Verifying Switched-Mode Computer Controlled Systems [1]

James Kapinski and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
jpk3@andrew.cmu.edu/krogh@ece.cmu.edu

Abstract

This paper presents a new approach to verifying properties of computer control systems with periodic sampling when the control programs include both continuous-variable computations and discrete-state transitions (mode switching). We focus on the novel aspects of the approach, which are (1) a formal model for hybrid systems with continuous-time and discrete-time behaviors and (2) a method for computing conservative approximations to the sets of reachable states. The formal model and computational routines are incorporated into *CheckMate*, a MATLAB-based tool for verification of properties of hybrid dynamic systems. This tool is being applied to the verification of embedded controllers in automotive engines.

1 Introduction

A hybrid system is one that contains both continuous and discrete dynamics. For an introduction to the theory of hybrid systems see van der Schaft and Schumacher[1]. Recently, there has been considerable interest in developing and applying tools for hybrid system verification to embedded control systems [2, 3]. In contrast to simulation studies that characterize the system behavior for particular initial conditions and parameter values, verification tools make it possible to analyze the system behavior for full ranges of parameter values. This is particularly useful when it is not easy to identify what parameter values will lead to worst-case behaviors.

Current tools for hybrid system verification apply best to situations where the sampling rate of the controller is fast enough so that sampling can be neglected in the analysis [4]. The plant and controller dynamics are combined into a single continuous-time system. To evaluate the effects of sampling, a clock variable needs to be included to model the sampling events explicitly. Recently, Silva introduced the concept of sampled-data hybrid automata to capture the effects of periodic sampling more efficiently, both in the model and the verification computations [5]. In that work the sampled behavior applies only to conditions for discrete-state transitions, and the continuous aspects of the controller must be incorporated into the continuous-time (unsampled) portion of the hybrid model. The objective of the work reported in this paper is to extend formal verification techniques to sampled-data systems that include both continuous-state and discrete-state computations in the controller.

The following section describes the types of computer control systems considered in this paper. Section 3 presents the hybrid automaton with continuous-time and discrete-time dynamics (HACDD), a formal model for sampled-data systems with mode switching. Section 4 describes verification of HACDD's using the tool *CheckMate*. Section 5 presents our method for computing conservative polyhedral approximations to the sets of reachable states for continuous dynamic systems. Section 6 describes an automotive example on which the new method has been applied. The concluding section summarizes the contributions of this paper.

2 Computer-Controlled Systems

Figure 1 illustrates the type of systems considered in this paper. The plant is a continuous-time process with state x and output y that is sampled periodically. The control computations are modeled as a set of difference equations that update the continuous-valued controller state z (herein referred to as the controller state). The controller state equations are determined by the controller discrete-valued state q (herein called the controller mode), which makes transitions based on specified conditions. The controller output u, a function of the controller state and the sampled plant output, is applied to the plant as a piecewise-constant signal from a zero-order hold.

We assume the controller computations are ordered as follows: (1) the plant output is sampled; (2) the mode is updated according to the mode-switching logic; (3)

[1] This research was supported in part by Ford, the US Defense Advance Projects Research Agency (DARPA) contract no. F33615-00-C-1701, US Army Research Office (ARO) contract no. DAAD19-01-1-0485, and the US National Science Foundation (NSF) contract no. CCR-0121547.

if the mode switches, the controller state may also be reset; (4) the controller state and control output are computed, based on the updated mode. The computations of most single-rate sampled-date control systems can be organized into this sequence with appropriate definitions of the control mode and state variables.

The novel aspect of this model with respect to formal verification is the inclusion of continuous variables in the controller. Typically, the controller model in hybrid automata includes only discrete-state transition logic (see, e.g., [6]).

