhedral invariant"); and 3) the guards are faces of the invariants, which means that a discrete-state transition occurs immediately when the continuous-state trajectory reaches a guard set. Point 3) is reflected in the semantics defined for the PIHA Section III-B.

B. Discrete-Trace Transition Systems

A hybrid system can be thought of as a *transition system* $T = (Q, \to, Q_0)$ where Q is the set of states, $\to$ is the transition relation, and Q_0 is the set of initial states [1], [2]. In this paper, we are interested in the transition system that abstracts away the continuous dynamics and retains the hybrid system behaviors only at the instants of discrete transitions. We call this the *discrete-trace transition system*. To define this transition system, we use the following notation and definitions. Given an initial PIHA state (x_0, u), we denote the continuous trajectory in location u by $\zeta_{(x_0,u)}(\cdot)$ where $\zeta_{(x_0,u)}(0) = x_0$ and $\dot{\zeta}_{(x_0,u)}(t) = f_u(\zeta_{(x_0,u)}(t))$, $\forall\, t \geq 0$ (until a discrete transition occurs). Given a PIHA H, the set of states through which the PIHA can enter a location, called the *entry states*, is defined as $X_{\text{entry}} = \{(x', u') \in X \,|\, \text{for some } (x, u) \in X \text{ and } (u, u') \in E, x' \in G((u, u'))\}$. We now define the discrete-trace transition system for a PIHA.

Definition 2: Given a PIHA H, its *discrete-trace transition system* is given by $T_H = (Q_H, \to_H, Q_0)$ where $Q_0 = X_0$, $Q_H = X_0 \cup X_{\text{entry}} \bigcup_{u \in X_D} \{q_u^{\perp}\}$, and the transition relation $\to_H$ is defined as follows.

i) *Discrete Transitions:* $(x, u) \to_H (x', u')$ iff $u' \neq u$ and there exist $e = (u, u') \in E$ and $t_1 > 0$ such that $\zeta_{(x,u)}(t_1) = x'$, $x' \in G(e)$, and $\zeta_{(x,u)}(t) \in I(u) \setminus \partial I(u)$, i.e., the interior of $I(u)$, for all $t \in (0, t_1)$;

ii) *Null Transitions:* $(x, u) \to_H q_u^{\perp}$ iff $\zeta_{(x,u)}(t) \in I(u) \setminus \partial I(u)$ for all $t \geq 0$.

In Definition 2, the discrete transitions comprise all continuous-state trajectories in the PIHA between location transitions. The null transitions comprise all continuous-state trajectories that remain in a location indefinitely.

C. AQTSs

The standard approach to verification of hybrid systems is to construct a finite-state *bisimulation* of the infinite-state transition system [1], [24]. Bisimulations are constructed using a finite partition of the original state space, leading to a so-called *quotient transition system* (QTS). The difficulty is that finite-state bisimulations are known to exist only for hybrid systems with trivial continuous dynamics (e.g., see [25]). In general, finite bisimulations do not exist, which means that verification problems for hybrid systems are usually undecidable (e.g., see [22]). Nevertheless, the quotient transition system computed for any partition of the transition system state space is a *simulation* of the transition system. This means if a *universal specification* (that is, a specification that must be true for all possible trajectories) is true for the QTS, it is also true for the infinite-state transition system (that is, for the hybrid system). Therefore, even for problems that are undecidable in general, it is possible to verify certain specifications. It is not possible, however, to predict whether or not a given specification is going to be verifiable.

The state–space for the PIHA discrete-state transition system is the set of continuous states on the invariant boundaries. To construct the QTS for a partition of this transition system, it is necessary to compute the transitions between elements of the partitions. This involves computing the sets of continuous states that are reached starting from an element of the partition (a set of continuous states on the boundary of an invariant) and finding out where the set of reachable states intersects with other elements of the partition (other subsets of the invariant boundary). We call this set of reachable states the *flow pipe* for the continuous dynamics corresponding to the invariant.

Flow pipes can be represented and computed exactly only for particular types of simple dynamics (e.g., for *clock* dynamics). In general, one must settle for a conservative approximation of the flow pipe, leading to an approximation of the QTS called an AQTS. Fig. 3 illustrates the computation of AQTS transitions

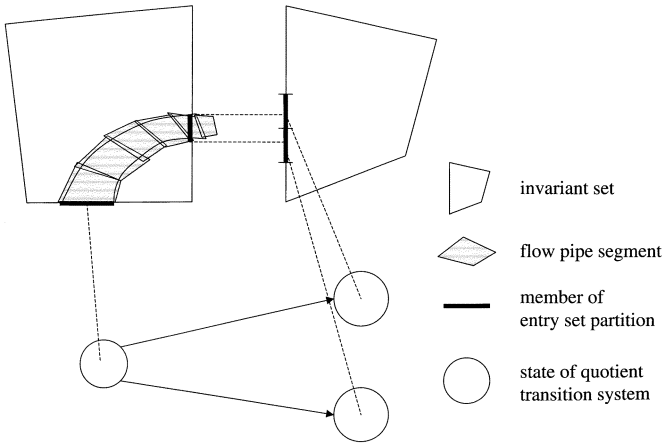

Fig. 3. Illustration of the construction of transitions in the AQTS.

using flow-pipe approximations. In this example, one element of the partition is mapped to two other elements.

The general theory of using the AQTS for verification of infinite state systems is developed in a companion paper [5]. Section IV presents the details of a particular way to construct the flow-pipe approximations for general continuous dynamics.

IV. Computing Flow Pipe Approximations

This section describes our method for computing flow-pipe approximations, presented originally in [7]. We consider an autonomous dynamical system with state equation $\dot{x}(t) = f(x(t))$ in the bounded and connected domain $W \subset R^n$. We assume that vector field f is *Lipschitz*, that is, there exists a constant L such that $||f(x_1) - f(x_2)|| \leq L||x_1 - x_2||$ for all $x_1, x_2 \in W$. The solution to the state equation starting from the initial state x_0 at time t is denoted by $x(t, x_0)$. The Lipschitz condition implies that for every initial state x_0 there is a unique solution $x(t, x_0)$ to the state equation. The set of reachable states at time t from a set of initial states X_0 is defined as $\mathcal{R}_t(X_0) = \{x_f \,|\, x_f = x(t, x_0), \text{ for some } x_0 \in X_0\}$. The flow pipe from X_0 in the time interval $[t_1, t_2]$ is defined as $\mathcal{R}_{[t_1,t_2]}(X_0) = \bigcup_{t\in[t_1,t_2]} \mathcal{R}_t(X_0)$.

To construct the AQTS, it is necessary is to compute flow-pipe approximations that are conservative. That is, given a polyhedral set of initial continuous states X_0 and a final time t_f, we must compute a flow-pipe approximation, denoted by $\hat{\mathcal{R}}_{[0,t_f]}(X_0)$, such that $\mathcal{R}_{[0,t_f]}(X_0) \subseteq \hat{\mathcal{R}}_{[0,t_f]}(X_0)$. Our approximation method constructs $\hat{\mathcal{R}}_{[0,t_f]}(X_0)$ as the union of convex polyhedra, where each polyhedron is an over approximation to a flow-pipe segment corresponding to an interval of time. If the time interval $[0, t_f]$ is divided into N time segments, $[0, t_1], [t_1, t_2], \ldots, [t_{N-1}, t_f]$, the complete flow-pipe approximation from $t = 0$ to $t = t_f$ is the union of all N flow-pipe segments $\hat{\mathcal{R}}_{[0,t_f]}(X_0) = \bigcup_{k=1,\ldots,N} \hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0)$, where $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq \hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0)$. As illustrated in Fig. 3, the AQTS transition relation is computed by finding intersections of flow-pipe segment approximations with the faces of the associated polyhedral invariant.

A. Approximating a Flow Pipe Segment

We use the following notation to describe the polyhedral approximation of a flow-pipe segment. Given a pair $(C, d) \in R^{m\times n} \times R^m$, we write POLY$(C, d)$ to denote the polytope $\{x \,|\, Cx \leq d\}$. Each row c_i^T, $i = 1, \ldots, m$ of C is the unit normal vector to the ith face of the polytope. Given a polytope P, we write $V(P)$ to denote the set of vertices of P. Given a finite set of points Υ, we write $CH(\Upsilon)$ to denote the *convex hull* of Υ [26].

The polyhedral approximation of a flow-pipe segment $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$ is computed as a matrix-vector pair $(C, d) \in R^{m\times n} \times R^m$, such that $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq$ POLY(C, d). We are also interested in making the approximation error as small as possible.

