Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

Discrete-Time Hybrid Modeling and Verification

Fabio Danilo Torrisi[1] and Alberto Bemporad[2,1]

Abstract

For hybrid systems described by interconnections of linear dynamical systems and logic devices, we recently proposed Mixed Logical Dynamical (MLD) systems and the language HYSDEL (Hybrid System Description Language) as a modeling tool. For MLD models, we developed a reachability analysis algorithm which combines forward reach-set computation and feasibility analysis of trajectories by linear and mixed-integer linear programming. In this paper the versatility of the overall analysis tool is illustrated on the verification of an automotive cruise control system for a car with robotized manual gear shift.

1 Introduction

Hybrid models describe processes that evolve according to dynamic equations and logic rules [1]. The interest in hybrid systems has grown over the last few years not only because of the theoretical challenges, but also because of their impact on applications, for instance in the automotive industry. A typical example of hybrid systems are real-time systems, where physical processes are controlled by embedded controllers. For this reason, it is important to have tools to guarantee that this combination behaves as desired. Formal verification is aimed at providing such a certification. This amounts to solving the following reachability problem: For a given set of initial conditions and disturbances, guarantee that all possible trajectories never enter a set of unsafe states, or possibly provide a counterexample. Unfortunately, it is well known that formal verification of hybrid systems is an undecidable problem. In spite of this complexity, several tools for formal verification of hybrid systems have been proposed in the literature [2, 8, 10, 12, 16].

Formal verification is strictly related to the modeling framework used to describe the process, whose safety properties we need to certify. Different models lead to different verification algorithms. Timed automata [8] and hybrid automata [12], for instance, were proved to be successful modeling frameworks and led to very efficient verification tools based on symbolic computation.

Recently, we proposed an algorithm [6, 7] based on mathematical programming for solving safety analysis problems on *Piecewise Affine* (PWA) and *Mixed Logical Dynamical* (MLD) systems as an alternative to other modeling frameworks. PWA and MLD systems are formulated in discrete time. Despite the fact that the effects of sampling can be neglected in most applications, subtle phenomena such as Zeno behaviors [14] cannot be captured in discrete time. Nevertheless, a discrete-time formulation allows one to develop numerically tractable schemes for solving complex analysis and synthesis problems. Several questions related to MLD systems can indeed be suitably formulated as mixed-integer linear/quadratic optimization problems, such as control [5] and scheduling problems [4].

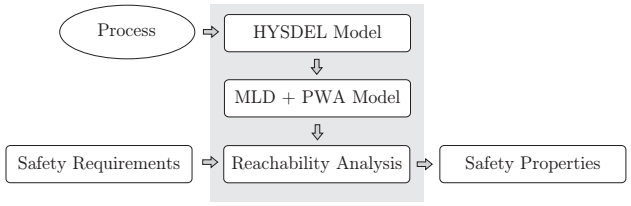

Figure 1: Verification of hybrid systems: The process is modeled in HYSDEL. The model is automatically translated it into MLD and PWA form, and used for proving safety properties

After reviewing the basics of PWA and MLD systems, we briefly recall the reachability analysis algorithm described in [6], and address the formal verification problem of an automotive control system for automatized gear-shift. We model the automotive system in our language HYSDEL to obtain an MLD system. For formal verification, we adopt the algorithm described in [6]. The overall modeling and verification procedure is depicted in Figure 1.

2 PWA Systems, MLD Systems, and HYSDEL

PWA systems [11] are defined by partitioning the state space into polyhedral regions, and associating with each region a different linear state-update equation

$$\begin{array}{ll} x(t+1) & = A_{i(t)}x(t) + B_{i(t)}u(t) + f_{i(t)} \\ & \text{if } x(t) \in \mathcal{C}_{i(t)} \triangleq \{x : \ H_{i(t)}x \leq K_{i(t)}\} \end{array} \quad (1)$$

where $x \in X \subseteq \mathbb{R}^n$, $u \in U \subseteq \mathbb{R}^m$, $\{\mathcal{C}_i\}_{i=0}^{s-1}$ is a polyhedral partition[1] of the sets of states[2] X, the matrices A_i, B_i, f_i, H_i, K_i are constant and have suitable

