Formal Verification of Hybrid Systems Using *CheckMate*: A Case Study

B. Izaias Silva and Bruce H. Krogh
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890
Izaias@cmu.edu/krogh@ece.cmu.edu

Abstract

We present a formal verification of a control algorithm from the literature for a four-cylinder four-stroke engine in the cutoff mode. The controlled system is modeled, simulated and verified using *CheckMate*, a tool for formal verification of hybrid systems developed at Carnegie Mellon University. *CheckMate* automatically constructs a polyhedral-invariant hybrid automaton (PIHA) from a Matlab/Simulink model of the hybrid system and performs the verification using discrete model approximations. This case study illustrates how verification can be performed directly on a model of the hybrid system dynamics without first constructing an approximation to the continuous dynamics using timed automata or so-called linear hybrid automata models.

Index Terms: hybrid systems, formal verification, finite-state approximations

1 Introduction

Recently, a tool called *CheckMate* was developed at Carnegie Mellon University to perform simulation and verification of a class of hybrid dynamic systems. This paper presents the application of *CheckMate* to an automotive control problem described in [1] and [2]. Our objective is to demonstrate the features of *CheckMate* and its application to practical problems. *CheckMate* deals with a class of hybrid systems called *threshold-event-driven hybrid systems* (TEDHS) for which a verification procedure was proposed in [3]. In a TEDHS the changes in the discrete state can occur only when continuous state variables encounter specified thresholds.

CheckMate models are constructed using a custom graphical user interface (GUI) in the MATLAB Simulink environment. Thresholds in the TEDHS model are hyperplanes. Given the Simulink model, *CheckMate* constructs an equivalent hybrid automaton, which is the basis for the formal verification procedure. The key theoretical concepts used in *CheckMate* are described in [4] and [8].

We present a *CheckMate* model a four-cylinder four-stroke engine and its controller (initially presented in [1]). The problem is to verify properties of an event-driven control law that schedules air/fuel injection when the driver has released the throttle while the car is moving. This is the so-called cut-off mode. The objective is to minimize the mechanical oscillations as the car decelerates. Using the formulation of the resulting verification problem described in [1] and [2], we illustrate how *CheckMate* can be applied directly to the dynamic model of the hybrid system. This is in contrast to the application of the tool *HyTech* presented in [2] for this problem, which requires significant effort to first develop a linear hybrid system model of the system dynamics.

The following section gives an overview of the *CheckMate* user interface and the verification procedure implemented in *CheckMate.* Section 3 describes the model of the automotive cutoff mode dynamics and control algorithm. Section 4 presents the *CheckMate* model for the cutoff mode control algorithm and section 5 presents the verification results. Section 6 summarizes the results of this case study and describes some directions of current research on extending the capabilities of the *CheckMate* tool. The latest version of *CheckMate* can be downloaded from the *CheckMate* webpage at http://go.to/CheckMate.

2 Overview of *CheckMate*

CheckMate is a formal verification tool developed in Matlab/Simulink for TEDHSs with thresholds defined by hyperplanes. TEDHSs are modeled using Simulink block diagrams. The model is then converted into a *polyhedral-invariant hybrid automaton* (PIHA), used for verification. The resulting PIHA is equivalent to the original TEDHS within a bounded region of the continuous state space called the *analysis region*. The system specification is expressed as an ACTL formula [7], a restriction of the CTL (Computation Tree Logic). For the theoretical background of *CheckMate*, see [3] and [8].

A. CheckMate GUI

The *CheckMate* GUI is based on Matlab/Simulink using customized blocks. The components of a TEDHS model are switched continuous system blocks (SCSBs), polyhedral threshold blocks (PTHBs) and finite state machine blocks (FSM). Figure 1 shows a *CheckMate* block diagram.

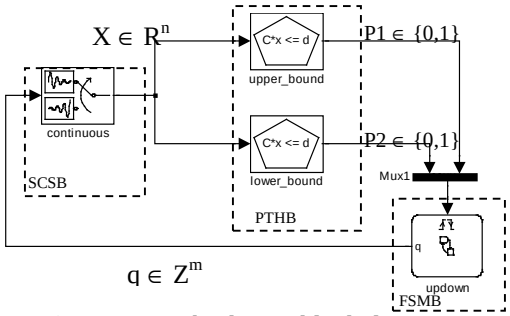

Figure 1. A *CheckMate* block diagram.

