Proceedings of the 41st IEEE
Conference on Decision and Control
Las Vegas, Nevada USA, December 2002

An Efficient Technique for Translating Mixed Logical Dynamical Systems into Piecewise Affine Systems

Alberto Bemporad

Dip. Ing. dell'Informazione, Università di Siena, Via Roma 56, 53100 Siena, Italy
Email: bemporad@dii.unisi.it, Web: http://www.dii.unisi.it/~bemporad

Abstract

For linear hybrid systems consisting of linear dynamic equations interacting with linear threshold events, automata, and logic propositions, provided in mixed-logical dynamical (MLD) form, this paper describes an efficient technique for transforming such systems into an equivalent piecewise affine (PWA) form, where equivalent means that for the same initial conditions and input sequences the trajectories of the system are identical.

1 Introduction

Hybrid systems provide a unified framework for describing processes evolving according to continuous dynamics, discrete dynamics, and logic rules [1,13,17,18,23]. The interest in hybrid systems is mainly motivated by the large variety of practical situations where physical processes interact with digital controllers, as for instance in embedded systems. Several modeling formalisms have been developed to describe hybrid systems [20]. Among them, the class of Piecewise Affine (PWA) systems [24], Linear Complementarity (LC) systems [19,27], and Mixed Logical Dynamical (MLD) systems [9]. In particular the language HYSDEL (HYbrid Systems DEscription Language) was developed in [26] to obtain MLD models from of a high level textual description of the hybrid dynamics. Examples of real-world applications that can be naturally modeled within the MLD framework are reported in [9,25,26].

Each subclass has its own advantages over the others. For instance, stability criteria were proposed for PWA systems [15, 22], control and state-estimation techniques for MLD hybrid models [3,8,9] and PWA models [4], and verification techniques for PWA [2,14] and MLD [11] systems. In addition, simulation of hybrid systems is much easier for PWA systems (evaluation of a PWA function per time step), than for MLD systems (one mixed-integer feasibility test per time step [9]) and LC system (one linear complementarity problem per time step).

In [5,20] we showed that MLD, PWA, LC, and other classes of hybrid systems are equivalent. Some of the equivalences were obtained under additional assumptions related to well-posedness (i.e., existence and uniqueness of solution trajectories) and boundedness of (some) system variables. These results are extremely important, as they allow to transfer all the analysis and synthesis tools developed for one particular class to any of the other equivalent subclasses of hybrid systems.

While the transformation of a PWA system into MLD form can be done immediately by using appropriate "big-M" techniques [9], the reverse transformation from MLD to PWA described in [5] requires the enumeration of all possible combinations of the integer variables contained in the MLD model (binary states and inputs and auxiliary Boolean variables). This paper provides an efficient algorithm that avoids such an enumeration and computes very efficiently the equivalent PWA form of a given MLD system. We believe that the proposed algorithm will extend the use of tools tailored for PWA systems to many real-life non-trivial hybrid problems, as those that can be described in the modeling language HYSDEL. A Matlab implementation of the technique described in this paper is available at http://www.dii.unisi.it/~hybrid/tools/mld2pwa.

2 PWA and MLD Systems

Piecewise affine (PWA) systems are described by

$$\begin{array}{rcl} x(k+1) &=& A_i x(k) + B_i u(k) + f_i \\ y(k) &=& C_i x(k) + D_i u(k) + g_i \end{array} \quad \text{for } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \Omega_i, \tag{1}$$

where $u(k) \in \mathbb{R}^m$, $x(k) \in \mathbb{R}^n$ and $y(k) \in \mathbb{R}^p$ denote the input, state and output, respectively, at time k, $\Omega_i \triangleq \{[\begin{smallmatrix} x \\ u \end{smallmatrix}] : H_{ix}x + H_{iu}u \leq K_i\}$, $i = 1, \ldots, s$, are convex polyhedra in the input+state space. A_i, B_i, C_i, D_i, H_{ix} and H_{iu} are real matrices of appropriate dimensions and f_i and g_i are real vectors for all $i = 1, \ldots, s$.