[1]Automatic Control Laboratory, ETH Zentrum - ETL, 8092 Zürich, Switzerland `torrisi,bemporad@aut.ee.ethz.ch` `http://control.ethz.ch/~hybrid`

[2]Dip. Ingegneria dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy `bemporad@dii.unisi.it`

[1]The double definition of the state-update function over common boundaries of sets $\mathcal{C}_i$ (the boundaries will also be referred to as *guardlines*) is a technical issue that can be resolved by allowing a part of the inequalities in (1) to be strict. However, from a numerical point of view, this issue is not relevant.

[2]More generally, $\mathcal{C}_i$ are subsets the of state+input space $\mathbb{R}^{n+m}$, for instance when the PWA system is obtained as an equivalent representation of an MLD system [11].

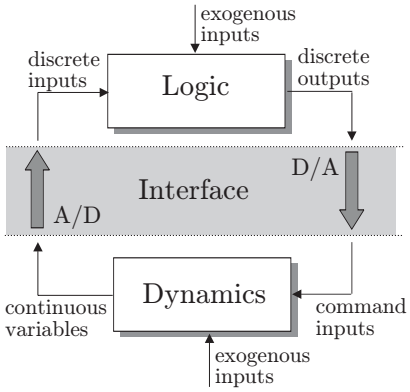

Figure 2: Hybrid MLD system: Finite state machines and continuous dynamics interact through analog-to-digital (A/D) and D/A interfaces

dimensions, and the inequality $H_i x \leq K_i$ should be interpreted componentwise. PWA systems can model a large number of physical processes, such as systems with static nonlinearities, and can approximate nonlinear dynamics via multiple linearizations at different operating points.

MLD systems [5] allow specifying the evolution of continuous variables through linear dynamic equations, of discrete variables through propositional logic statements and automata, and the mutual interaction between the two, as shown in Figure 2. The MLD modeling framework relies on the idea of translating logic relations into mixed-integer linear inequalities [5, 19]. Linear dynamics are represented as difference equations $x(t+1) = Ax(t) + Bu(t)$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$. Boolean variables are defined from linear-threshold conditions over the continuous variables: $[d = 1] \leftrightarrow [a'x \leq b]$, $x,\ a \in \mathbb{R}^n$, $b \in \mathbb{R}$, $d \in \{0,1\}$. This can be equivalently expressed by the two mixed-integer linear inequalities:

$$\begin{array}{rcl} a'x - b & \leq & M(1-d) \\ a'x - b & > & md \end{array} \tag{2}$$

where, assuming that $x \in \mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{X}$ is a given bounded set, $M \geq \sup_{x\in\mathcal{X}}(a'x - b)$, $m \leq \inf_{x\in\mathcal{X}}(a'x - b)$, are upper and lower bounds, respectively, on $(a'x - b)$.

Similarly, a relation defining the continuous variable z defined by IF d THEN $z = a_1'x - b_1$ ELSE $z = a_2'x - b_2$, can be expressed as

$$\begin{array}{rcl} (m_2 - M_1)d + z & \leq & a_2'x - b_2 \\ (m_1 - M_2)d - z & \leq & -a_2'x + b_2 \\ (m_1 - M_2)(1-d) + z & \leq & a_1'x - b_1 \\ (m_2 - M_1)(1-d) - z & \leq & -a_1'x + b_1. \end{array} \tag{3}$$

A Boolean variable $d_n \in \{0,1\}$ can be defined as a Boolean function of Boolean variables $f : \{0,1\}^{n-1} \mapsto \{0,1\}$, namely

$$d_n = f(d_1, d_2, \ldots, d_{n-1}) \tag{4}$$

where f is a combination of "not" ($\sim$), "and" ($\wedge$), "or" ($\vee$), "exclusive or" ($\oplus$), "implies" ($\leftarrow$), and "iff" ($\leftrightarrow$) operators. Eq. (4) can be translated into the conjunctive normal form (CNF): $\bigwedge_{j=1}^{k} \left(\bigvee_{i\in P_j} d_i \bigvee_{i\in N_j} \sim d_i \right)$, $N_j, P_j \subseteq \{1, \ldots, n\}$ by using standard methods [18]. Subsequently, the CNF can be translated into the set of integer linear inequalities

$$\begin{cases} 1 \leq \sum_{i\in P_1} d_i + \sum_{i\in N_1}(1 - d_i) \\ \vdots \\ 1 \leq \sum_{i\in P_k} d_i + \sum_{i\in N_k}(1 - d_i) \end{cases} \tag{5}$$