Switched Continuous System Block (SCSB)

A SCSB models continuous dynamics with state equations that are selected according to the value of a discrete input signal. Three types of state equations are supported, *clocks* (pure integrators), *affine* ($\dot{x} = Ax + b$), and *nonlinear* ($\dot{x} = f(x)$). The routine used to approximate the reachability set in the verification procedure is selected according to the type of dynamics in the SCSBs.

Polyhedral Threshold Block (PTHB)

A PTHB represents a convex polyhedron parameterized by the matrix-vector pair (C,d). The input is a continuous state vector x and the output is a Boolean signal indicating whether or not the continuous state x lies within the polyhedron $Cx \leq d$. The input must be a single vector of SCSBs outputs. If necessary, outputs from multiple SCSBs must be combined using a Simulink MUX block to form a single vector as an input to a PTHB.

Finite State Machine (FSMB)

A FSMB block is a statechart block from the MATLAB Stateflow Toolbox with the following restrictions:

- Event Inputs: Each component of the event input vector must be a logical function of the outputs of PTHBs.
- Each input signal must be a logical function of the outputs of PTHBs or FSMBs.
- Output: Only one discrete data output is allowed.
- The Stateflow diagram must contain no hierarchy. Each state in a Stateflow diagram must assign a unique value to the data output in its entry action. No other action is permitted on any state or transition label.

Standard Simulink Blocks

- For simulation, any Simulink/StateFlow block can be used. For example, a step signal generator block can be used to create initializing events for the FSMB, and scope blocks used to plot simulation trajectories.
- For verification, logical operators (AND, OR, XOR, etc) and signal MUX/DEMUX blocks can be used to construct the TEDHS model.

B. CheckMate Verification Procedure

Verification in *CheckMate* is based on the theory of *approximate quotient transition systems* (AQTSs) for hybrid automata [8]. Given an ACTL expression, *CheckMate* verifies whether it is true for an AQTS, which is a conservative (outer) approximation to the sequential behavior of the hybrid automaton. If the assertion is true for the AQTS, it is true for the hybrid automaton. If it is not true, a less conservative AQTS can be constructed and the verification can be tried again. *CheckMate* handles *polyhedral-invariant hybrid automata* (PIHA), which are automata with invariants defined by the hyperplanes defining guards for the events leaving each discrete state (or *location*). Figure 2 illustrates the steps in the verification procedure. The first step in the verification procedure is to construct a PIHA that is equivalent to the TEHDS modeled by the *CheckMate* block diagram. In the current implementation, the locations in the PIHA correspond to *cells* in the continuous state space defined by the hyperplanes in the PTHBs. The continuous dynamics associated with each PIHA location and the PIHA events are guards are inferred from the logic in the FSMBs in the *CheckMate* model.

The next step is to create an initial partition of the hyperplane faces for the PIHA invariants. The elements of this partition become the states in the first AQTS. The transitions in the AQTS are computed using *flowpipe approximations* that are conservative approximations to the flows (reachable states) of the continuous dynamics leaving each AQTS state. Transitions are introduced when the entry states from one location can lead to the entry states of another location.

Given the AQTS for a partition of the PIHA invariants, along with an ACTL expression, verification is performed using fixed-point computations on the ACTL state space. If the ACTL expression is found false, the approximation could be too conservative. In this case, *CheckMate* has a procedure for refining the invariant partitions and computing a new AQTS, and the verification can be attempted again. As shown in Figure 2, *CheckMate* tests to see if the current AQTS is a *bisimulation*, in which case the AQTS is an exact, rather than approximate, representation of the hybrid system dynamics and the verification on the AQTS is conclusive.