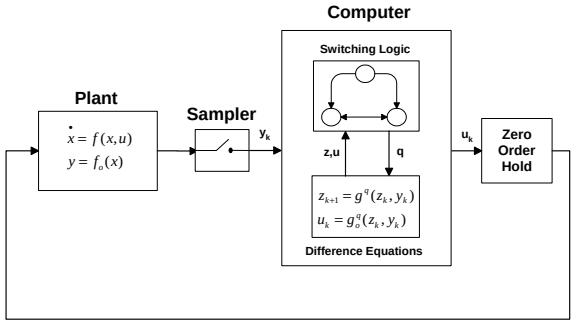

Figure 1: Sampled-Data Computer Controlled System

3 A Formal Model

Formal verification requires a formal model of the system dynamics with a well-defined semantics. For this purpose we introduce *hybrid automata with continuous-time and discrete-time dynamics* (HACDD), as a modification of standard hybrid automata [7]. The state of an HACDD is given by the mode, $q \in Q$, and the continuous state variables, $(x, z) \in X \times Z$. The initial mode is given by $q_o \in Q$, and the set of possible initial continuous states is given by $X_o \subseteq X$ and $Z_o \subseteq Z$. Figure 2 illustrates the following components of an HACDD:

Modes: These discrete states, represented by circles in figure 2, correspond to the logical states in the controller. The following are associated with each mode:

- *Difference Equations*: $z_{k+1} = g^q(z_k, x_k)$, determining the value of the controller state before the next sampling time. The plant output, $y = f_o(x)$ in figure 1, is absorbed into the controller difference equation, and $x_k = x(kT)$ is the sampled value of the process state.
- *Differential Equations*: A set of differential equations that governs the continuous-time dynamics, denoted as $\dot{x} = f^q(x, z_k)$. The controller output, shown as $u = g_o^q(z_k, x_k)$ in figure 1, along with the zero-order-hold operation are absorbed into the plant differential equation. The differential equations are mode dependent to reflect the dependence of the control output on the mode.

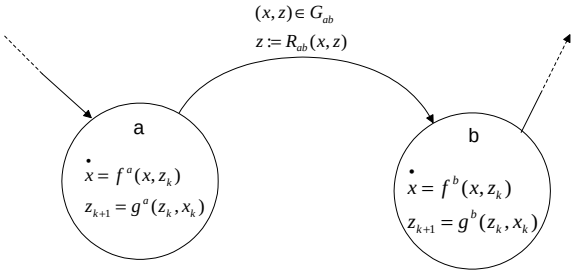

Figure 2: HACDD Example

Transitions: The mode-transitions in the controller, represented by arrows in Figure 2, possess the following elements:

- *Guards*: These are regions of the system continuous state space that determine when a transition is taken. G_{ab} denotes the guard region associated with the transition from mode $q = a$ to mode $q = b$.
- *Resets*: $R_{ab}(x, z)$ denotes the reset of the controller state when a transition from mode $q = a$ to mode $q = b$ occurs, where z is the state of the controller before the transition from mode $q = a$.

We define the *interior region* for each mode to be all points within the state space where no guard region is present.

The sampling of the plant state and the update of the controller state are performed at each sampling instant with a specified sampling period T.

The behaviors of an HACDD are described by the notion of *trajectories*.

Definition 1 *Given an HACDD H, a* trajectory *is defined as a discrete-time sequence (x_k, z_k, q_k) that satisfies the following for all $k \in \{0, 1, \ldots\}$:*

- $x_0 \in X_o$, $z_0 \in Z_o$, $q_0 = q_o$
- $z_1 = g^{q_o}(z_0, x_0)$
- q_0 *is the initial mode*
- *There exists a function of time $x(t)$ where $x(kT) = x_k$, $x((k+1)T) = x_{k+1}$, and $\dot{x} = f^{q_k}(x, z_k)$ for $kT < t < (k+1)T$*
- *If $q_{k+1} \neq q_k$ then:*

- $(x_{k+1}, z_{k+1}) \in G_{q_k q_{k+1}}$
- $q_k q_{k+1}$ *is a valid transition*
- $z_{k+2} = g^{q_{k+1}}(R_{q_k q_{k+1}}(z_{k+1}), x_{k+1})$

- *If* $q_{k+1} = q_k$ *then:*
  - $(x_{k+1}, z_{k+1}) \notin G_{q_k q'} \; \forall \; q' \neq q_k$
  - $z_{k+2} = g^{q_k}(z_{k+1}, x_{k+1})$

Notice that definition 1 requires that the system remain in any mode that it enters for at least one sample period, even the first location. This means that there can be no mode transition at the first instant.

Given an HACDD H, we define the *reachable set* to be the set of all states that are reachable by a trajectory of H.