There are two steps in the procedure for finding (C, d). First, the rows of C are selected. This determines the normal vectors for the faces of the polyhedron to approximate $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$. Then, given C, we compute d as the solution to the following optimization problem:

$$\begin{aligned} \min_{d} \quad & \text{volume}[\text{POLY}(C, d)] \\ \text{s.t.} \quad & \mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq \text{POLY}(C, d). \end{aligned} \tag{1}$$

This optimization problem finds the polyhedron that minimizes the approximation error for $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$ given that the normal vectors for the polyhedron are specified by the rows of C. We denote the set POLY(C, d^*), where d^* is the solution to (1), by $S_C^{\min}(\mathcal{R}_{[t_{k-1},t_k]}(X_0))$. Hence, our approximation for the k^{th} flow-pipe segment is $\hat{\mathcal{R}}_{[t_{k-1},t_k]}(X_0) = S_C^{\min}(\mathcal{R}_{[t_{k-1},t_k]}(X_0))$. Throughout the remainder of the paper, the notation $\hat{\mathcal{R}}$ will stand for approximations computed using this particular procedure.

The components of d^* solving (1) can be found by solving the following constrained optimization problems for $i = 1, \ldots, m$:

$$\begin{aligned} \max_{x} \quad & c_i^T x \\ \text{s.t.} \quad & x \in \mathcal{R}_{[t_{k-1},t_k]}(X_0). \end{aligned} \tag{2}$$

Using the definition of $\mathcal{R}_{[t_{k-1},t_k]}(X_0)$, we can rewrite (2) as

$$\begin{aligned} \max_{x_0,t} \quad & c_i^T x(t, x_0) \\ \text{s.t.} \quad & x_0 \in X_0 \\ & t \in [t_{k-1}, t_k]. \end{aligned} \tag{3}$$

Proposition 1: Let $(x_{0,i}^*, t_i^*)$ be solutions to (3) for $i = 1, \ldots, m$. The solution to (1) is given by $d_i^* = c_i^T x(t_i^*, x_{0,i}^*)$, for $i = 1, \ldots, m$.

Proof: To show this, first note that since $x(t_i^*, x_{0,i}^*)$ is the solution to (2) for $i = 1, \ldots, m$, we have that d^* is feasible, i.e., $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \subseteq$ POLY(C, d^*). Now, consider an alternative solution $d \neq d^*$. For any i such that $d_i \neq d_i^*$, if $d_i < d_i^*$, then there exists a state $x \in \mathcal{R}_{[t_{k-1},t_k]}(X_0)$ such that $c_i^T x > d_i$, namely the state x^* such that $c_i^T x^* = d_i^*$. This implies that $\mathcal{R}_{[t_{k-1},t_k]}(X_0) \not\subseteq$ POLY(C, d) and d is not feasible. If $d_i > d_i^*$, assume that $d_j \geq d_j^*$ for all $j \neq i$ so that d is a feasible solution. Then, POLY$(C, d^*) \subseteq$ POLY(C, d). Thus, no solution produces a feasible set smaller than POLY(C, d^*). ■

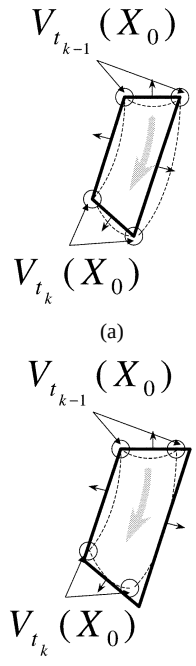

Fig. 4. Flow-pipe approximation procedure for one segment.

To solve (3), one needs to solve the state equation to find $x(t, x_0)$ for each t and x_0. The solution $x(t, x_0)$ can be computed numerically using an ordinary differential equation (ODE) solver. Thus, by embedding numerical simulation of the continuous state equations into the routine for computing the objective function, one can use a software package such as the MATLAB Optimization Toolbox to solve (3) numerically. Note that (3) is not a convex optimization problem in general. We discuss this issue in Section IV-D.

We have found the following heuristic effective for computing the set of normal vectors (the rows of the matrix C) used in (3). We begin by computing the sets $V_{t_{k-1}}(X_0)$ and $V_{t_k}(X_0)$, where $V_t(X_0) = \{x(t, v) \,|\, v \in V(X_0)\}$, of vertices of X_0 at times t_{k-1} and t_k using numerical simulations. We then use these points to form a convex hull $\Phi_{[t_{k-1},t_k]}(X_0) = \text{POLY}(C_\Phi, d_\Phi) = CH(V_{t_{k-1}}(X_0) \cup V_{t_k}(X_0))$, as illustrated for a 2-D case in Fig. 4(a). Finally, we use the *outward* pointing normal vectors from this convex hull to solve (3), as illustrated in Fig. 4(b).

B. Flow-Pipe Approximations for Affine Systems

The procedure described above applies to arbitrary nonlinear systems. It is computationally expensive, however, since a simulation of the state equations is embedded in the optimization problem for each face of the polyhedral approximation. The computations can be reduced significantly for affine dynamic systems with dynamics of the form $\dot{x}(t) = Ax(t) + b$, where A is an $n \times n$ constant matrix and b is an $n \times 1$ constant vector. In this case, the analytical solution to the state equation is given by $x(t, x_0) = e^{At}x_0 + \nu(t)$, where $\nu(t) = e^{At}\int_0^t e^{-A\tau}b d\tau$, and Lemma 1 follows from time invariance.

Lemma 1: Given an affine system $\dot{x}(t) = Ax(t) + b$ and $h \geq 0$, $x(t + h, x_0) = e^{At}x(h, x_0) + \nu(t)$, where $\nu(t)$ is previously defined.

We introduce the following notation. Given a set X, a matrix M, and a vector v, we write the set Y obtained by applying the affine transformation $M(\cdot) + v$ to each element $x \in X$ as $Y = MX + v = \{y | y = Mx + v, \text{ for some } x \in X\}$. A polytope described by a set of inequality constraints can be transformed by applying Lemma 2 to each inequality constraint in the set.

Lemma 2: Given a set $X = \{x \,|\, c^T x \leq d\}$, an invertible linear transformation M, and a vector v, the set $Y = MX + v$ can be written as $Y = \{y \,|\, \hat{c}^T y \leq \hat{d}\}$ where $\hat{c} = (M^{-1})^T c$ and $\hat{d} = d + c^T M^{-1} v$

Proof: $y \in Y \iff x = M^{-1}(y - v) \in X \iff c^T M^{-1}(y - v) \leq d \iff c^T M^{-1} y \leq d + c^T M^{-1} v$. ■

The following proposition states that when the dynamics are affine, the set of reachable states for any time interval $[t, t + \Delta]$ is equal to an affine transformation of the set of reachable states for the time interval $[0, \Delta]$.

Proposition 2: Given an affine system $\dot{x}(t) = Ax(t) + b$, $\mathcal{R}_{[t,t+\Delta]}(X_0) = e^{At}\mathcal{R}_{[0,\Delta]}(X_0) + \nu(t)$.

Proof: First, note that since e^{At} is invertible, the affine transformation from $\mathcal{R}_{[0,\Delta]}$ to $\mathcal{R}_{[t,t+\Delta]}$ is a bijection. Thus, for each $z \in \mathcal{R}_{[0,\Delta]}$ there is a $z' \in \mathcal{R}_{[t,t+\Delta]}$ such that $z' = e^{At}z + \nu(t)$ and *vice versa*. In particular, for each $z \in \mathcal{R}_{[0,\Delta]}$, we have that $z = x(h, x_0)$ for some $x_0 \in X_0$ and $h \in [0, \Delta]$ and we have by Lemma 1 that $z' = x(t + h, x_0)$. Similarly, for each $z' \in \mathcal{R}_{[t,t+\Delta]}$, we have that $z' = x(t+h, x_0)$ for some $x_0 \in X_0$ and $h \in [0, \Delta]$, and we have by Lemma 1 that $z = x(h, x_0)$. ■

The following proposition states that the affine transformation in Proposition 2 also applies to the polyhedral approximations to flow-pipe segments.

Proposition 3: $\hat{\mathcal{R}}_{[t,t+\Delta]}(X_0) = e^{At}\hat{\mathcal{R}}_{[0,\Delta]}(X_0) + \nu(t)$.

Proof: We compare the procedure for computing the flow-pipe segments $\hat{\mathcal{R}}_{[0,\Delta]}(X_0)$ and $\hat{\mathcal{R}}_{[t,t+\Delta]}(X_0)$. For the time intervals $[0, \Delta]$ and $[t, t + \Delta]$, we construct the convex hulls $\Phi_{[0,\Delta]}(X_0) = CH(V_0(X_0) \cup V_\Delta(X_0))$ and $\Phi_{[t,t+\Delta]}(X_0) = CH(V_t(X_0) \cup V_{t+\Delta}(X_0))$, respectively. Proposition 2 implies that $\Phi_{[t,t+\Delta]}(X_0) = e^{At}\Phi_{[0,\Delta]}(X_0) + \nu(t)$. Thus, if $\Phi_{[0,\Delta]}(X_0) = \text{POLY}(C, d)$ then we have by Lemma 2 that $\Phi_{[t,t+\Delta]}(X_0) = \text{POLY}(\hat{C}, \hat{d})$ with $\hat{C} = Ce^{-At}$ and $\hat{d} = d + Ce^{-At}\nu(t)$.