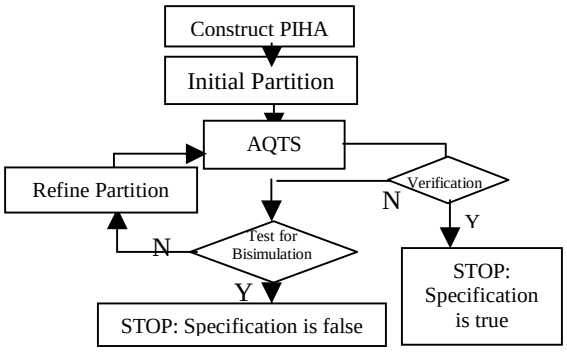

Figure 2. The *CheckMate* verification procedure.

3 Cutoff Control Problem

The cutoff-mode control problem is described in [1] and [2]. The control objective is to reach injection cutoff while minimizing acceleration oscillations. The model can be divided into two sub-systems: the *driveline model* and the *hybrid cylinder model*.

A. Driveline Model

The *driveline model* deals with the physical variables of the car engine. The original continuous model was developed by Magnetti-Marelli [2]. The physical state variables are: z_1 (engine block angle (radians)), z_2 (wheel revolution speed (radians/s)), z_3 (axle torsion angle (radians)), z_4 (crankshaft revolution speed (rpm)), and ϕ (crankshaft angle (degrees)). The parameters for the system can be found in [2]. According to [1] the oscillation in the system can be studied in the two-dimensional space of state variables x_2 and x_3 in a system obtained by a similarity transformation. Thus, the design in [1] is derived using the system sub-matrices

$$A_{23} = \begin{bmatrix} -3.3216 & -25.736 \\ 25.736 & -3.3216 \end{bmatrix} \quad b_{23}{}^{T} = [26.3824 \quad -34.9729].$$

The crankshaft angle rate depends on all of the state variables, but the influence of state variables x_1 and x_4 can be reduced to a constant input by evaluating their average values from simulations.

B. Hybrid cylinder model

In a four-cylinder four-stroke engine there are four discrete states for each cylinder: exhaust, intake, compression and combustion. The state of each cylinder changes when the crankshaft angle satisfies **mod**(ϕ,180) = 0 which defines the *phase_change* event. The finite state machine in Fig. 3 shows this discrete-state behavior.

C. Cut-off Mode Control Law

Torque is supplied in the combustion phase when fuel has been injected at the intake phase. The controller must decide whether or not to inject fuel for a particular cylinder during when the cylinder is in its exhaust phase. Therefore, the effect of a control decision is realized three cycles after the decision is made. We introduce the following notation. Let u(k) denote the value of the control decision determining whether or not there will be torque from the combustion phase after the k^{th} phase change. The value of u(k) is either 0 (no torque) or a constant m =10Nm (torque). At phase change k, the controller must choose the value of u(k+3). The cut-off mode control law uses a predictive model to evaluate at each exhaust phase whether or not to inject fuel on the subsequent intake phase. Let the state vector be $x = [x_2\ x_3]^T$, and let x(k) denote the value of the state vector at the k^{th} phase_change event. The predicted values of the state for the next four phase changes are denoted x(1/k),...,x(4/k). We have then that $x(4/k) = P^4x(k) + P^3qu(k) + P^2qu(k+1) + Pqu(k+2) + qu(k+3)$, where T=0.00642271sec is the sampling rate, $P = \exp(A_{23}T)$, $q = A_{23}^{-1}(P - I)b_{23}$ and the values of u(k), u(k+1), and u(k+2) are already determined. Torque decisions in the cutoff mode are taken according to "comfort criteria." In [2] these criteria are translated into the following rule:

$$u(k+3) = \begin{cases} 0 & if\ |\gamma^T x^0(4/k)| \le |\gamma^T x^m(4/k)| \\ m & otherwise \end{cases},$$

where γ^T=[0.783, -0.621] and $x^0(4/k)$ and $x^m(4/k)$ are the predicted state values for u(k+3)=0 and m, respectively.

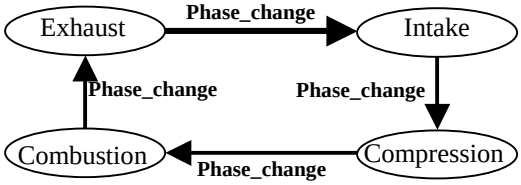

Figure 3 - States for one cylinder.