4 CheckMate: A Tool for Verification

The purpose of verification, in general, is to determine if all possible behaviors of a system satisfy some property, which is often specified in a temporal logic such as Computation Tree Logic (CTL)[8]. The CheckMate tool uses the procedure illustrated in figure 3 to verify properties of hybrid systems[9]. Basically, the approach consists of building a finite state machine (FSM) that conservatively represents all behaviors of the original system, and then performing model checking on the FSM.

CheckMate allows the designer to build a representation of the system suitable for simulation using MATLAB's Simulink/Stateflow front-end. This representation, corresponds to the *Simulation Diagram* in figure 3. In order to analyze the system, the Simulink model is transformed into an HACDD with convex guard and interior regions.

The next step involves dividing the interior regions for each mode of the HACDD into convex, polyhedral regions. The dotted lines in figure 4 illustrate the partitioning for one mode of the HACDD in figure 2. This partitioned version of the system constitutes the *Partitioned HACDD* in figure 3. These partition elements will represent states in the FSM that is constructed.

In order to construct the FSM, the relevant connectivity between the partition elements must be determined. In other words, the following must be answered: Given that the system occupies some point inside a region, to which other modes can the state evolve[10]? This question is answered in the next step, the *Perform Reachability* step in figure 3, which is by far the most computationally expensive step. Section 5 discusses this process in detail.

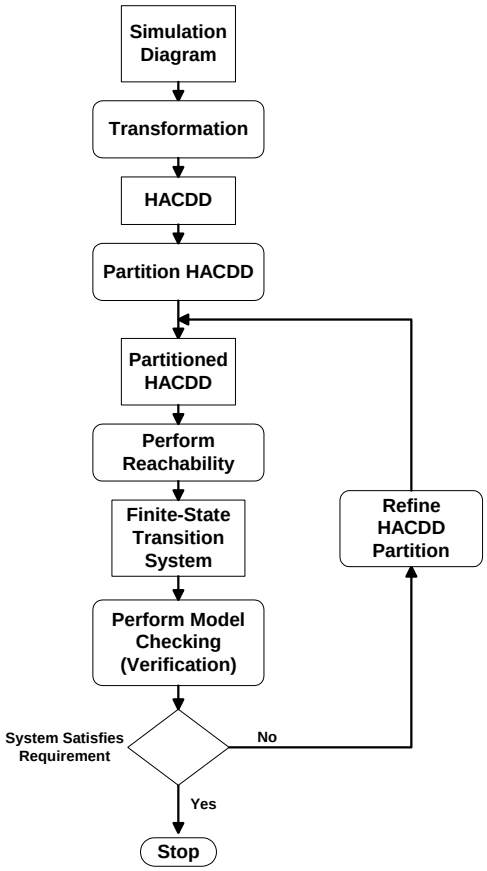

Figure 3: CheckMate Verification Procedure

Once the connectivity between partitioned elements has been established, the FSM is be constructed, and then model checking is performed.

CheckMate performs verification for a restricted class of CTL expressions called ACTL[11], which allows only universal quantification, meaning that only properties that test all possible behavior paths may be expressed. The model checking process will yield a positive or negative response for a given ACTL expression. A positive response indicates that the given ACTL expression is true. In this case, no further analysis needs to be performed. If the model checker yields a negative

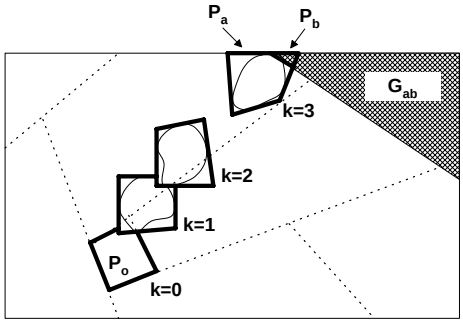

Figure 4: Example of an HACDD state-space and DTFA

response, this means that the ACTL expression does not hold for the transition system. This does not necessarily mean that the specification does not hold for the original system, since the partitioning and reachability analysis provide a conservative approximation. In this case, another attempt can be made by refining the HACDD partition, yielding a less conservative approximation. Then the reachability, transition system creation, and model checking steps can be performed again. This process can continue until either the specification is satisfied or the process is terminated by the user.