For the time interval $[0, \Delta]$, we use C and $\mathcal{R}_{[0,\Delta]}(X_0)$ to solve (1) and write the optimization problem (3) corresponding to the i^{th} normal vector in C as

$$\begin{aligned} \max_{x_0,\tau} \quad & c_i^T x(\tau, x_0) \\ \text{s.t.} \quad & x_0 \in X_0, \qquad \tau \in [0, \Delta]. \end{aligned} \tag{4}$$

Similarly, for the time interval $[t, t + \Delta]$, we use $\hat{C}$ and $\mathcal{R}_{[t,t+\Delta]}(X_0)$ to solve (1) and write the optimization problem (3) corresponding to the i^{th} normal vector in $\hat{C}$ as

$$\begin{aligned} \max_{x_0,\tau} \quad & \hat{c}_i^T x(t + \tau, x_0) \\ \text{s.t.} \quad & x_0 \in X_0, \qquad \tau \in [0, \Delta]. \end{aligned} \tag{5}$$

Substituting $x(t+\tau, x_0) = e^{At}x(\tau, x_0) + \nu(t)$ (using Lemma 1) and $\hat{c}_i^T = c_i^T e^{-At}$ into (5), we obtain

$$\begin{aligned} \max_{x_0,\tau} \quad & c_i^T x(\tau, x_0) + c_i^T e^{-At}\nu(t) \\ \text{s.t.} \quad & x_0 \in X_0, \qquad \tau \in [0, \Delta]. \end{aligned} \tag{6}$$

We observe that (4) and (6) differ only by the constant term $c_i^T e^{-At}\nu(t)$ in the objective function. Thus, we conclude that if d_i^* is the solution to (4) for the interval $[0, \Delta]$, then $\hat{d}_i^* = d_i^* + c_i^T e^{-At}\nu(t)$ is the solution to (6) for the interval $[t, t+\Delta]$. This implies that $\hat{\mathcal{R}}_{[0,\Delta]}(X_0) = \text{POLY}(C, d^*)$ and $\hat{\mathcal{R}}_{[t,t+\Delta]}(X_0) = \text{POLY}(\hat{C}, \hat{d}^*)$. By Lemma 2, we have that $\hat{\mathcal{R}}_{[t,t+\Delta]}(X_0) = e^{At}\hat{\mathcal{R}}_{[0,\Delta]}(X_0) + \nu(t)$. ■

Proposition 3 implies that our computational procedure for each flow-pipe segment depends only on the size of the time step Δ. To compute the segment between time t and $t+\Delta$, we may apply our computational procedure for the time interval $[0, \Delta]$ and then apply the transformation $e^{At}(\cdot) + \nu(t)$, which depends on the starting time of the segment, to the resulting polytope. The matrix e^{At} and the vector $\nu(t)$ can be computed numerically (using numerical integration for $\nu(t)$). This suggests that the efficiency of the flow-pipe computation can be improved by caching the resulting polytopes (before the transformation) for different Δ's and transforming them to the starting times of the segments as needed. Examples of flow pipes computed for nonlinear and linear systems can be found in [7].

C. Error Analysis for the Flow Pipe Approximations

In this section, we show that it is always possible to make the approximation error arbitrarily small using the polyhedral approximations to flow-pipe segments to approximate the complete flow pipe for a time interval $[0, t_f]$. For the affine system case, one can see that the accuracy of the approximation can be improved by simply using smaller time steps. This is because the reachable set at any time $\mathcal{R}_t(X_0)$ is a polytope which is an affine transformation of X_0. The flow pipe is simply the union of all these polytopes over the time interval $[0, t_f]$. Each segment beginning at time t_k approaches $\mathcal{R}_{t_k}(X_0)$ as Δ_k gets smaller and smaller.

For nonlinear systems, reducing the lengths of the time segments (i.e., Δ) may not be sufficient to guarantee the approximation converges to the flow pipe. The reason is that the reachable set at a given time starting from a polyhedral set of initial state will not necessarily be a polyhedron [7]. In order to approximate the flow pipe with an arbitrarily small approximation error, it may be necessary to partition the initial set X_0 as well as the time interval $[0, t_f]$ and into subsets that are small enough. The precise definition of "small enough" is presented in Proposition 4.

Before stating and proving Proposition 4, we introduce some notations and mathematical preliminaries from [9] and [27]. For a vector $x \in R^n$, $\|x\|$ denotes the Euclidean norm. A unit ball centered at the origin is denoted by $B = \{x \,|\, \|x\| \le 1\}$. For sets $U, V \subset R^n$, $U + V = \{u + v \,|\, u \in U \text{ and } v \in V\}$. For $\alpha \in R$, $\alpha U = \{\alpha u \,|\, u \in U\}$. For $\epsilon > 0$, $B_\epsilon(x)$ is a ϵ-ball centered at x, i.e., $B_\epsilon(x) = \{x' \,|\, \|x' - x\| \le \epsilon\}$. If the ball center is the origin, we simply write B_ϵ. As the metric for the approximation error, we use the *Hausdorff distance* between two sets $U, V \subset R^n$, which is defined as

$$\text{dist}(U, V) = \inf\{\epsilon \,|\, U \subseteq V + \epsilon B \text{ and } V \subseteq U + \epsilon B\}.$$

We define the metric on the size of a connected set $U \subset R^n$ as $\|U\| = \sup_{x_1, x_2 \in U} \|x_1 - x_2\|$. The following lemmas are used in the sequel.

Lemma 3: Given $U, V, W \subset R^n$, $(U \cup V) + W = (U + W) \cup (V + W)$.

Proof: $(U \cup V) + W \subseteq (U + W) \cup (V + W)$: Suppose $x \in (U \cup V) + W$. Then, $x = x' + w$ for some $x' \in U \cup V$ and $w \in W$. Thus, $x = u + w$ or $x = v + w$ for some $u \in U, v \in V$, and $w \in W$, which implies that $x \in (U + W) \cup (V + W)$.

$(U \cup V) + W \supseteq (U + W) \cup (V + W)$: Suppose $x \in (U + W) \cup (V + W)$. Then, $x = u + w$ or $x = v + w$ for some $u \in U$, $v \in V$, and $w \in W$. Thus, $x = x' + w$ for some $x' \in U \cup V$ and $w \in W$, which implies that $x \in (U \cup V) + W$. ■

Lemma 4: Given $U, V \subset R^n$, $x \in R^n$, and $\epsilon > 0$, if $U \subseteq B_{\epsilon/2}(x)$ and $V \subseteq B_{\epsilon/2}(x)$, then $\text{dist}(U, V) \le \epsilon$.

Proof: First, note that for all $x_1 \in U$ and $x_2 \in V$, $\|x_1 - x_2\| \le \epsilon$. We show that $U \subseteq V + \epsilon B$ and $V \subseteq U + \epsilon B$. We show $U \subseteq V + \epsilon B$ by contradiction. Suppose $x_1 \in U$ but $x_1 \notin V + \epsilon B$. Then there exists $x_2 \in V$ such that $\|x_1 - x_2\| > \epsilon$, a contradiction. $V \subseteq U + \epsilon B$ follows from a similar argument. ■

Lemma 5: Given $U_1, U_2, V_1, V_2 \subset R^n$, if $\text{dist}(U_1, V_1) \le \epsilon$ and $\text{dist}(U_2, V_2) \le \epsilon$, then $\text{dist}(U_1 \cup U_2, V_1 \cup V_2) \le \epsilon$.

Proof: Since $\text{dist}(U_1, V_1) \le \epsilon$ and $\text{dist}(U_2, V_2) \le \epsilon$, we have that $U_1 \subseteq V_1 + \epsilon B$, $V_1 \subseteq U_1 + \epsilon B$, $U_2 \subseteq V_2 + \epsilon B$, and $V_2 \subseteq U_2 + \epsilon B$. These results imply that $U_1 \cup U_2 \subseteq (V_1 + \epsilon B) \cup (V_2 + \epsilon B)$ and $V_1 \cup V_2 \subseteq (U_1 + \epsilon B) \cup (U_2 + \epsilon B)$. It then follows from Lemma 3 that $U_1 \cup U_2 \subseteq (V_1 \cup V_2) + \epsilon B$ and $V_1 \cup V_2 \subseteq (U_1 \cup U_2) + \epsilon B$. ■

Lemma 6: Gronwall–Bellman Inequality [27]: Let $\lambda : [a, b] \to R$ be continuous and $\mu : [a, b] \to R$ be continuous and nonnegative. If a continuous function $y : [a, b] \to R$ satisfies $y(t) \le \lambda(t) + \int_a^t \mu(s)y(s)ds$ for $a \le t \le b$, then on the same interval $y(t) \le \lambda(t) + \int_a^t \lambda(s)\mu(s)e^{\left[\int_s^t \mu(\tau)d\tau\right]}ds$.