D. The Verification Problem

Figure 4 shows a trajectory for x(0) = [74.3275,-39.7538] and $\phi_0 = 0^0$. We focus on the region where the system starts to chatter, which continues until the state enters a ball of radius ρ. At that point fuel injection ceases and free evolution behavior takes place [1]. We show the region of interest for formal verification in Fig where d = 2.74 and h = 8.8762. The initial set is $(x_2,x_3) \in$ ([6.6, 6.9] x [9.9,10.4]). The objective of the verification is to show that the trajectory does not cross the boundaries of the verification region a distance d from the switching line before reaching the "finish line" in Fig. 5. The trajectories shown in Fig. 5 are simulations starting at the four corners of the set of initial states. These trajectories are insufficient to verify that *no* trajectories will leave the verification region due to the nonlinearities introduced by the switching rules.

4 *CheckMate* Model

The *CheckMate* model for the cutoff control system comprises driveline dynamics, angle modulation and torque decision/prediction models. Figure 6 shows a *CheckMate* block diagram for this system. In this section, we describe the details of this diagram.

A. Driveline Dynamics model

The driveline dynamics are modeled in a SCSB with two modes representing the free evolution when there is no torque and the force evolution when there is torque. There are three state variables, x_2, x_3, and ϕ. The mode is selected by the discrete output coming from the FSM.

B. Angle Modulation

Two PTHBs *reach_zero* and *reach_180* generate events when the torque angle ϕ crosses 0^0 or 180^0 . An FSM receives the output of PTHB and generates event *phase_change*, which toggles the sign of $\dot{\phi}$ in the continuous dynamics.

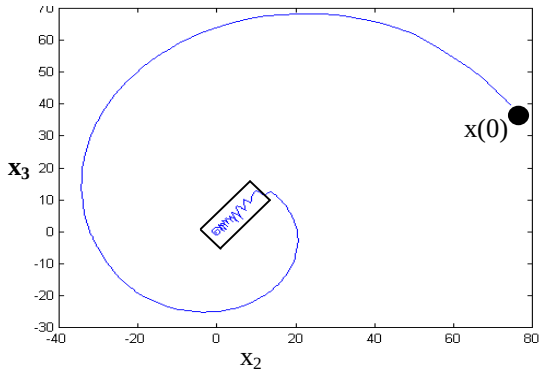

Figure 4. Simulation of the system.

C. Torque decision and Prediction

The controller behavior consists of a FSM shown in Fig. 7 that keeps track of the three torque input values that have already been decided. The states in this FSM are labeled h000, h001, h010, h011, h100, h101, h110 and h111 corresponding the eight possible on-off values for the controls u(k),u(k+1),u(k+2). From each state there are two possible transitions. To represent the decision logic, we consider the following predicates:

A : $\gamma^T x^0(4/k) \geq 0$
B : $\gamma^T x^m(4/k) \geq 0$
C : $\gamma^T x^0(4/k) \geq \gamma^T x^m(4/k)$
D : $\gamma^T x^0(4/k) \leq -\gamma^T x^m(4/k)$.

In terms of these predicates, the control u(k+3)=m if and only if the following predicate is true:

S(A,B,C,D)=ABC + (~A)BD + A~B~D + ~A~B~C

Letting $h = P^3 qu(k) + P^2 qu(k+1) + Pqu(k+2)$ $G = P^4$, $j = qm$

A : $-\gamma^T P^4 x_0 \leq \gamma^T h$
B : $-\gamma^T P^4 x_0 \leq \gamma^T h + \gamma^T qm$
C: $\gamma^T qm \leq 0$.
D : $2\gamma^T P^4 x_0 \leq -2\gamma^T h - \gamma^T qm$

Observing that: $\gamma^T qm > 0$, A→ B, ~D → B and D→ ~A we can reduce S(A,B,C,D) = S(D) = D. In other words, only predicate D is necessary to make the control decision. Figure 8 shows the torque decision regions depending on the prediction for the next three states: the white (black) region means proposition D is false (true).

D. Simulation and Validation using CheckMate

For this example, the ACTL specification is AF(angle==cross_finish_line)& (AG~out_of_bound)). It says: "eventually" the system will cross the line named "finish line" and "never" go out of the "out_of_bound" box (the region is presented in Fig. 5). With *CheckMate* it is possible simulate and check if the specification (in ACTL) is satisfied by trajectories starting at vertices of the initial state set. Simulated trajectories are shown in Fig. 5.