PWA systems have been studied by several authors (see [9, 15, 22, 24] and the references therein) as they form the "simplest" extension of linear systems that can still model non-linear and non-smooth processes with arbitrary accuracy and are capable of handling hybrid phenomena, such as linear-threshold events and mode switch.

A PWA system of the form (1) is called *well-posed*, if (1) is uniquely solvable in $x(k+1)$ and $y(k)$, once $x(k)$ and $u(k)$ are specified. A necessary and sufficient condition for the PWA system (1) to be well-posed over $\Omega \triangleq \cup_{i=1}^{s}\Omega_i$ is therefore that $x(k+1)$, $y(k)$ are single-valued PWA functions of $x(k)$, $u(k)$. Therefore, typically the sets Ω_i have mutually disjoint interiors, and are often defined as the partition of a convex polyhedral set Ω. In case of discontinuities of the PWA functions over overlapping boundaries of the regions Ω_i, one may ensure well-posedness by writing some of the inequalities in the form $(H_{ix})^j x + (H_{iu})^j u < K_i^j$, where ^{j} denotes the j-th row. Although this would be important from a system theoretical point of view, it is not of practical interest from a numerical point of view, as "$<$" cannot be represented in numerical algorithms working in finite precision. In the following we shall neglect this issue for the sake of compactness of notation.

2.1 Mixed Logical Dynamical (MLD) Systems

In [9] a class of hybrid systems has been introduced in which logic, dynamics and constraints are integrated. This lead to a description of the form

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \quad \text{(2a)}$$
$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \quad \text{(2b)}$$
$$E_2\delta(k) + E_3z(k) \leq E_1u(k) + E_4x(k) + E_5, \quad \text{(2c)}$$

where $x(k) = [\,x_c{}'(k)\ x_\ell{}'(k)\,]'$, $x_c(k) \in \mathbb{R}^{n_c}$ and $x_\ell(k) \in \{0,1\}^{n_\ell}$ ($y(k)$ and $u(k)$ have a similar decomposition), and where $z(k) \in \mathbb{R}^{r_c}$ and $\delta(k) \in \{0,1\}^{r_\ell}$ are auxiliary variables. A, B_i, C, D_i and E_i denote real constant matrices and E_5 is a real vector. The inequalities (2c) have to be interpreted componentwise. Systems that can be described by model (2) are called Mixed Logical Dynamical (MLD) systems.

The MLD system (2a) is called *completely well-posed*, if $\delta(k)$ and $z(k)$ are uniquely defined in their domain, once $x(k)$ and $u(k)$ are assigned [9]. From (2a)–(2b) this implies that also $x(k+1)$, $y(k)$ are uniquely defined functions of $x(k)$, $u(k)$.

The MLD formalism allows 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. The key idea of the approach consists of embedding the logic part in the state equations by transforming Boolean variables into 0-1 integers, and by expressing the relations as mixed-integer linear inequalities (see [9, 26] and references therein). MLD systems are therefore capable of modeling a broad class of systems, in particular those systems that can be modeled through the hybrid system description language HYSDEL [26].

2.2 Equivalence of MLD and PWA Systems

The following proposition has been stated in [5] and is an easy extension of the corresponding result in [9] for piecewise linear (PWL) systems (i.e. PWA systems with $f_i = g_i = 0$).