Lemma 7: [27] Let $f(x)$ be Lipschitz in x on W with a Lipschitz constant L, where $W \subset R^n$ is an open connected set. Let x_1, x_2 be initial conditions such that $x(t, x_1), x(t, x_2) \in W$ for all $t \in [0, t_f]$. Then, $\|x(t, x_1) - x(t, x_2)\| \le e^{Lt}\|x_1 - x_2\|$.

Lemma 8: Let $f(x)$ be Lipschitz in x on W with a Lipschitz constant L, where $W \subset R^n$ is an open connected set. Let x_0 be an initial condition such that $x(t, x_0) \in W$ for all $t \in [0, t_f]$. Let $y(t) = x(t, x_0)$. Then, for $t, t_0 \in [0, t_f]$ such that $t \ge t_0$, $\|y(t) - y(t_0)\| \le (\|f(y(t_0))\|/L)\left(e^{L(t-t_0)} - 1\right)$.

Proof: First, we note that $y(t) - y(t_0) = \int_{t_0}^t f(y(\tau))d\tau$. Thus

$$\|y(t) - y(t_0)\| \le \int_{t_0}^t \|f(y(\tau))\|d\tau. \tag{7}$$

From the Lipschitz condition, we have $\|f(y(\tau)) - f(y(t_0))\| \le L\|y(\tau) - y(t_0)\|$, which implies that

$$\|f(y(\tau))\| \le \|f(y(t_0))\| + L\|y(\tau) - y(t_0)\|. \tag{8}$$

Therefore, we have from (7) and (8) that $\|y(t) - y(t_0)\| \leq \|f(y(t_0))\|(t - t_0) + \int_{t_0}^{t} L\|y(\tau) - y(t_0)\|d\tau$. By the Gronwall–Bellman inequality (Lemma 6), we have

$$\|y(t) - y(t_0)\| \leq \|f(y(t_0))\|(t - t_0) + \int_{t_0}^{t} \|f(y(t_0))\|(s - t_0)Le^{\left[\int_s^t L d\tau\right]} ds. \quad (9)$$

The second term on the right-hand side of the aforementioned inequality reduces to $-\|f(y(t_0))\|(t - t_0) + (\|f(y(t_0))\|/L)\left(e^{L(t-t_0)} - 1\right)$. Replacing the second term on the right-hand side of (9) with this simplification, we have the desired result. ∎

Lemma 9: Let $f(x)$ be Lipschitz in x on W with a Lipschitz constant L, where $W \subset R^n$ is an open connected set. Let x_0 and x_0^* be initial conditions such that $x(t, x_0), x(t, x_0^*) \in W$ for all $t \in [0, t_f]$. For $t', t \in [0, t_f]$ and $x_0, x_0^* \in W$ such that $t' \geq t$, $t' - t \leq \delta_t$, and $\|x_0 - x_0^*\| \leq \delta_{x_0}$

$$\|x(t', x_0) - x(t, x_0^*)\| \leq \frac{\|f(x(t, x_0^*))\|}{L}\left(e^{L\delta_t} - 1\right) + e^{L(t+\delta_t)}\delta_{x_0}.$$

Proof: Since $x(t', x_0) - x(t, x_0^*) = [x(t', x_0) - x(t, x_0)] + [x(t, x_0) - x(t, x_0^*)]$, we have that $\|x(t', x_0) - x(t, x_0^*)\| \leq \|x(t', x_0) - x(t, x_0)\| + \|x(t, x_0) - x(t, x_0^*)\|$. By Lemmas 7 and 8

$$\|x(t', x_0) - x(t, x_0^*)\| \leq \frac{\|f(x(t, x_0))\|}{L}\left(e^{L(t'-t)} - 1\right) + e^{Lt}\|x_0 - x_0^*\|. \quad (10)$$

From the Lipschitz condition, we have that $\|f(x(t, x_0)) - f(x(t, x_0^*))\| \leq L\|x(t, x_0) - x(t, x_0^*)\|$, which implies that $\|f(x(t, x_0))\| \leq \|f(x(t, x_0^*))\| + L\|x(t, x_0) - x(t, x_0^*)\|$. It then follows from Lemma 7 that

$$\|f(x(t, x_0))\| \leq \|f(x(t, x_0^*))\| + Le^{Lt}\|x_0 - x_0^*\|. \quad (11)$$

From (10) and (11), we have that

$$\|x(t', x_0) - x(t, x_0^*)\| \leq \left(\frac{\|f(x(t, x_0^*))\|}{L} + e^{Lt}\|x_0 - x_0^*\|\right) \times \left(e^{L(t'-t)} - 1\right) + e^{Lt}\|x_0 - x_0^*\|.$$

Simplifying the right-hand side of this inequality, we have

$$\|x(t', x_0) - x(t, x_0^*)\| \leq \frac{\|f(x(t, x_0^*))\|}{L} \times \left(e^{L(t'-t)} - 1\right) + e^{Lt'}\|x_0 - x_0^*\|.$$

The proposition then follows from the fact that $t' - t \leq \delta_t$ and $\|x_0 - x_0^*\| \leq \delta_{x_0}$. ∎

Recall that given a set P and a matrix C containing a set of normal vectors in its rows, we denote the smallest polyhedron with face-normal vectors given by the rows of C that contains P with $S_C^{\min}(P)$. Let I_n denote the $n \times n$ identity matrix. We define a special matrix BOX_n that gives the normal vectors for a hyper-rectangle in R^n as $\text{BOX}_n = \begin{bmatrix} I_n \\ -I_n \end{bmatrix}$.

Lemma 10: Given a matrix C, $S_{C'}^{\min}(B_\epsilon) \subseteq S_{\text{BOX}_n}^{\min}(B_\epsilon) \subseteq B_{\sqrt{n}\cdot\epsilon}$ where $C' = \begin{bmatrix} C \\ \text{BOX}_n \end{bmatrix}$.

Proof: The first containment follows immediately because $S_{C'}^{\min}(B_\epsilon)$ has more constraints than $S_{\text{BOX}_n}^{\min}(B_\epsilon)$. The second containment follows because $S_{\text{BOX}_n}^{\min}(B_\epsilon)$ is the hyperrectangle $[-\epsilon, \epsilon]^n$ and the maximum Euclidean norm of $\sqrt{n} \cdot \epsilon$ occurs at corner points. ∎

The following proposition demonstrates that the flow-pipe approximation error can be made arbitrarily small by using appropriate partitions of the time interval $[0, t_f]$ and the initial state set X_0.

Proposition 4: Given a connected set $W \subset R^n$ such that $\mathcal{R}_{[0,t_f]}(X_0) \subseteq W$, let $f(x)$ be Lipschitz in x on W with a Lipschitz constant L and define $M = \sup_{x \in W} \|f(x)\|$. For a time-step partition $[0 = t_0, t_1], \ldots, [t_{N-1}, t_N = t_f]$ of the time interval $[0, t_f]$, let $\mathcal{P}_{X_0}^k$ be a finite polyhedral partition of X_0 associated with the time step $[t_{k-1}, t_k]$. For any $\epsilon > 0$, if

i) the time step partition is uniform with $N = t_f/\delta_t$, $t_k = t_{k-1} + \delta_t$ for $k = 1, \ldots, N$, and $0 < \delta_t < (1/L)\ln(1 + (\epsilon/2\sqrt{n})(L/M))$;
ii) for each time interval $[t_{k-1}, t_k]$, $\mathcal{P}_{X_0}^k$ is such that for each $P \in \mathcal{P}_{X_0}^k$, $\|P\| \leq \delta_{x_0}^k$ where $0 < \delta_{x_0}^k \leq e^{-Lt_k}((\epsilon/2\sqrt{n}) - (M/L)(e^{L\delta_t} - 1))$

then

$$\text{dist}(\hat{\mathcal{R}}_{[0,t_f]}(X_0), \mathcal{R}_{[0,t_f]}(X_0)) \leq \epsilon$$

where $\mathcal{R}_{[0,t_f]}(X_0) = \bigcup_{k=1}^{N} \bigcup_{P \in \mathcal{P}_{X_0}^k} \mathcal{R}_{[t_{k-1},t_k]}(P)$ and $\hat{\mathcal{R}}_{[0,t_f]}(X_0) = \bigcup_{k=1}^{N} \bigcup_{P \in \mathcal{P}_{X_0}^k} \hat{\mathcal{R}}_{[t_{k-1},t_k]}(P)$.