The state update law of finite state machines can be described by logic propositions involving binary states, their time updates, and binary signals, under the assumptions that transitions are clocked and synchronous with the sampling time of the continuous dynamical equations, and that the automaton is well posed (i.e., at each time step a transition exists and is unique)

$$x_\ell(t+1) = F(x_\ell(t), u_\ell(t)) \tag{6}$$

where u_l is the vector of Boolean signals triggering the transitions of the automaton. Therefore the automaton is equivalent to a nonlinear discrete time system where F is a purely Boolean function. The translation technique mentioned above can be directly applied to translate the automaton (6) into a set of linear integer equalities and inequalities. An example will be provided when modeling the control code of the gear-shift controller in Section 4.

By collecting the equalities and inequalities derived from the representation of the hybrid system depicted in Figure 2, we obtain the Mixed Logical Dynamical (MLD) system [5]

$$x(t+1) = \Phi x(t) + \mathcal{G}_1 u(t) + \mathcal{G}_2 d(t) + \mathcal{G}_3 z(t) \tag{7a}$$

$$y(t) = \mathcal{H} x(t) + \mathcal{D}_1 u(t) + \mathcal{D}_2 d(t) + \mathcal{D}_3 z(t) \tag{7b}$$

$$\mathcal{E}_2 d(t) + \mathcal{E}_3 z(t) \leq \mathcal{E}_1 u(t) + \mathcal{E}_4 x(t) + \mathcal{E}_5 \tag{7c}$$

where $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ is a vector of continuous and binary states, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ are the inputs, $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ the outputs, $d \in \{0,1\}^{r_\ell}$, $z \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous variables respectively, which are introduced when transforming logic relations into mixed-integer linear inequalities, and Φ, $\mathcal{G}_{1-3}$, $\mathcal{H}$, $\mathcal{D}_{1-3}$, $\mathcal{E}_{1-5}$ are matrices of suitable dimensions.

We assume that system (7) is *completely well-posed* [5, 11], which means that $x(t+1)$ and $y(t)$ are uniquely defined once $x(t)$, $u(t)$ are given. Note that the constraints (7c) allow to specify additional linear and logic constraints on the variables of the system (e.g., limits on physical variables).

The whole translation procedure is the core of the tool HYSDEL (Hybrid Systems Description Language) [18], which automatically generates an MLD model from a high-level textual description of the system. Once the

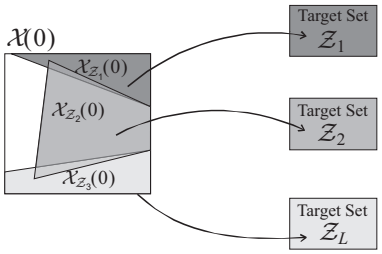

Figure 3: Reachability analysis problem

MLD model is available, its equivalent PWA form (1) is obtained by using the procedure suggested in [11]. In the next section, we will assume that both the PWA and the MLD forms are available and discuss their complementary role in the reachability analysis algorithm.

3 Reachability Analysis

The *reachability analysis* of hybrid dynamical systems allows the verification of *safety properties*: For a given set of initial conditions and exogenous signals, verify that the set of unsafe states cannot be entered, or provide a counterexample. More precisely, we define the following:

Reachability Analysis Problem. Given a hybrid system Σ (either in PWA form (1) or MLD (7)), a set of initial conditions $\mathcal{X}(0)$, a collection of disjoint target sets $\mathcal{Z}_1$, $\mathcal{Z}_2$, ..., $\mathcal{Z}_L$, a bounded set of inputs $\mathcal{U}$, and a time horizon $t \leq T_{\max}$, determine (*i*) if $\mathcal{Z}_j$ is reachable from $\mathcal{X}(0)$ within $t \leq T_{\max}$ steps for some sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$ of exogenous inputs; (*ii*) if yes, the subset of initial conditions $\mathcal{X}_{\mathcal{Z}_j}(0)$ of $\mathcal{X}(0)$ from which $\mathcal{Z}_j$ can be reached within $T_{\max}$ steps; (*iii*) for any $x_1 \in \mathcal{X}_{\mathcal{Z}_j}(0)$ and $x_2 \in \mathcal{Z}_j$, the input sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$, $t \leq T_{\max}$, which drives x_1 to x_2.

From now on, we will assume that $\mathcal{X}(0)$, $\mathcal{Z}_j$, $\mathcal{U}$ are polyhedral sets, and, without loss of generality, that they are also convex. We will also denote by $\mathcal{X}(t, \mathcal{X}(0))$ the *reach set* at time t starting from any $x \in \mathcal{X}(0)$ and by applying any input $u(k) \in \mathcal{U}$, $0 \leq k \leq t-1$. The reason for focusing on finite-time reachability is that states which are not reachable in less than $T_{\max}$ steps are, in practice, unreachable. Although finite time reachability analysis cannot guarantee certain liveness properties (for instance, if $\mathcal{Z}_i$ will be ever reached), the reachability problem stated above is clearly decidable. Nevertheless, the problem is $\mathcal{NP}$-hard [7].

The previous questions of reachability can be answered once all the switching sequences $I(T) \triangleq \{i(0), \ldots, i(T-1)\}$, $\forall T \leq T_{\max}$ leading to $\mathcal{Z}_1$, or $\mathcal{Z}_2$, ..., or $\mathcal{Z}_L$ from $\mathcal{X}(0)$ are known. In fact, it is enough to check that the reach set at time T satisfies $\mathcal{X}(T, \mathcal{X}(0)) \cap \mathcal{Z}_j \neq \emptyset$ for all admissible switching sequences $I(T)$.

Algorithm 3.1

```
1   Given: X(0) (to be explored), target sets Z_j, j = 1,...,L;
2   push X(0, X(0)) in STACK ;
3   while STACK not empty do
4     pop X(t, R_j) from STACK, let i such that R_j ⊆ C_i;
5     while not terminate(X(t, R_j))
6       if X(t, R_j) ∩ Z_k ≠ ∅ then Z_k can be reached from R_j
7       t ← t + 1; X(t, R_j) ← A_i X(t − 1, R_j) + B_i U + {f_i};
          % Update the reach set
8       for all h ≠ i such that R_h ≜ C_h ⋂ X(t, R_j) ≠ ∅
          % R_h can be reached from R_j
9         R̄_h ← outer approximation of R_h;
10        Push R̄_h on STACK;
11      X(t, R_j) ← X(t, R_j) ⋂ C_j;
```

Table 1: Reachability analysis algorithm described in [7]

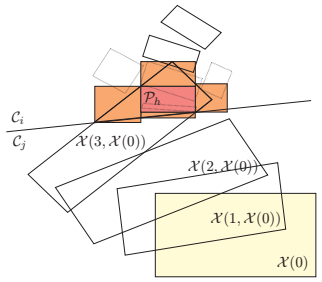

Figure 4: Reach-set evolution, guardline crossing, outer approximation of a new intersection

3.1 Verification Algorithm

In order to determine admissible switching sequences $I(T)$, we need to exploit the special structure of the PWA system (1). This allows an easy computation of the reach set, as long as the evolution remains within a single region $\mathcal{C}_i$ of the polyhedral partition. Whenever the reach set crosses a guardline and enters a new region $\mathcal{C}_j$, a new reach-set computation based on the j-th linear dynamics is computed, as shown in Figure 4. The Algorithm proposed in [6] is summarized by the pseudo-code reported in Table 1, where we assume that $\mathcal{X}(0) \subset \mathcal{C}_i$ is a convex polyhedral set (more generally, we can consider all nonempty intersections $\mathcal{X}(0) \cap \mathcal{C}_i$, for all $i = 1, \ldots, s$). The algorithm determines the reach set $\mathcal{X}(T, \mathcal{X}(0))$ for all $T \leq T_{\max}$ (i.e., the *reachable set*).