5 Reachability Analysis for the HACDD

The partition elements for each mode, as described in section 4, constitute states in the finite state machine that is created during the verification procedure. Reachability analysis is the process of identifying connectivity between these states by overapproximating the set of reachable states.

Chutinan and Krogh developed a method, called flow pipe approximation, to overapproximate reachable sets of states for a class of continuous-time hybrid systems, given a set of starting states[12]. We propose a method similar to Chutinan and Krogh's, called *discrete-time flow analysis* (DTFA), that considers the discrete-time behavior of the difference equations to overapproximate the set reachable set.

An example of DTFA is illustrated in figure 4. Here, $P_o = X_o \times Z_o$ is one of the partition elements, and portions of the reachable set are shown as the regions inside the heavy solid lines. Polyhedra that overapproximate the reachable set are shown as the heavy solid lines themselves. It is the goal of DTFA to perform this overapproximation.

Once an approximation of the reachable set for P_o has been calculated, all modes that intersect the reachable set via guard regions are considered connected to P_o. Each *entry region* (i.e. regions of the partition where the mode is entered after a reset into that region) constitutes a state in the transitions system that is created, and the connectivity identified by the reachability analysis represents the transitions between the states.

The DTFA method conservatively approximates the evolution of a polyhedral region due to the dynamical equations that govern the HACDD. To this end, the set $Reach_T(P, q)$, which is the set of states that are reachable over one sample period, is overapproximated.

Definition 2 *Given an HACDD H, a polyhedral region in its state space $P \subseteq X \times Z$, and a location q, the set $Reach_T(P, q)$ is defined as:*

$$\begin{aligned} Reach_T(P,q) \quad = \quad & \{(x^*, z^*) | x^* = \phi^q(T, x', z'), \\ & z^* = g^q(z', x') where (x', z') \in P\} \end{aligned}$$

where $\phi^q(T, x', z')$ denotes the solution to $\dot{x} = f^q(x, z')$ at time $t = T$ with initial condition x'.

The steps taken in approximating $Reach_T(P, q)$ are as follows:

1. Pass each vertex of P, (x_0, z_0) through the mapping $(x_1, z_1) = \left(\phi^{q_0}(T, x_0, z_0), g^{q_0}(z_0, x_0)\right)$
2. Perform a convex hull operation on the points found in the previous step. This will result in a new polyhedral set, P_C, which can be described by the inequality $C^T \begin{bmatrix} x \\ z \end{bmatrix} \leq d$ where C is a matrix whose columns are the normal vectors of the faces of P_C and d is a column vector.
3. Perform the following optimizations:

$$d_i^* = \max_{(x,z) \in P} \quad C_i^T \begin{bmatrix} \phi^q(T, x, z) \\ g^q(z, x) \end{bmatrix}$$

   where C_i^T is the i^{th} row of C^T.

The approximation to $Reach_T(P, q)$ is the polyhedron $C^T \begin{bmatrix} x \\ z \end{bmatrix} \leq d^*$.

Figures 5 and 6 illustrate the procedure. Portions of the actual reachable set lie outside the convex hull of the simulation points, in general. The optimization step pushes the faces of the convex hull out so that they contain the entire reachable set.

In order to calculate the entire reachable set, we must compute the evolution of the set of initial conditions P_o over time by iteratively calculating the set $Reach_T(P, q)$. The following gives a brief overview of the steps involved in accomplishing this task.

1. This is the initialization step. The initial set P_o is transformed by overapproximating the set $Reach_T(P_o, q_o)$ as described above.
2. For all q' such that $q_o q'$ is a valid transition, if $S_{q'} = Reach_T(P_o, q_o) \cap G_{q_o q'} \neq \emptyset$, then the $q_o q'$ transition must be taken. In this case, if $S_{q'} \neq Reach_T(P_o, q_o)$, then the analysis must consider both the $S_{q'}$ region, which satisfies the $q_o q'$ guard, and the $Reach_T(P_o, q_o) - S_{q'}$ region,