Proof: The choice of δ_t in i) implies that $(\epsilon/2\sqrt{n}) - (M/L)(e^{L\delta_t} - 1) > 0$. Consequently, for each $\delta_{x_0}^k$, we have that

$$\frac{M}{L}\left(e^{L\delta_t} - 1\right) + e^{Lt_k}\delta_{x_0}^k \leq \frac{\epsilon}{2\sqrt{n}}. \quad (12)$$

For any $x_P^* \in P \in \mathcal{P}_{X_0}^k$, since $\|f(x(t_{k-1}, x_P^*))\| \leq M$, it follows from (12) and Lemma 9 that

$$\forall\, x_0 \in P \qquad \forall\, t \in [t_{k-1}, t_k], \quad \|x(t, x_0) - x(t_{k-1}, x_P^*)\| \leq \frac{\epsilon}{2\sqrt{n}}. \quad (13)$$

Let $x_{P,k}^* = x(t_{k-1}, x_P^*)$. Then, (13) implies that

$$\mathcal{R}_{[t_{k-1},t_k]}(P) \subseteq B_{(\epsilon/2\sqrt{n})}(x_{P,k}^*). \quad (14)$$

Using a set of normal vectors C computed with our heuristic in Section IV-A together with the hyperrectangle directions, the flow-pipe segment is approximated by

$$\hat{\mathcal{R}}_{[t_{k-1},t_k]}(P) = S_{C'}^{\min}(\mathcal{R}_{[t_{k-1},t_k]}(P)) \quad (15)$$

where C' is defined as in Lemma 10. From (14) and (15), we have that $\hat{\mathcal{R}}_{[t_{k-1},t_k]}(P) \subseteq S_{C'}^{\min}(B_{\epsilon/2\sqrt{n}}(x_{P,k}^*))$. The result from Lemma 10 with the origin translated to $x_{P,k}^*$ implies that

$S^{\min}_{C'}(B_{\epsilon/2\sqrt{n}}(x^*_{P,k})) \subseteq B_{\epsilon/2}(x^*_{P,k})$. In summary, we have that

$$x^*_{P,k} \in \mathcal{R}_{[t_{k-1},t_k]}(P) \subseteq \hat{\mathcal{R}}_{[t_{k-1},t_k]}(P) \subseteq S^{\min}_{C'}(B_{(\epsilon/2\sqrt{n})}(x^*_{P,k})) \subseteq B_{\epsilon/2}(x^*_{P,k}). \quad (16)$$

By Lemma 4, we have $\mathrm{dist}(\hat{\mathcal{R}}_{[t_{k-1},t_k]}(P), \mathcal{R}_{[t_{k-1},t_k]}(P)) \leq \epsilon$. Since the approximation error is within ϵ for each time interval $[t_{k-1}, t_k]$ and each initial subset $P \in \mathcal{P}^k_{X_0}$, we conclude by Lemma 5 that the proposition holds. ∎

The objective of the previous proposition is to demonstrate that in principle the flow pipe for a Lipschitz system can be approximated arbitrarily closely. The flow-pipe construction outlined in the proof of the proposition may be used to compute the flow-pipe approximation provided that the constants L and M are known. To obtain these constants, one may need to resort to global optimization techniques, since the optimization problem may be nonlinear and nonconvex in general.

Although Proposition 4 shows the flow-pipe approximation can be made arbitrarily tight, it is often sufficient to know what the approximation error is for a given time segment and set of initial states. The following proposition gives a bound on this approximation error.

Proposition 5: Given a set P and an interval $[t, t+\delta_t]$, let $\delta_{x_0} = \|P\|$. Then

$$\mathrm{dist}(\hat{\mathcal{R}}_{[t,t+\delta_t]}(P), \mathcal{R}_{[t,t+\delta_t]}(P)) \leq \epsilon$$

where $\epsilon = 2\sqrt{n}\left((\|f(x(t,x^*_0))\|/L)(e^{L\delta_t} - 1) + e^{L(t+\delta_t)}\delta_{x_0}\right)$ for any $x^*_0 \in P$.

Proof: By Lemma 9, we have that $\forall\ x_0 \in P, \forall\ t' \in [t, t+\delta_t]$, $\|x(t', x_0) - x(t, x^*_0)\| \leq (\epsilon/2\sqrt{n})$, which implies that $\mathcal{R}_{[t,t+\delta_t]}(P) \subseteq B_{(\epsilon/2\sqrt{n})}(x(t, x^*_0))$. By a similar argument to the one that leads from (14)–(16) in the proof of Proposition 4, we have that $x^*_0 \in \mathcal{R}_{[t,t+\delta_t]}(P) \subseteq \hat{\mathcal{R}}_{[t,t+\delta_t]}(P) \subseteq B_{(\epsilon/2)}(x(t, x^*_0))$. We conclude by Lemma 4 that the proposition holds. ∎

We note that the bound given in Proposition 5 can be computed by simulating the system beginning at *any* initial state x^*_0 in the set P to find $x(t, x^*_0)$.

D. Global Optima in the Flow-Pipe Approximations

The approximation obtained from the flow-pipe approximation procedure is an outer approximation only if the optimization software provides the global solution to (3). Since (3) is not a convex problem in general, there may be multiple local maxima. To guarantee a global maximum is found, one needs to resort to a global optimization method.

Consider an optimization problem $\max_{z \in F} J(z)$, where $J(\cdot)$ is a given objective function and F is a compact set in R^n. Let z^* denote a global solution to the optimization problem and let $J^* = J(z^*)$. General global optimization methods called *bounding methods* [28] rely on the ability to compute the bounds on the objective function for any compact subset of F. Each method starts with a partition $\mathcal{F}$ of the feasible set F and computes, for each compact subset $S \in \mathcal{F}$, the upper and lower bounds on the objective function, denoted $J_{\max}(S)$ and $J_{\min}(S)$, respectively, such that $J_{\min}(S) \leq J(z) \leq J_{\max}(S)$, for all $z \in S$. The upper and lower bounds on the global maximum J^*, denoted $J^*_{\max}$ and $J^*_{\min}$, can be established by computing the maximum over all upper and lower bounds for all subsets $S \in \mathcal{F}$, that is, $J^*_{\max} = \max_{S \in \mathcal{F}} J_{\max}(S)$ and $J^*_{\min} = \max_{S \in \mathcal{F}} J_{\min}(S)$. It is clear that $J^*_{\min} \leq J^* \leq J^*_{\max}$ and, thus, any subset S for which $J_{\max}(S) \leq J^*_{\min}$ is *rejected*, since it cannot possibly contain the global solution. The remaining subsets are refined further and the bounds are recomputed. The rejection and refinement process continues until the difference between $J^*_{\max}$ and $J^*_{\min}$ lies within some error tolerance $\epsilon > 0$, i.e., $J^*_{\max} - J^*_{\min} \leq \epsilon$.

For our objective function $J(t, x_0) = c_i^T x(t, x_0)$ in (3), we do not have an explicit formula for J since there is no closed-form solution for $x(t, x_0)$ in general. Nevertheless, we can compute the bound on the objective function for a subset $P \subseteq X_0$ and a time interval $[t, t+\delta_t]$ using the Lipschitz constant L for the vector field $f(x)$ as follows. Let $\delta_{x_0} = \|P\|$. By Lemma 9, we have that

$$\forall\ x_0 \in P \qquad \forall\ t' \in [t, t+\delta_t],\ \|x(t', x_0) - x(t, x^*_0)\| \leq \gamma$$

for some $x^*_0 \in P$ where $\gamma = (\|f(x(t, x^*_0))\|/L)\left(e^{L\delta_t} - 1\right) + e^{L(t+\delta_t)}\delta_{x_0}$. Thus, all trajectories from P during time interval $[t, t+\delta_t]$ are contained in the γ-ball centered at $x(t, x^*_0)$. Assuming that the face normal vector c_i is of unit length, the maximum and the minimum values of the objective function are bounded by $c_i^T x(t, x^*_0) \pm \gamma$.