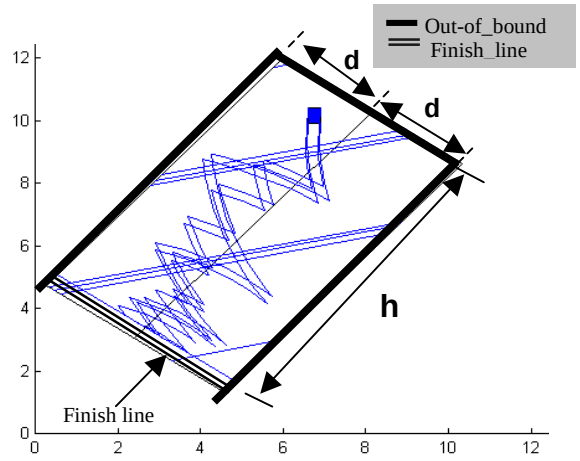

Figure 5. Sliding mode of the Cutoff control System.

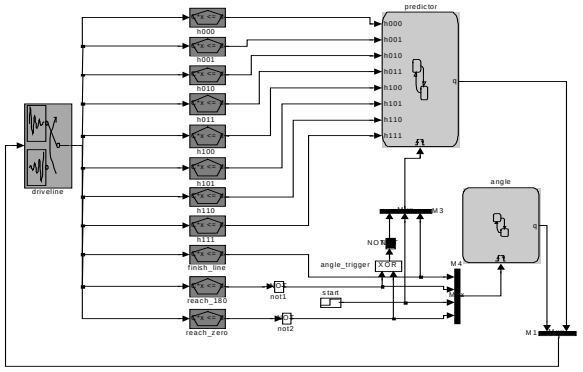

Figure 6. *CheckMate* model for cutoff control System.

5 Verification using *CheckMate*

The verification of the cutoff control system was done using the same parameters presented in [1]. The initial condition set is: $(x_2, x_3) \in [6.6,6.9]x[9.9,10.4]$, $\phi = 0.1$. $\dot{\phi}(0) > 0$. We started in the discrete state "110", reflecting the torque decisions already stored at the initial state.

The verification was successful. Table 1 presents some of the data from the verification. The elapsed time was 111 minutes (on a 600MHz PC). No refinements were necessary; the verification was true for the first AQTS.

Figure 9 shows the sets if reachable states in the x_2-x_3 plane at the switching instants. Note that these sets include points that would not be predicted from the simulations in Fig. 5 starting from the vertices of the initial state set.

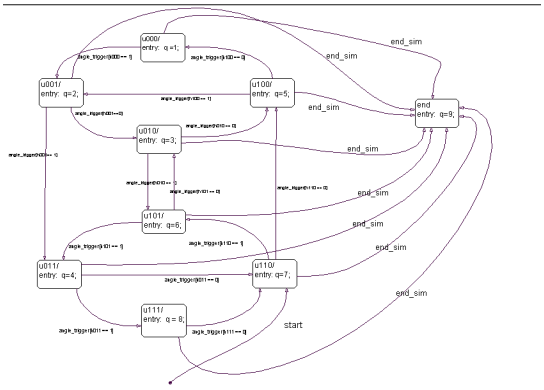

Figure 7. FSM predictor.