Reach-Set Computation

The reach set $\mathcal{X}(t, R_j) \cap \mathcal{C}_i$ is computed iteratively in steps *4*, *7*, and *11*. Note that the subset $S_i(t, \mathcal{X}(0))$ of $\mathcal{X}(0)$ whose evolution lies in $\mathcal{C}_i$ for t steps is given by the explicit representation

$$S(t, \mathcal{X}(0)) = \big\{ x \in \mathbb{R}^n : \; S_0 x_0 \leq T_0, \; H_i A_i^k x_0 \leq S_i - H_i \textstyle\sum_{j=1}^{k} A_i^{j-1} [B_i u(k-j) + f_i], \; k = 1, \ldots, t \big\} \tag{8}$$

which is a polyhedral set in the augmented space of tuples $(x, u(0), \ldots, u(t-1))$.

Guardline Crossing Detection

Step *8* requires a switching detection, namely find-

ing all possible new regions $\mathcal{C}_h$ entered by the reach set at the next time step, or, in other words, all the nonempty sets $R_h \triangleq \mathcal{X}(t, \mathcal{X}(0)) \bigcap \mathcal{C}_h$, $h \neq i$. Rather than enumerating and checking nonemptiness for all $h = 0, \ldots, i-1, i+1, \ldots, s-1$, we exploit the equivalence between PWA and MLD, and solve the switching detection problem via mixed-integer linear programming. In fact, switching detection amounts to finding all feasible vectors $d(t) \in \{0,1\}^{r_\ell}$ which are compatible with the constraints in (7) plus the constraint $x(t-1) \in \mathcal{X}(t-1, \mathcal{X}(0)) \cap \mathcal{C}_i$.

Approximation of Intersection
The computation of the reach set proceeds in each region $\mathcal{C}_h$ from each new intersection R_h. A new reach-set computation is started from R_h, unless R_h is contained in some larger subset of $\mathcal{C}_h$ which was already explored. The set R_h is outer approximated in step *9* by the union of hyper-rectangles [3], as depicted in Figure 4, in order to avoid that the complexity of R_h grows linearly with time, and so that checking for set inclusion reduces to a simple comparison of the coordinates of the vertices of the hyper-rectangles.

Termination of Explorations
In (8) we showed how to compute the evolution of the reach set $\mathcal{X}(t, R_j)$ inside a region $\mathcal{C}_i$. The computation is stopped (step *5*) once one of the following condition happens: (1) The set $\mathcal{X}(t, R_j)$ is empty (this means that the whole evolution has left region $\mathcal{C}_i$); (2) $\mathcal{X}(t, R_j) \subseteq \mathcal{Z}_j, j = 1, \ldots, L$, the target set $\mathcal{Z}_j$ has been reached by all possible evolutions from R_j; (3) $t > T_{\max}$. These conditions can be easily checked through linear programming. Note that the termination condition (2) implies that once a target set has been reached no further exploration is performed.

Post-processing
The result of the exploration algorithm detailed in the previous sections is conveniently stored in a graph G. The nodes of G represent sets from which a reach-set evolution is computed, and an oriented arc of G connects two nodes if a transition exists between the two corresponding sets. Each arc has an associated weight which represents the time-steps needed for the transition. After Algorithm 3.1 terminates, the oriented paths on G from initial node $\mathcal{X}(0)$ to terminal nodes $\mathcal{Z}_j, j = 1, \ldots, L$ determine a superset of feasible switching sequences $I(t) = \{i(0), \ldots, i(t-1)\}$. In fact, because of the outer approximation $\bar{R}_h$ of new intersections R_h (step *9*), not all switching sequences are feasible. Feasibility can be simply tested via linear programming over the sets of linear inequalities generated by an explicit reach-set representation as in (8).