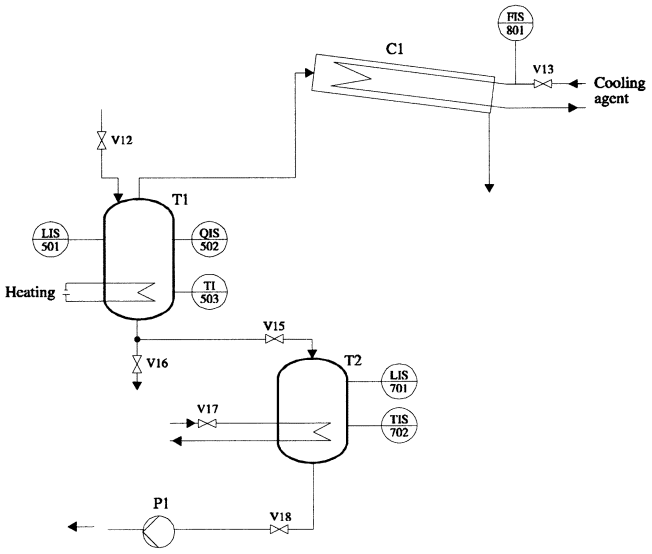

Fig. 5. Batch evaporation system.

TABLE I
INPUT CONFIGURATIONS FOR THE BATCH EVAPORATOR SYSTEM

Config.	Heat	V_{15}	V_{18}	Description
u_1	on	closed	open	heating T_1
u_2	off	closed	open	cool T_1/drain T_2
u_3	off	open	closed	cool/drain T_1

V. EXAMPLE: VERIFICATION OF A BATCH EVAPORATOR

We consider the verification problem for a batch evaporator example presented in [12]. The evaporation system is shown in Fig. 5. The controller is designed to implement the following production sequence. First, tank T_1 is filled with a solution which is evaporated until a desired concentration is reached.

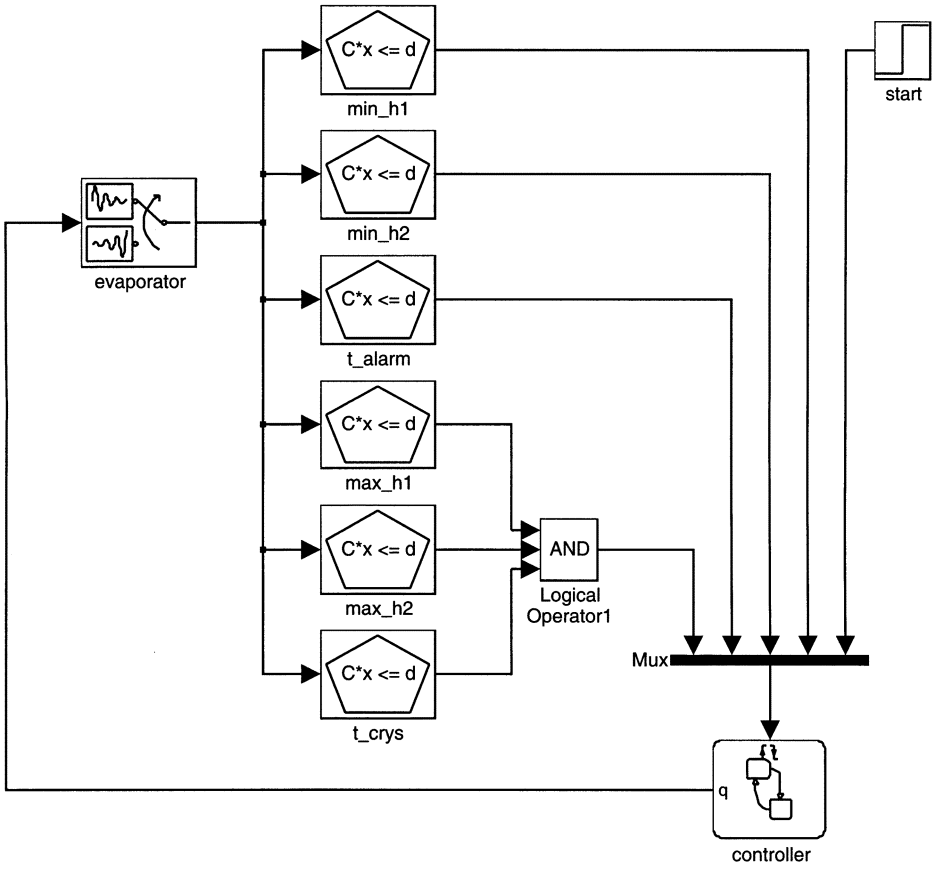

Fig. 6. Simulink block diagram for the batch evaporator system.

Tank T_1 is then drained as soon as tank T_2 is emptied from the previous batch. For safety reasons, the heating is shut off when the alarm temperature, T_{alarm}, is reached. When the temperature in tank T_1 falls below a certain temperature T_{crys}, crystallization will occur and spoil the batch. Our objective is to verify that the alarm temperature is chosen appropriately such that from a given set of initial conditions the temperature in tank T_1 never falls below the crystallization temperature before T_1 is completely drained.

The control inputs to the system are the states of the heater (on/off) and valves V_{15} and V_{18} (open/closed). A given set of values for the three control inputs is referred to as an *input configuration*, denoted by the discrete variable u. Table I lists the three input configurations used by the controller.

The continuous state variables are the heights of the liquid in tanks T_1 and T_2, denoted H_1 and H_2, and the temperature in tank T_1, denoted T. The continuous dynamics depends on the input configuration. Let

$$\alpha(T) = -1.327 \cdot 10^{9} T^{-2} + 2.819 \cdot 10^{6} T^{-1} + 6.433 \cdot 10^{3} - 10.513 T.$$

For configuration u_1, the state equations are

$$\begin{aligned} \dot{H}_1 &= 0 \\ \dot{H}_2 &= -3.333 \cdot 10^{-4} \sqrt{19.62 H_2} \\ \dot{T} &= \frac{5000 - 24(T - 283)}{1.23 \cdot 10^{5} H_1 + \alpha(T)}. \end{aligned}$$

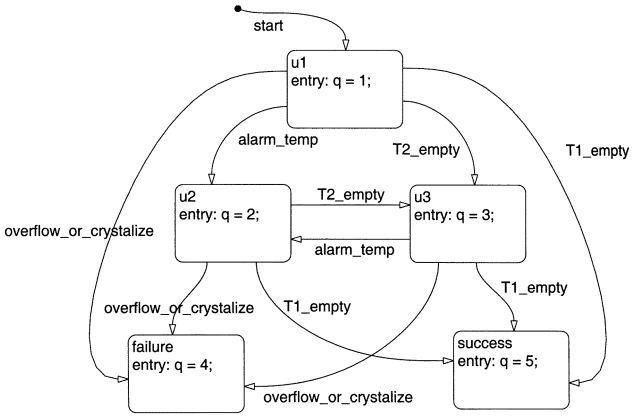

Fig. 7. Stateflow diagram for the block *controller* in Fig. 6.

For u_2, the state equations are

$$\begin{aligned} \dot{H}_1 &= 0 \\ \dot{H}_2 &= -3.333 \cdot 10^{-4} \sqrt{19.62 H_2} \\ \dot{T} &= \frac{-24(T - 283)}{1.23 \cdot 10^{5} H_1 + step(T - 373) \cdot \alpha(T)} \end{aligned}$$

where $step(\cdot)$ denotes the standard step function. Finally, the state equations for u_3 are

$$\begin{aligned} \dot{H}_1 &= -6.667 \cdot 10^{-4} \sqrt{19.62 H_1} \\ \dot{H}_2 &= 3.333 \cdot 10^{-4} \sqrt{19.62 H_1} \\ \dot{T} &= \frac{-0.036(T - 283)(0.03 + 0.628 H_1)}{29.1 H_1}. \end{aligned}$$