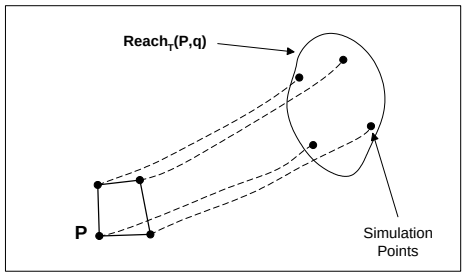

Figure 5: Step 1 of the procedure to approximate $Reach_T(P, q)$

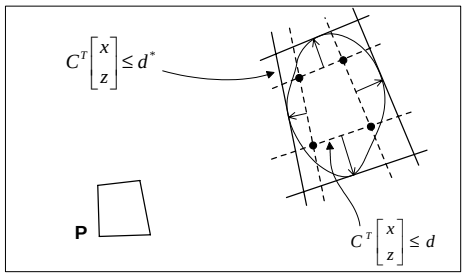

Figure 6: Steps 2 and 3 of the procedure to approximate $Reach_T(P, q)$

which does not. The $Reach_T(P_o, q_o) - S_{q'}$ region, if it does not satisfy any other guards, continues on to step four and is propagated forward using the dynamics governing the q_o mode. The $S_{q'}$ region continues on to step three. This division of the reachable set due to transitions necessitates the use of a queue in order to explore the reachability of each piece. The queue used in CheckMate performs a breadth first search of these sets. That is, all mode transitions occurring at one time instant are considered first, then the sample time advances one period to consider the next set of transitions.

3. The reset function is applied to $S_{q'}$ producing

$$R_{q_o q'}(S_{q'}) = \{(x, z) | (x, z) = R_{q_o q'}(x', z'), (x', z') \in S_{q'}\}$$

For each P_i, where P_i is an element of the q' partition, if $R_{q_o q'}(S_{q'}) \cap P_i \neq \emptyset$ then P_i becomes a new region to be propagated by the reachability (i.e. it is added to the queue) with the dynamics governing the evolution of P_i being those of the q' mode.

4. At this stage in the process, we are ready to calculate the next time step, and so we are in the same position as we were at the beginning of step 1. The procedure repeats steps 1 through 3, replacing q_o and P_o with the appropriate mode and point set, for some finite number of iterations, which is specified by the user. Again, for cases where the reachable set splits, each piece must be propagated separately.

Figures 4 and 7 illustrate the intersection/division event described in step 2 above. Figure 4 shows the approximation to the reachable set calculated at $k = 3$ divided into two pieces P_a and P_b. The P_b region corresponds to the portion of the reachable set that intersects the guard region G_{ab} while the P_a region corresponds to the portion of the reachable set at $k = 3$ that does not. Figure 7 shows P_b in terms of the b mode (i.e. the destination mode of the e_{ab} transition). The region labeled $R_{ab}(P_b)$ is the P_b region after the reset associated with the ab transition has been applied. Since $R_{ab}(P_b) \cap B_1 \neq \emptyset$, the partition region B_1 constitutes an entry region, and a transition is created in the finite state machine, which is created by the verification process (figure 3), from the state representing P_o to the state representing B_1.

With respect to the example shown in figures 4 and 7, the next step in the reachability analysis would be to propagate the $R_{ab}(P_b)$ region forward in the b mode and propagate the P_a region forward in the a mode.

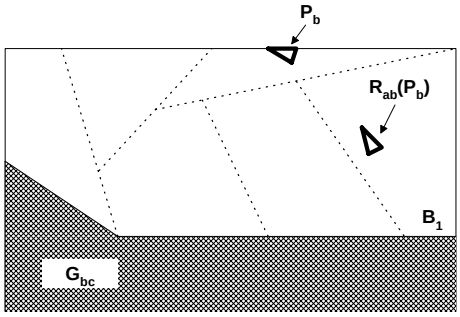

Figure 7: Reachable set after transition

6 Variable Cam Timing System Example

DTFA is being applied to the verification of a variable cam timing (VCT) system. The VCT system, suggested by Stefanopoulou et al.[13] and Butts[14], is an automotive application in which a computer controlled hydraulic actuator is used to vary the phase of the cam shaft of an engine with respect to its crankshaft. The controller's task is to regulate the position of the actuator to a given setpoint. The controller has three modes: one in which standard PID control is employed, one in which the controller output is saturated high, and one in which it is saturated low. The property to be verified is that no more than one mode transition should ever be taken.