4 Verification of a Cruise Control System

In this section we apply the tools proposed in the previous sections to verify an automotive cruise control system for a car with robotized manual gear shift. The cruise control system controls the throttle, the brakes and the gear box to let the speed of the car track a desired reference. In this section we verify that the controlled system will never exceed the target speed by some tolerance, for instance to guarantee that the speed limits imposed by local authorities will never be exceeded. The complete system under exam is composed by two subsystems: the car dynamic model and the controller. We focus on a car equipped with manual transmission, and we assume that the gear command is robotized, namely that the controller can select the gear and a slave control system takes care of releasing the clutch, shifting the gear, and engaging the clutch.

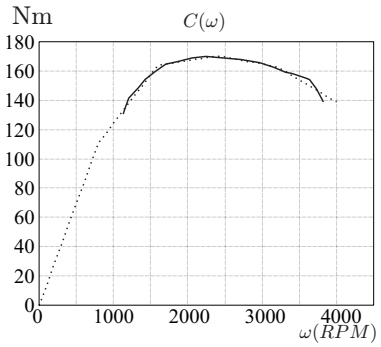

Figure 5: Torque of the engine of the Clio 1.9 DTI RXE (solid line) and PWL approximation (dashed line)

4.1 Car model

We only consider the longitudinal dynamics of the car: the continuous variables are the scalar position x (m) and the speed $v = \dot{x}$ (m·s^{-1}). The continuous inputs are the engine torque u_t (Nm), the braking force u_b (N), and the sinus of the road slope u_s, plus five binary inputs g_1, g_2, g_3, g_4, and $g_5 \in \{0,1\}$ corresponding to the selected gear. By denoting by ω the engine speed (rad·s^{-1}), we have the kinematic relation

$$v = \frac{k_{\text{loss}} r_{\text{wheel}}}{R_g(i) R_{\text{fin}}} \omega \tag{9}$$

where $R_g(i)$ is the gear ratio corresponding to the i-th gear, R_{fin} is the final drive ratio, r_{wheel} is the wheel radius, and k_{loss} is the drive train efficiency level [9]. Note that by using a kinematic relation for the speed engine, we are neglecting the clutch, the motor dynamic behavior, and we are assuming that the time spent for gear shift is negligible[3].

The dynamic equation of motion of the car is $m\ddot{x} = F_e - F_b - F_r$ where m (kg) is the vehicle mass, F_e (N) is the traction force, $F_b = u_b$ is the braking force (N), and F_r (N) is the friction force. As a first approximation, we assume that F_r is linear in v, $F_r = \beta v$ where β (kg·m·s^{-1}) is a constant that takes into account all

[3]This assumption is not limitative, as there are systems which able to synchronize the engine-shaft speed with the car speed and the selected gear while the clutch is not engaged, and then engage the clutch [13].

Figure 6: Renault Clio 1.9 DTI RXE (courtesy of Quattroruote, 538:88-97, August, 2000)

the frictions (i.e. aerodynamic, tires deformation, drive train).

From the conservation of mechanical power, we have $F_e v = \omega u_1$, which gives $F_e = \frac{k_{loss}}{R_g(i)} u_1$. The commanded torque u_t is upper-bounded by the maximum torque deliverable at a certain engine speed ω, $u_t(t) \leq C_e^+(\omega)$ where $C_e^+(\omega)$ is a nonlinear function typically reported in the data sheets of the car. In order to derive a hybrid model of the car as described in Section 2, we piecewise-linearize $C_e^+(\omega)$ into four regions using the PWL toolbox [15], cf. Figure 5.

To validate the model, we took the parameters of the car from `http://www.renault.com/`. In particular we chose the data from the Renault Clio 1.9 DTI RXE (Figure 6). The simulated acceleration and max speed tests gave the same results as the experimental counterpart, reported in the technical documentation. In the reader's convenience we report the main parameters of the car under consideration: $R_g(1) = 3.7271$, $R_g(2) = 2.048$, $R_g(3) = 1.321$, $R_g(4) = 0.971$, $R_g(5) = 0.756$, $R_g(R) = -3.545$, $R_{\text{fin}} = 3.2940$, $k_{\text{loss}} = 0.925$, $r_{\text{wheel}} = 0.2916$ m, $\beta = 25$ kg·m·s^{-1}, $m = 1020$ kg. Figure 5 reports the measured torque and the piece wise affine approximation, the maximum error is 5.7 Nm. Finally the dynamics is discretized with sampling time $T_s = 0.5$ s using forward finite differences to obtain the discrete-time hybrid models (1),(7).