Proposition 1 *Every well-posed PWA system can be rewritten as an MLD system assuming that the set of feasible states and inputs is bounded.*

Remark 1 As MLD models only allow for nonstrict inequalities in (2c), in rewriting a discontinuous PWA system as an MLD model strict inequalities like $x(k) < 0$ must be approximated by $x(k) \leq -\varepsilon$ for some $\varepsilon > 0$ (typically the machine precision), with the assumption that $-\varepsilon < x(k) < 0$ cannot occur due to the finite number of bits used for representing real numbers (no problem exists when the PWA is continuous, where the strict inequality can be equivalently rewritten as non-strict, or $\varepsilon = 0$). See [9, 20] for more details. From a strictly theoretical point of view, the inclusion stated in Proposition 1 is therefore not exact for discontinuous PWA systems. As discussed before, one way of circumventing such an inexactness is to allow part of the inequalities in (2c) to be strict. On the other hand, from a numerical point of view this issue is not relevant. $\Box$

The reverse statement of Proposition 1 has been established in [5] under the condition that the MLD system is completely well-posed. A slightly different and more general proof is reported here below, as it will be an essential ingredient of the MLD-to-PWA translation algorithm described in Section 3.

Definition 1 *The* feasible state+input set $\Omega \subseteq \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell} \times \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ *is the set of states+inputs pairs* $(x(k), u(k))$ *for which (2c) has a solution for some* $\delta(k) \in \mathbb{R}^{r_\ell}$, $z(k) \in \mathbb{R}^{r_c}$.

Proposition 2 *Every completely well-posed MLD (2) system can be rewritten as a PWA system (1), i.e., the feasible state+input set of (2)* Ω *can be partitioned into a collection of convex polyhedra* $\{\Omega_i\}_{i=1}^{s}$, $\Omega = \bigcup_{i=1}^{s}\Omega_i$, *and there exist 5-tuples* $(A_i, B_i, C_i, f_i, g_i)$, $i = 1, \ldots, s$, *such that all trajectories* $x(k)$, $u(k)$, $y(k)$ *of the MLD system (2) also satisfy (1).*

Proof: By well-posedness of system (2), given $x(k)$ and $u(k)$ the vector $\delta(k)$ is uniquely defined, namely

$\delta(k) = F(x(k), u(k))$. The idea is to partition the space $\mathbb{R}^{n_c+m_c}$ of continuous states and inputs by grouping in regions Ω_i all $\left[\begin{smallmatrix} x_c(k) \\ u_c(k) \end{smallmatrix}\right] \in \mathbb{R}^{n_c+m_c}$ corresponding to the same logic state $x_\ell(k) = x_{\ell i} \in \{0,1\}^{n_\ell}$, binary input $u_\ell(k) = u_{\ell i} \in \{0,1\}^{u_\ell}$, and binary vector $\delta(k) = F(x(k), u(k)) \in \{0,1\}^{r_\ell}$. Let us fix $x_\ell(k) = x_{\ell i}$, $u_\ell(k) = u_{\ell i}$, $\delta(k) \equiv \delta_i$, $i = 1, \ldots, 2^{n_\ell+m_\ell+r_\ell}$. The inequalities (2c) define a polyhedron $\mathcal{P}$ in $\mathbb{R}^{n_c+m_c+r_c}$. Moreover, from (2c) it is possible to extract linear relations that involve $z(k)$, $x_c(k)$, $u_c(k)$ (for instance pairs of symmetric inequalities that correspond to linear equalities), and because of well-posedness of $z(k)$ (i.e., given a pair $x(k)$, $u(k)$ there exists only one value $z(k) \in \mathbb{R}^{r_c}$ satisfying (2c)), we obtain that there exist matrices K_{4i}, K_{1i}, K_{5i} such that

$$\begin{aligned} z(k) &= K_{4i}x_c(k) + K_{1i}u_c(k) + K_{5i}, \\ &\quad \forall x(k), u(k) : \begin{bmatrix} x_\ell(k) \\ u_\ell(k) \\ F(x(k),u(k)) \end{bmatrix} = \begin{bmatrix} x_{\ell i} \\ u_{\ell i} \\ \delta_i \end{bmatrix}, \end{aligned} \tag{3}$$