The dynamics of the VCT system are represented by the HACDD shown in figure 8. The difference

equations for each mode are the same. They are as follows:

$$\begin{bmatrix} z^1_{k+1} \\ z^2_{k+1} \end{bmatrix} = \begin{bmatrix} .3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z^1_k \\ z^2_k \end{bmatrix} + \begin{bmatrix} -1 \\ -T \end{bmatrix} x_k + \begin{bmatrix} ref \\ Tref \end{bmatrix}$$

where z^1 and z^2 are controller states used to implement the PID controller, x is the position of the cam actuator, $T = .008$ seconds, and $ref = 18$.

Mode changes are reflected in the controller output, which does not appear explicitly in the HACDD representation, but instead is reflected in the differential equation governing the continuous dynamics. The modes dynamics and guard conditions are shown in figure 8. In the figure, $u^f_k = -.7x_k - .49z^1_k + .7ref$ and $e_k = ref - x_k$.

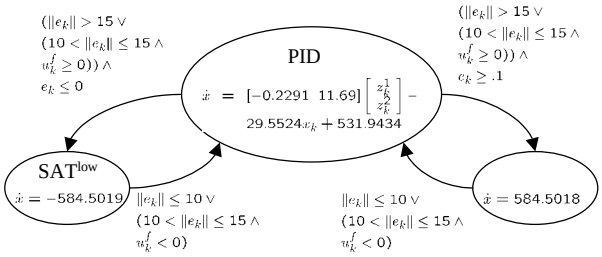

Figure 8: Variable cam timing system automaton

For all transitions, the resets are identity, i.e. $R_{ab}(z) = z$ for all ab in the set of transitions.

The VCT system does not satisfy the given property for all possible initial conditions. Using DTFA, however, we have been able to identify sets of initial conditions for which the system does satisfy the given property and sets of initial conditions for which the system never satisfies the property.

7 Discussion

This paper presents a new model of sampled data control systems, the hybrid automaton with continuous-time and discrete-time dynamics (HACDD), and describes how the sets of reachable states for this model can be computed effectively to verify properties of the system behavior. The role of the HACDD model in the hybrid system verification tool *CheckMate* is also described.

References

[1] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, Springer-Verlag, first edition, 2000.

[2] R. Alur, T.A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," *IEEE Trans. on Software Engineering*, vol. 22, no. 3, pp. 181–201, Mar 1996.

[3] J. J. Scillieri, K. R. Butts, and J. S. Freudenberg, "Validating executable controller specifications through formal model checking," in *Proceedings IEEE International Symposium on Computer-Aided Control System Design*, Sept. 2000.

[4] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell, "An assessment of the current status of algorithmic approaches to the verification of hybrid systems," in *Proceedings of the 40th IEEE Conference on Decision and Control*. 2001, pp. 2867–2874, IEEE Press.

[5] B. I. Silva and B. Krogh, "Modeling and verification of hybrid systems with clocked and unclocked events," in *Proceedings of the 40th IEEE Conference on Decision and Control*. 2001, pp. 762–767, IEEE Press.

[6] S. Kowalewski, S. Engell, J. Preussig, and O. Stursberg, "Verification of logic controllers for continuous plants using timed condition/event-system models," *Automatica*, vol. 35, pp. 505–518, 1999.

[7] T.A. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. 1996, pp. 278–292, IEEE Computer Society Press, Invited tutorial.

[8] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.

[9] B. Silva and B. Krogh, "Formal verification of hybrid systems using *checkmate*: A case study," in *Proceedings of the American Control Conference*, 2000.

[10] A. Chutinan and B. Krogh, "Verification of infinite state dynamic systems using approximate quotient transition systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1401–1410, Sept. 2001.

[11] O. Grumberg and D.E. Long, "Model checking and modular verification," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 3, pp. 843–871, May 1994.

[12] A. Chutinan and B.H. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid Systems: Computation and Control, 2nd International Workshop, HSCC'99*, F.W. Vaandrager and J.H. Van Schuppen, Eds., Berg en Dal, The Netherlands, 1999, Springer-Verlag.

[13] A. G. Stefanopoulou, J. A. Cook, J. S. Freudenberg, J. W. Grizzle, M. Haghgooie, and P. S. Szpak, "Modeling and control of a spark ignition engine with variable cam timing," in *1995 American Control Conference*, pp. 2576–2581.

[14] Kenneth Butts Ford Motor Company, "private correspondence," .