4.2 Controller model

The controller commands throttle position, brake force, and selected gear, based on the desired vehicle speed and measurements of the actual car speed.

The automaton reported in Figure 7 selects the gear. If the engine angular speed is faster than a given threshold ω_u then the gear is shifted up. Similarly, if lower than a threshold ω_l, the gear is shifted down. The two thresholds are chosen by looking at the max torque plot in Figure 5. To track the desired speed reference $v_r(t)$, the throttle and the brakes are operated by a PI controller. Let $e(t)$ be the integral error, $e(t+1) = e(t) + T_s \Delta v(t)$,

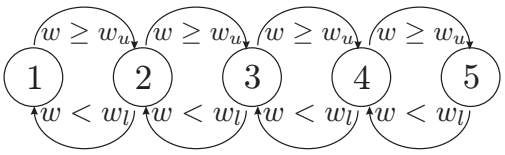

Figure 7: Gear shift logic controller

$\Delta v(t) = v_r(t) - v(t)$. The controller is

$$u_t(t) = \begin{cases} k_t \Delta v(t) + i_t e(t) & \text{if } v(t) \leq v_r(t) + 1 \\ 0 & \text{otherwise} \end{cases} \quad (10a)$$

$$u_b(t) = \begin{cases} k_b \Delta v(t) & \text{if } v(t) > v_r(t) + 1 \\ 0 & \text{otherwise} \end{cases} . \quad (10b)$$

The control variables u_t an u_b are saturated against the maximum torque and braking force, respectively. The integrator in the PI controller uses an anti-windup scheme: $e(t)$ is integrated only when the control inputs u_t and u_b are not saturated. Note that, as the threshold in equation (10), the fine tracking of the speed reference is performed using only the command coming for the throttle. By fixing the gear ratio in fifth gear we calibrate the parameters k_t, k_b, and i_t on the resulting linear system ($k_t = 70, k_b = 20$, and $i_t = 10$). The HYSDEL model of the car and the cruise control system is reported in [17].The corresponding MLD model (7) has 173 MLD constraints, $x \in \mathbb{R}^2 \times \{0,1\}^5$, $d \in \{0,1\}^{15}$, $z \in \mathbb{R}^{19}$.

4.3 Verification

The HYSDEL compiler is used to generate the MLD and PWA models of the cruise control system. The verification is performed using the algorithm recalled in Section 3.1. We check the safety using the following initial set $\mathcal{X}(0) = \{x = \left[\begin{smallmatrix} v \\ e \end{smallmatrix}\right] : v \in [0,1], e \in [0,1]\}$ and target set: $\mathcal{Z}_1 = \{v : v > v_r + r_{\text{toll}}\}$ where r_{toll} is a tolerance, in this example we set $r_{\text{toll}} = 1.3889$ m·s^{-1} (5 km/h) that is the tolerance of the speed control devices adopted in Switzerland. Moreover, we check the liveness of the controller by adding the set $\mathcal{Z}_2 = \{v, t : v \leq v_r - 2r_{\text{toll}}, t > 10/T_s\}$, where we require that the controller reaches the target speed minus the tolerance $2r_{\text{toll}}$ in 10 s (a controller that stops the car would be safe against fines, but not at all desirable!). We perform parametric verification [7] for a class of constant references $v_r \in [8.333, 19.444]$ $m/s (= [30, 70]$ $km/h)$. The exploration horizon is fixed to $T_{max} = 15.5s$.

4.4 Verification Results

The algorithm produces the result that the controlled system satisfies both the specifications: It does not enter the unsafe region $\mathcal{Z}_1$ (over the speed limit) and guarantees the liveness of the control action (within $10s$ the speed v is in a bounded set around the target speed v_r.

The verification required $9109s$ on a PC Pentium 650 MHz running Matlab 5.3.