and that $\mathcal{P} \subset \mathbb{R}^{n_c+m_c+r_c}$ is a polyhedral set of dimension less than or equal to $n_c + m_c$ (for instance if $n_c = 1$, $m_c = 0$, $r_c = 1$, $\mathcal{P}$ would be a segment in $\mathbb{R}^2$). By substituting (3) in (2a)–(2b), and by partitioning $A = \left[\begin{smallmatrix} A_{cc} & A_{c\ell} \\ A_{\ell c} & A_{\ell\ell} \end{smallmatrix}\right]$, $B_1 = \left[\begin{smallmatrix} B_{1cc} & B_{1c\ell} \\ B_{1\ell c} & B_{1\ell\ell} \end{smallmatrix}\right]$, $B_2 = \left[\begin{smallmatrix} B_{2c} \\ B_{2\ell} \end{smallmatrix}\right]$, $B_3 = \left[\begin{smallmatrix} B_{3c} \\ B_{3\ell} \end{smallmatrix}\right]$, $C = \left[\begin{smallmatrix} C_{cc} & C_{c\ell} \\ C_{\ell c} & C_{\ell\ell} \end{smallmatrix}\right]$, $D_1 = \left[\begin{smallmatrix} D_{1cc} & D_{1c\ell} \\ D_{1\ell c} & D_{1\ell\ell} \end{smallmatrix}\right]$, $D_2 = \left[\begin{smallmatrix} D_{2c} \\ D_{2\ell} \end{smallmatrix}\right]$, $D_3 = \left[\begin{smallmatrix} D_{3c} \\ D_{3\ell} \end{smallmatrix}\right]$ (without loss of generality, we assumed that the continuous components of a vector are always the first), we obtain

$$\begin{aligned} x_c(k+1) &= (A_{cc} + B_{3c}K_{4i})x_c(k) + (B_{1cc} + B_{3c}K_{1i})u_c(k) + \\ &\quad + (B_{2c}\delta_i + B_{3c}K_{5i} + A_{c\ell}x_{\ell i} + B_{1c\ell}u_{\ell i}) \quad \text{(4a)} \\ x_\ell(k+1) &= (A_{\ell c} + B_{3\ell}K_{4i})x_c(k) + (B_{1\ell c} + B_{3\ell}K_{1i})u_c(k) + \\ &\quad + (B_{2\ell}\delta_i + B_{3\ell}K_{5i} + A_{\ell\ell}x_{\ell i} + B_{1\ell\ell}u_{\ell i}) \quad \text{(4b)} \\ y_c(k) &= (C_{cc} + D_{3c}K_{4i})x_c(k) + (D_{1cc} + D_{3c}K_{4i})u_c(k) + \\ &\quad + (C_{c\ell}x_{\ell i} + D_{1c\ell}u_{\ell i} + D_{3c}K_{5i} + D_{2c}\delta_i) \\ y_\ell(k) &= (C_{\ell c} + D_{3\ell}K_{4i})x_c(k) + (D_{1\ell c} + D_{3c}K_{4i})u_c(k) + \\ &\quad + (C_{\ell\ell}x_{\ell i} + D_{1\ell\ell}u_{\ell i} + D_{3\ell}K_{5i} + D_{2\ell}\delta_i), \end{aligned} \tag{4c}$$

which, by suitable a choice of A_i, B_i, C_i, f_i, g_i, corresponds to (1) for

$$\begin{aligned} \Omega_i = \{\left[\begin{smallmatrix} x_c \\ u_c \end{smallmatrix}\right] :\ & (E_3K_{4i} - E_{4c})x_c + (E_3K_{1i} - E_{1c})u_c \le \\ & \le (E_{1\ell}u_{\ell i} - E_2\delta_i - E_3K_{5i} + E_{4\ell}x_{\ell i} + E_5)\} \times \{x_{\ell i}\} \times \{u_{\ell i}\}, \end{aligned} \tag{5}$$

where $E_1 = [E_{1c}\ E_{1\ell}]$, $E_4 = [E_{4c}\ E_{4\ell}]$. ■

Note that the well-posedness of the original MLD system implies that the $x_\ell(k+1)$ and $y_\ell(k)$ mappings in (4) are $\{0,1\}$-valued. We also remark that in general the feasible state+input set of (2) $\Omega = \bigcup_{i=1}^{s} \Omega_i$ is nonconvex.