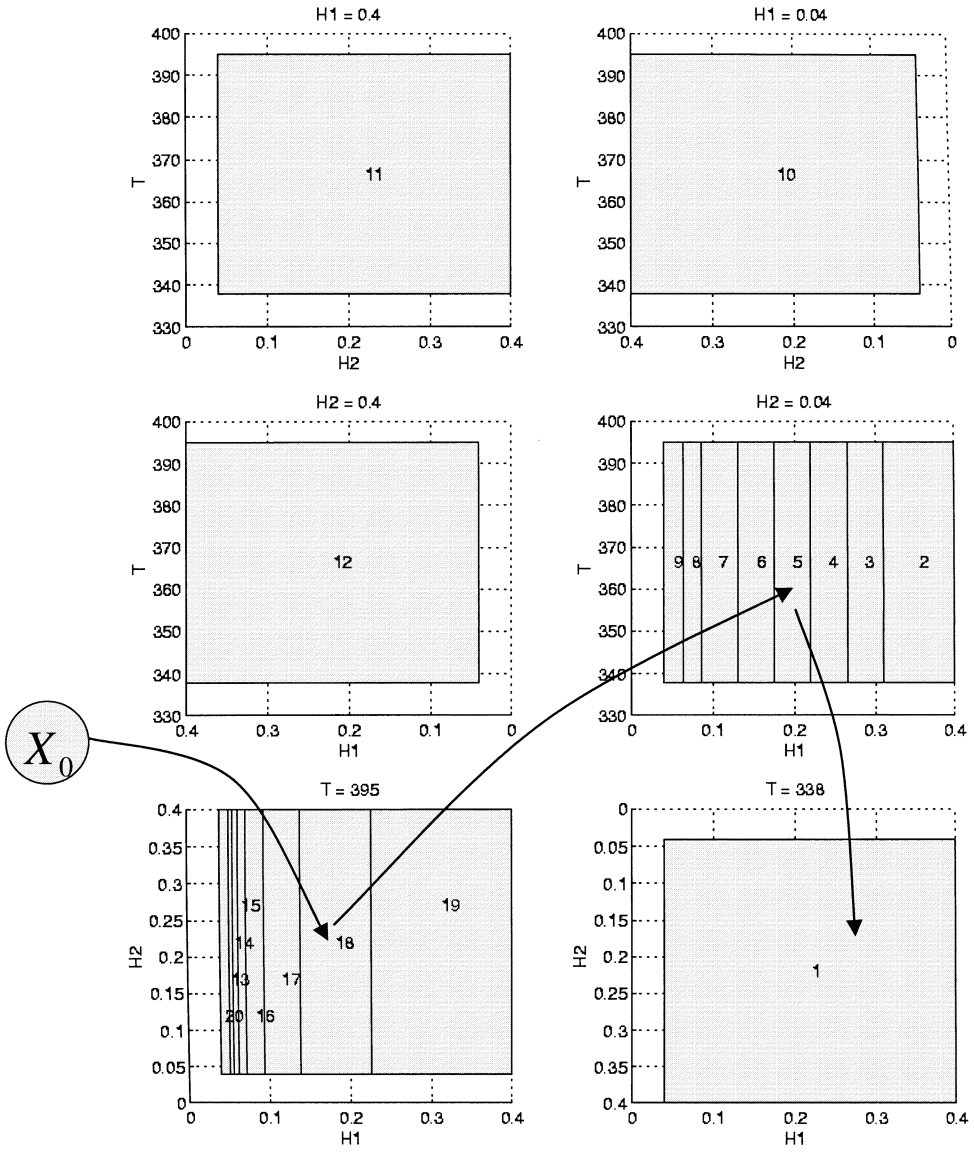

Fig. 8. Quotient system $T/\mathcal{P}_0$.

Figs. 6 and 7 show the Simulink and Stateflow diagrams in *CheckMate* corresponding to the production sequence described above. The discrete states u_1, u_2, u_3 correspond directly to the input configurations u_1, u_2, u_3. Discrete states u_4 and u_5 are used to indicate the *failure* and *success* of the production sequence, respectively.

The system starts with the discrete state u_1 and the continuous states $H_1 \in [0.2, 0.22]$ m, $H_2 \in [0.28, 0.3]$ m, and $T = 373$ K. Since the liquid level in each tank in the ODE model can only reach zero asymptotically, we approximate the event that a tank is empty by small thresholds $H_{i\min}, i = 1, 2$. The numerical values for the thresholds in the system are

$$\begin{aligned} H_{i\min} &= 0.04 \text{ m (tank empty)} \\ H_{i\max} &= 0.4 \text{ m (tank overflow)} \\ T_{\text{crys}} &= 338 \text{ K (crystallize) and} \\ T_{\text{alarm}} &= 395 \text{ K (alarm).} \end{aligned}$$

The problem is to verify that all trajectories from the initial continuous state set $X_0 = \{0.2 \leq H_1 \leq 0.22, 0.28 \leq H_2 \leq 0.3, T = 373\}$ and the initial discrete state u_1 eventually reach the discrete state u_5.

CheckMate constructs the initial AQTS from the partition of the threshold hyperplanes shown in Fig. 8, where the state X_0 represents the set of initial continuous states. Each continuous subset in the partition is referred to as a patch. For this partition, the AQTS does not satisfy the specification because the crystallization temperature is reachable, as indicated in the figure. After three iterations of the verification procedure, we have the partition in Fig. 9 that satisfies the specification. For this partition, all paths from X_0 eventually reach the empty threshold for Tank 1 without reaching the overflow and crystallization threshold. Further details about this example are given in [29].

VI. Discussion

This paper presents computational methods for constructing finite-state approximations, called AQTSs, for a class of hybrid systems to verify properties of the hybrid system behaviors. Representing and computing the flow pipes for continuous dynamic sys-

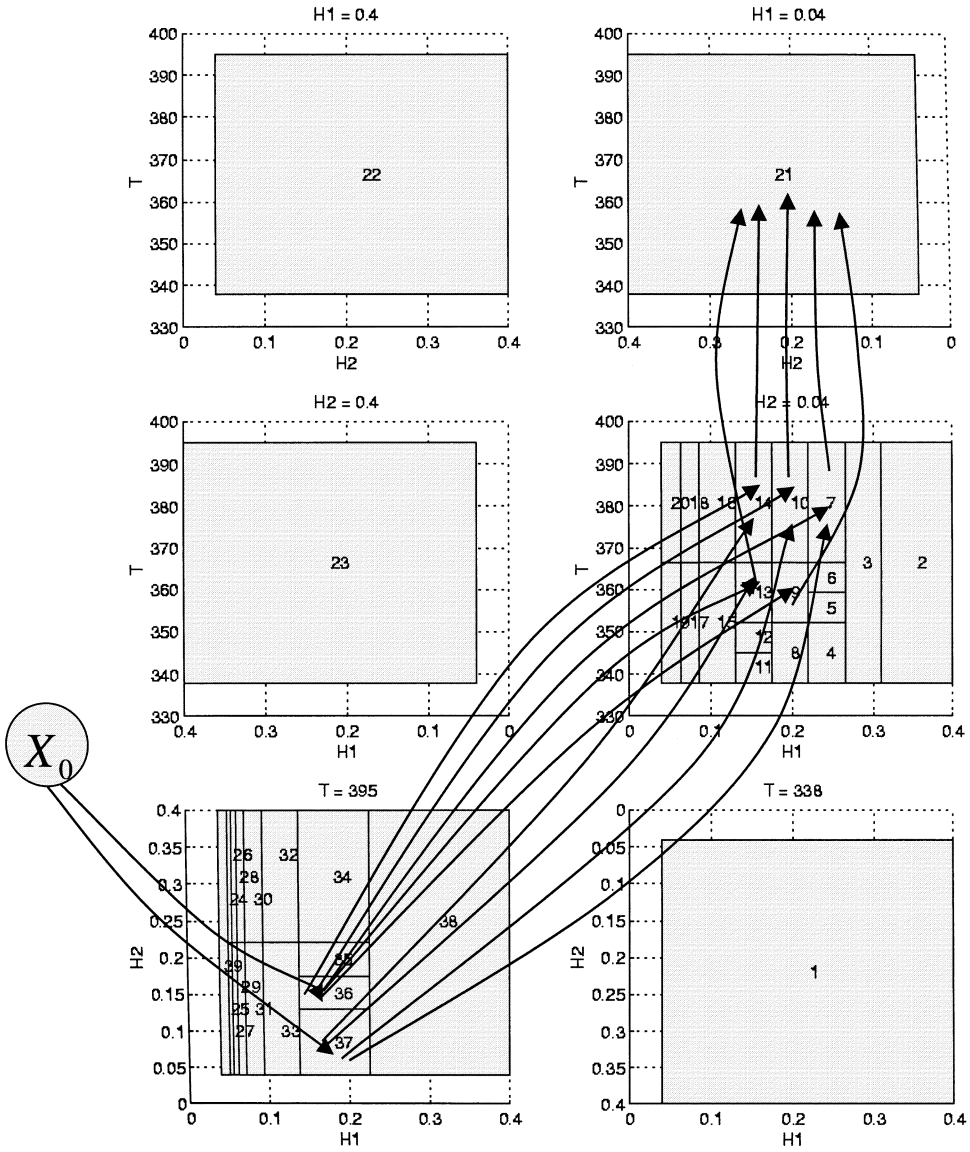

Fig. 9. Quotient system $T/\mathcal{P}_3$.

tems is the fundamental problem in constructing the AQTS. We propose a method for constructing flow-pipe approximations as the union of convex polyhedra. We show that proposed flow-pipe approximations can be made arbitrarily accurate for general nonlinear systems. We also present extensions and new results on efficient flow-pipe computations for affine systems.

To guarantee the flow-pipe approximation is conservative, we show that, in principle, we can use a global optimization method to compute the flow-pipe approximations. Implementation of the proposed global optimization remains a topic for future research, however. Experiments with global optimization routines are needed to assess the tradeoffs between computational cost and the guarantees provided by the global optimization.