The algorithm was run also with the same initial and

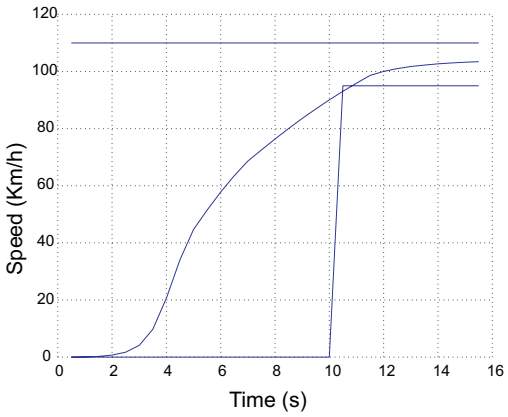

Figure 8: Counterexample to the liveness property

target sets and with an extended range for the parameter $v_r \in [8.333, 33.333] m/s$. The algorithm reported the first counterexample in $415s$: For $v_r = 29.167 m/s$ the liveness condition is not satisfied. However, by examining the plot of the counterexample reported in Figure 8, one can see that the controller fails to reach the requested vehicle speed within the specified time frame.

Acknowledgments

This research was supported by the Swiss National Science Foundation and the Federal Office for Education and Science through the Esprit Project 26270 VHS (Verification of Hybrid Systems).

References

[1] P.J. Antsaklis. A brief introduction to the theory and applications of hybrid systems. *Proc. IEEE, Special Issue on Hybrid Systems: Theory and Applications*, 88(7):879–886, July 2000.

[2] E. Asarin, O. Bournez, T. Dang, and O. Maler. Reachability analysis of piecewise-linear dynamical systems. volume 1790 of *Lecture Notes in Computer Science*, pages 20–31. Springer-Verlag, 2000.

[3] A. Bemporad, C. Filippi, and F.D. Torrisi. Inner and outer approximation of polytopes using hyper-rectangles. Technical Report AUT00-02, Automatic Control Lab, ETH Zurich, 2000.

[4] A. Bemporad, L. Giovanardi, and F.D. Torrisi. Performance driven reachability analysis for optimal scheduling and control of hybrid systems. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 969–974, Sydney, Australia, December 2000.

[5] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[6] A. Bemporad, F.D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. volume 1790 of *Lecture Notes in Computer Science*, pages 45–58. Springer Verlag, 2000.

[7] A. Bemporad, F.D. Torrisi, and M. Morari. Discrete-time hybrid modeling and verification of the batch evaporator process benchmark. *European Journal of Control*, 7(4):382–399, 2001.

[8] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, Y. Wang, and C. Weise. New Generation of UPPAAL. In *Int. Workshop on Software Tools for Technology Transfer*, June 1998.

[9] R. Bosch. *Automotive Handbook*. Society of Automotive Engineer, fifth edition, December 2000.

[10] A. Chutinan and B.H. Krogh. Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations. volume 1569 of *Lecture Notes in Computer Science*, pages 76–90. Springer-Verlag, 1999.

[11] W.P.M.H. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[12] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.

[13] Fredriksson J. and B.S. Egardt. Nonlinear control applied to gearshifting in automated manual transmissions. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 444–449, Sydney, Australia, December 2000.

[14] K.H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of Zeno hybrid automata. *System & Control Letters*, 38:141–150, 1999.

[15] P. Julián. *A High Level Canonical Piecewise Linear Representation: Theory and Applications*. PhD thesis, Univerisidad Nacional del Sur, May 1999. Toolbox available online at `http://lcr.uns.edu.ar/personales/pjulian/cpwl.htm`.

[16] J. Preussig, O. Stursberg, and S. Kowalewski. Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. volume 1569 of *Lecture Notes in Computer Science*, pages 210–222. Springer-Verlag, 1999.

[17] F.D. Torrisi and A. Bemporad. Discrete-time hybrid modeling and verification. Technical Report AUT01-17, Automatic Control Lab, ETH Zurich, 2001.

[18] F.D. Torrisi, A. Bemporad, and D. Mignone. HYSDEL - A language for describing hybrid systems. Technical Report AUT00-03, ETH Zurich, 2000. Tool available at: `http://control.ethz.ch/~hybrid/hysdel`.

[19] H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition, 1993.