3 Translation Algorithm

For any given MLD system, Proposition 2 is constructive, as it returns the equivalent PWA system. However, it is based on the enumeration of all $2^{n_\ell+m_\ell+r_\ell}$ combinations of binary (x_ℓ, u_ℓ, δ) variables. In general, most combinations lead to empty regions Ω_i in (5), and a method that avoid the enumeration of all possibilities is therefore desirable. In this paper, we propose to avoid such an enumeration by using techniques from multiparametric programming [10,16]. In particular, the idea is to determine a feasible combination (x_ℓ, u_ℓ, δ) via mixed-integer linear programming (MILP), for which several solvers exist (see e.g., [7,21]), generate the corresponding polyhedral cell Ω_i, and then partition and explore the rest of the state+input space recursively.

Before proceeding further, we first embed the sets Ω_i in $\mathbb{R}^{n+m}$ by treating the integer vectors x_ℓ, u_ℓ as real-valued vectors during the exploration of the state+input set. In particular, we replace the set $\{0,1\}$ with $[-1/2, 1/2) \cup [1/2, 3/2]$.

Let (x^*, u^*) a given point in $\mathbb{R}^{n+m}$. Usually, an information is available a priori on an over approximation $\mathcal{B}$ of Ω, as generally MLD models are obtained by HYSDEL through the application of the so-called "big-M" technique, which requires the specification of upper and lower bounds on state and input variables [9,26,28], see also the example in Appendix A. By letting $\Omega \subseteq \mathcal{B}$ (for example $\mathcal{B}$ can be an box), a good choice for (x^*, u^*) is the Chebychev center of $\mathcal{B}$ (the center of the largest Euclidean ball contained in $\mathcal{B}$). In general, for a polyhedron $\bar{\mathcal{P}} = \{\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] : \bar{A}\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \le \bar{B}\}$, its Chebychev center can be determined by solving the linear program (LP) [12, Chapter 3]

$$\begin{aligned} \max_{x,u,\epsilon} \quad & \epsilon \\ \text{subj. to} \quad & \bar{A}^i \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] + \epsilon\|\bar{A}^i\| \le \bar{B}^i, \end{aligned} \tag{6}$$

where the optimizer (x^*, u^*) is the center, and ϵ^* is the radius of the Chebychev ball ($\epsilon^* < 0$ means that $\bar{\mathcal{P}}$ is empty). As (x^*, u^*) may be not integer feasible (i.e., its components x_ℓ, u_ℓ are not in $\{0,1\}$), we find the the best approximation (in infinity-norm) $(x_1, u_1) = (\left[\begin{smallmatrix} x_{c1} \\ x_{\ell 1} \end{smallmatrix}\right], \left[\begin{smallmatrix} u_{c1} \\ u_{\ell 1} \end{smallmatrix}\right])$ of (x^*, u^*) which is integer feasible and satisfies the MLD constraints (2c), by solving the MILP

$$\begin{aligned} \min_{\substack{x, u, \delta, z, \sigma \\ x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell} \\ u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell} \\ \delta \in \{0,1\}^{r_\ell},\ z \in \mathbb{R}^{r_c},\ \sigma \in \mathbb{R}}} \quad & \sigma \\ \text{subj. to} \quad & \sigma e_{n+m} \ge \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] - \left[\begin{smallmatrix} x^* \\ u^* \end{smallmatrix}\right] \\ & \sigma e_{n+m} \ge -\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] + \left[\begin{smallmatrix} x^* \\ u^* \end{smallmatrix}\right] \\ & E_2\delta + E_2 z \le E_1 u + E_4 x + E_5 \\ & \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{B}, \end{aligned} \tag{7}$$