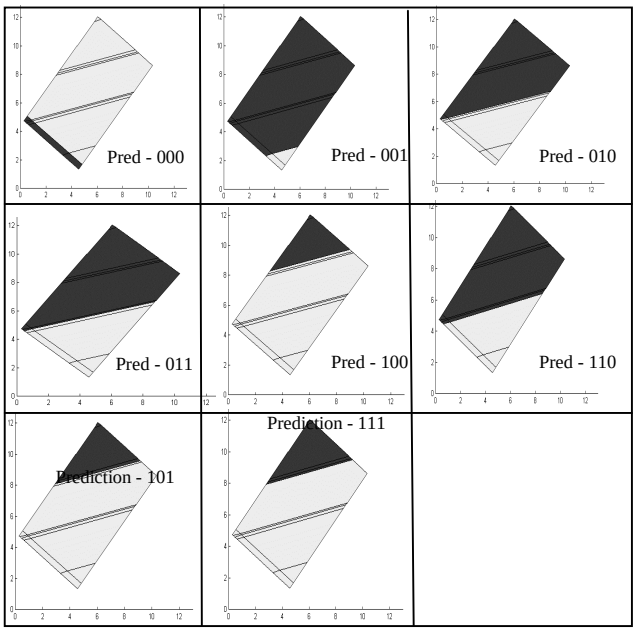

Figure 8. Decision mapping regions.

6 Concluding Remarks

This paper presents a case study applying *CheckMate*, a new tool for computer-aided verification of hybrid systems. We verified the correctness of a cutoff control system for a combustion engine using a model and data from the literature. This study demonstrates the following:

1) *CheckMate* deals directly with the system dynamics, rather than constructing a simplified model using linear or timed automata as required by other tools;

2) With *Checkmate* it is possible to simulate the system behavior for selected initial conditions. This is a quick and powerful way to debug the model, reason about the system, and obtain preliminary verification results.

In the course of doing this case study, we developed a new method in *CheckMate* for constructing the AQTS for systems affine continuous dynamics that improved the computation time dramatically. Details of this method will be published elsewhere. Current research is focusing on verification of sampled-data hybrid systems and methods for reducing the size and complexity of the PIHA representation of a given TEDHS.

Hyperplanes	11	Cells	42
PIHA locations	288	Discrete states	288
Refinements	None	Time elapsed	111 m

Table 1. Verification data.

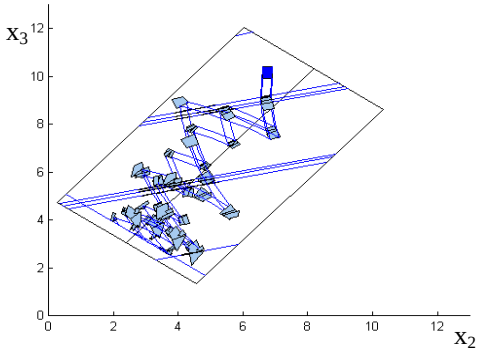

Figure 9. Reachable set (partial 2D view).

Bibliography

[1] Balluchi, A. et al., Hybrid Control for Automotive Engine Management: The Cut-Off Case. In HSCC 98: Hybrid Systems - Computation and Control, Lecture Notes in computer Science 1386, Springer- Verlag, 1998.

[2] Villa, T. et al., Formal Verification of an Automotive Engine Controller in Cutoff Mode, Proc. of the 37th CDC 1998, Tampa, Florida USA.

[3] Cury, J.E. R., Krogh, B.H. and Niinomi, T. . Synthesis of supervisory controller for hybrid systems based on approximating automata. IEEE Transaction on Automatic Control, 43(4):564-569, April 1998.

[4] Chutinan, A. and Krogh B.H. Computing polyhedral approximating automata for a class of linear hybrid systems. In Hybrid Systems V, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[5] Chutinan, A. and Krogh B.H. Computing polyhedral approximations to dynamic flow pipes. The 37th IEEE conference on Decision and Control: Invited Session on Synthesis and Verification of Controllers for Hybrid Systems, 1998.

[6] Zhao, Feng. Automatic analysis and synthesis of controllers for dynamical systems based on phase-space knowledge. PhD thesis, MIT artificial intelligence Laboratory, August 1992.

[7] Clarke, E.M., Grumberg, O and Long, D. Verification tools for finite-state concurrent systems. In Proceeding of A Decade of Concurrency: Reflections and Perspectives, pages 124-175, REX School/Symposium, Noordwijkerhout, The Netherlands, 1-4 June 1993. Springer-Verlag, Berlin, Germany, 1994.

[8] Chutinan,A. and Krogh B.H., Approximate Quotient Transition Systems for Hybrid Systems. In Proc. 2000 American Control Conference, Chicago, June 2000,

[9] Henzinger, T., et al. Hytech: A model checker for hybrid systems. Software tools for technology transfer, 1(1):110-122, 1997.