The study of hybrid systems has stimulated considerable interest in the problem of representing and computing sets of reachable states for continuous dynamic systems. Alternatives to the approach proposed in this paper include: grid-based discretizations of the continuous state space, which can be automated quite easily and robustly, but can lead to enormous finite-state approximations [9], [30]; ellipsoidal approximations of reachable sets (at a given time), for which exact analytical expressions are available for linear dynamic systems [31]; orthogonal polyhedra, for which there are efficient canonical representations and computational procedures for general nonlinear dynamics [32], [33]; interval arithmetic to compute conservative approximations to differential inclusions [34]; computing conservative projects of reachable sets onto lower-dimensional subspaces [35]; and dynamic programming (solving the Hamilton–Jacobi–Bellmen equation), which leads to explicit analytical representations of reachable sets (at a given time) for certain linear systems with bounded inputs [36]. Comparisons and refinements of the methods proposed thus far, including the approach proposed in this paper, are required to assess which approaches are best. It is likely that no single approach will be best for all situations. Future computational tools for hybrid systems should probably be "hybrid", incorporating multiple methods and techniques for reachability computations so that the best approach can be used for each application.

REFERENCES

[1] 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.

[2] T. A. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annual Symp. Logic Computer Science*, 1996, pp. 278–292.

[3] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA: MIT Press, 2000.

[4] R. P. Kurshan, *Computer-Aided Verifiation of Coordinating Processes*. Princeton, NJ: Princeton Univ. Press, 1995.

[5] A. Chutinan and B. H. Krogh, "Infinite-state transition system verification using approximate quotient transition systems," *IEEE Trans. Automat. Contr.*, vol. 46, pp. 1401–1410, Sept. 2001.

[6] F. Zhao, "Automatic analysis and synthesis of controllers for dynamical systems based on phase-space knowledge," Ph.D. dissertation, Art. Intell. Lab., Mass. Inst. Technol., Cambridge, MA, 1992.

[7] A. Chutinan and B. H. Krogh, "Computing polyhedral approximations to dynamic flow pipes," presented at the 37th IEEE Conf. Decision Control, 1998.

[8] E. K. Kornoushenko, "Finite-automation approximation to the behavior of continuous plants," *Automat. Rem. Control*, vol. 36, no. 12, pp. 2068–2074, 1975.

[9] A. Puri, P. Varaiya, and V. Borkar, "ϵ-approximation of differential inclusions," in *Hybrid Systems III: Verification and Control*, R. Alur, T. A. Henzinger, and E. D. Sontag, Eds. New York: Springer-Verlag, 1996, pp. 362–376.

[10] O. Stursberg, S. Kowalewski, and S. Engell, "On the generation of timed discrete approximations for continuous systems," *Math. Model. Syst.: Special Issue Discrete-Event Models Contin. Syst.*, vol. 6, no. 1, pp. 51–70, 2000.

[11] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Algorithmic analysis of nonlinear hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 43, pp. 540–554, Apr. 1998.

[12] S. Kowalewski and O. Stursberg, "The batch evaporator: A benchmark example for safety analysis of processing systems under logic control," presented at the 4th Int. Workshop Discrete Event Systems (WODES '98), Cagliari, Italy, Aug. 1998.

[13] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal—a tool suite for automatic verification of real-time systems," in *Hybrid Systems III*, vol. 1066, LNCS. New York, pp. 232–243.

[14] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," *Software Tools Technol. Transfer*, vol. 1, no. 1/2, pp. 110–122, 1997.

[15] E. Closse, M. Poize, J. Pulou, J. Sifakis, D. Weil, and S. Yovine, "Taxys = esterel + kronos: A tool for developing and verifying embedded real-time systems," presented at the IEEE Conf. Decision Control, Orlando, FL, 2001.

[16] O. Botchkarev and S. Tripakis, "Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations," in *Hybrid Systems: Computation and Control*, vol. 1790, Lecture Notes in Computer Science. New York, 2000, pp. 73–88.

[17] E. Asarin, T. Dang, and O. Maler, "d/dt: A verification tool for hybrid systems," presented at the IEEE Conf. Decision Control, Orlando, FL, Dec. 2001.

[18] A. Bemporad and F. D. Torrisi, "Discrete-time hybrid modeling and verification," presented at the IEEE Conf. Decision Control, Orlando, FL, Dec 2001.

[19] I. Silva and B. H. Krogh, "Formal verification of hybrid systems using CheckMate: A case study," presented at the 2000 Amer. Control Conf., June 2000.

[20] *Simulink: Dynamic System Simulation for MATLAB, Using Simulink Version 2.2*, The MathWorks, Inc., Natick, MA, 1998.

[21] *Stateflow: For Use With Simulink, User's Guide Version 1*, The MathWorks, Inc., Natick, MA, 1998.

[22] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?," *J. Comput. Sci.*, vol. 57, no. 1, pp. 94–124, 1998.

[23] G. Lafferriere, G. J. Pappas, and S. Yovine, "A new class of decidable hybrid systems," in *Proc. Hybrid Systems: Computation Control, 2nd Int. Workshop, HSCC'99*, F. W. Vaandrager and J. H. Van Schuppen, Eds., 1999, pp. 137–151.

[24] T. A. Henzinger, "Hybrid automata with finite bisimulations," in *ICALP 95: Automata, Languages, and Programming*, Z. Fülöp and F. Gécseg, Eds. New York: Springer-Verlag, 1995, vol. 944, Lecture Notes in Computer Science, pp. 324–335.

[25] J. S. Miller, "Decidability and complexity results for timed automata and semi-linear hybrid automata," in *Proc. Hybrid Systems: Computation Control HSCC'2000*, vol. 1790, LNCS, N. Lynch and B. H. Krogh, Eds., 2000, pp. 296–309.

[26] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[27] H. K. Khalil, *Nonlinear Systems*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1996.

[28] A. P. Leclerc, "Efficient and reliable global optimization," Ph.D. dissertation, The Ohio State Univ., Columbus, OH, 1992.

[29] A. Chutinan and B. H. Krogh, "Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations," in *Proc. Hybrid Systems: Computation Control, 2nd Int. Workshop, HSCC'99*, F. W. Vaandrager and J. H. Van Schuppen, Eds., 1999, pp. 76–90.

[30] J. Preußig, O. Stursberg, and S. Kowalewski, "Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis," in *Proc. Hybrid Systems: Computation Control, 2nd Int. Workshop, HSCC'99*, F. W. Vaandrager and J. H. Van Schuppen, Eds., 1999, pp. 209–222.

[31] A.B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Proc. Hybrid Systems: Computation Control HSCC'00*, vol. 1790, LNCS, N. Lynch and B. H. Krogh, Eds., 2000, pp. 202–214.

[32] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Proc. Hybrid Systems: Computation Control HSCC'98*, Apr. 13–15, 1998, pp. 96–109.

[33] O. Bournez, O. Maler, and A. Pnueli, "Orthogonal polyhedra: Representation and computation," in *Proc. Hybrid Systems: Computation Control, 2nd Int. Workshop, HSCC'YY*, F. W. Vaandrager and J. H. Van Schuppen, Eds., 1999, pp. 46–60.

[34] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Won'g-Toi, "Beyond hytech: Hybrid systems analysis using interval numerical methods," in *Proc. Hybrid Systems AI: Modeling Analysis Control Discrete Plus Continuous Systems: Papers 1999 AAAI Symp. (TR SS-99-05)*, Mar. 1999, pp. 89–95.

[35] M. R. Greenstreet and I. Mitchell, "Chapter reachability analysis using polygonal projections," in *Hybrid Systems: Computation and Control*, vol. 1569, LNCS. New York, 1999, pp. 103–116.

[36] A. B. Kurzhanski and P. Varaiya, "Dynamic optimization for reachability problems," *J. Optim. Theory Applications*, vol. 108, no. 2, pp. 227–51, Feb. 2001.

Alongkrit Chutinan received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1995, 1996, and 1999, respectively. His Ph.D. research topic was in the area of dynamical systems and control with emphasis on *hybrid systems verification*.

After graduation, he joined Ford Research Laboratory, Dearborn, MI, where he developed and applied computer tools to model-based analysis and design of automotive powertrain systems. In 2000, he joined Emmeskay, Inc., Plymouth, MI, as a Software Developer, developing MATLAB-based computational tools for clients in the automotive industry. He is currently a Lecturer for the Undergraduate Computer Science Program at Shinawatra University, Pathumthani, Thailand.

Bruce H. Krogh (S'82–M'82–SM'92–F'98) received the B.S. degree in mathematics and physics from Wheaton College, Wheaton, IL, and the Ph.D. degree in electrical engineering from the University of Illinois, Urbana, in 1975 and 1983, respectively.

He joined Carnegie Mellon University, Pittsburgh, PA, in 1983, where he is currently a Professor of electrical and computer engineering. His research interests include synthesis and verification of control algorithms and software for discrete event and hybrid dynamic systems.

Dr. Krogh was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.