where $e_{n+m} = [1 \dots 1]' \in \mathbb{R}^{n+m}$, and σ represents an upper bound on $\| \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] - \left[\begin{smallmatrix} x^* \\ u^* \end{smallmatrix}\right] \|_\infty$. If the MILP (7) is infeasible, then $\Omega = \emptyset$, which means that the MLD system is badly posed (for all initial states $x(0)$ no input $u(0)$ exists which provides a successor $x(1)$ and fulfills the MLD dynamics). Otherwise, let δ_1 be the corresponding optimal δ for problem (7), and for the triple $(x_{\ell 1}, u_{\ell 1}, \delta_1)$ compute the corresponding linear expression for z according to (3), the 5-tuple $(A_1, B_1, C_1, f_1, g_1)$ according to (4), and the corresponding region Ω_1 where such 5-tuple is valid according to (5), with the assumption that $x^i_{\ell 1} = 0$ is represented by $x^i_{\ell 1} \in [-1/2, 1/2]$, $x^i_{\ell 1} = 1$ is represented by $x^i_{\ell 1} \in [1/2, 3/2]$, where $x^i_{\ell 1}$ denotes the i-th component of $x_{\ell 1}$, and similarly for the components $u^i_{\ell 1}$ of the logic part of the input vector $u_{\ell 1}$. Clearly, Ω_1 is a polyhedron in $\mathbb{R}^{n+m}$ and represents the first region of the equivalent PWA system.

The nonconvex rest $\mathcal{B}^{n+m} \backslash \Omega_1$ is partitioned into convex polyhedral cells R_j, $j = 1, \dots, p_0$, in accordance with the following theorem (cf. [10, Theorem 3]):

Theorem 1 *Let $P \subseteq \mathbb{R}^{n+m}$ be a polyhedron, and let $\Theta = \{\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in P : G \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \le g\}$ be a nonempty polyhedral subset of P, where $G \in \mathbb{R}^{p \times (n+m)}$. Also let*

$$R_j = \left\{ \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in P : \begin{array}{l} G^j \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] > g^j \\ G^h \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \le g^h, \forall h < j \end{array} \right\}, \quad j = 1, \dots, p_0,$$

where G^j denotes the j-th row of G and g^j denotes the j-th entry of g. Then (i) $P = \left(\cup_{j=1}^{p} R_j\right) \cup \Theta$; (ii) $\Theta \cap R_j = \emptyset$ for all j and $R_j \cap R_h = \emptyset$ for all $j \neq h$; i.e., $\{\Theta, R_1, \dots, R_p\}$ is a partition of P.

After partitioning the rest of the space, we proceed recursively: we choose for each region R_i a new vector (x^*, u^*) by solving the LP (6), with $\bar{A}$, $\bar{B}$ such that $\{\left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] : \bar{A} \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \le \bar{B}\} = R_i$, and solve the MILP (7). If the MILP is infeasible, the region R_i is discarded. If the optimal solution (x_ℓ, u_ℓ, δ) provides a new combination, then (3), (4), and (5) are computed to calculate a new affine dynamics and polyhedral cell Ω_i. Then, Theorem 1 is applied with $P = R_i$, $\Theta = R_i \cap \Omega_i$ (clearly, in order to minimize the number p_0 of regions R_i generated at each recursion, before applying Theorem 1 it is convenient to remove all redundant inequalities from the representation of Θ), and the algorithm proceeds recursively.

At the end of the recursion, in a post-processing operation, in order to reduce the number of polyhedral cells in the PWA system we check all pairs of regions in the state+input space $\mathbb{R}^{n+m}$ where the affine dynamics (for both the continuous and logic components of the state and output vectors) are the same, and try to compute their union, provided that the union is a convex set [6].

In summary, the translation algorithm from MLD to PWA is described by the following algorithm:

Algorithm 1

```
1. Let $\mathcal{B}$ be a polytope in $\mathbb{R}^{n+m}$;
2. Find the Chebychev center $(x^*, u^*)$ of the current region $\mathcal{B}$ in the $(x, u)$-space using (6);
3. Find the best integer-feasible approximation $(x_i, u_i)$ of $(x^*, u^*)$ that satisfies the MLD constraints (2c), via the MILP (7);
4. If the MILP is infeasible, terminate the exploration of the current region $\mathcal{B}$;
5. Let $(x_\ell, u_\ell, \delta)$ be the optimal solution to the MILP;
6. If $(x_\ell, u_\ell, \delta)$ is a new combination, compute (3), (4), and $\Omega_i$ from (5);
7. Remove redundant inequalities from $\Theta = \mathcal{B} \cap \Omega_i$, where $x^i_\ell = 0$ is replaced by $x^i_\ell \in [-1/2, 1/2]$, $x^i_\ell = 1$ by $x^i_{\ell 1} \in [1/2, 3/2]$, and similarly for $u^i_\ell$;
8. Apply Theorem 1 with $P = \mathcal{B}$ and generate new polyhedral cells $R_1, \dots, R_p$;
9. For each polyhedron $R_i$, let $\mathcal{B} \leftarrow R_i$ and go to 2.;
10. At the end of the recursion, consider all pairs of regions where the dynamics (both continuous and logic) is the same, and try to compute their union [6];
11. End
```

We remark that after the PWA form has been generated, an optimized MLD with minimum number of integer variables can be easily obtained by efficiently coding the PWA dynamics as described in [9], for instance by using $\lceil \log_2 s \rceil$ integer variables, where $s \le 2^{n_\ell + m_\ell + r_\ell}$ is the number of regions in the PWA partition generated by Algorithm 1.

4 An Example

We consider the hybrid model of a car with robotized manual gear shift reported in [25], for which a cruise control was verified. The model was developed in HYSDEL (see Appendix A) and the corresponding MLD model has $n = 2$ continuous states (position and velocity of the car), no logic state, 2 continuous inputs (engine torque and braking force), 6 binary inputs (gears #1,...,#5 + reverse gear), 4 auxiliary binary variables and 16 auxiliary variables, 96 mixed-integer inequalities. The total number of binary variables is $0 + 6 + 4 = 10$, which gives a worst-case number of possible regions in the PWA system equal to $2^{10} = 1024$. By running the algorithm proposed in this paper, the

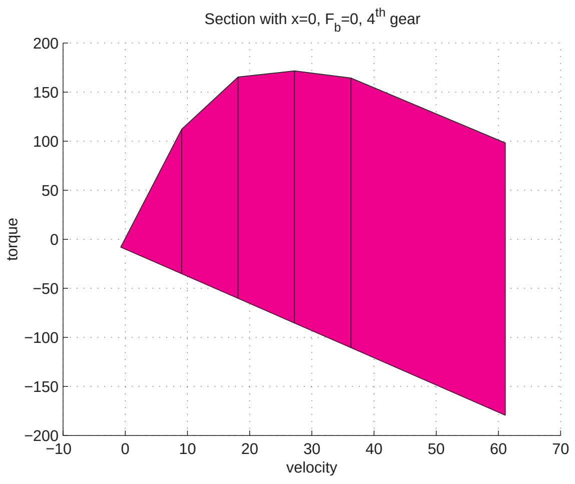

Figure 1: PWA system equivalent to the MLD model obtained through HYSDEL from the list reported in Appendix A – Section in the (velocity,torque) space for position $x = 0$, braking force $F_b = 0$, gear input vector $= [0\ 0\ 0\ 0\ 1\ 0]'$ (4th gear)
.

PWA equivalent to the hybrid MLD model has 28 regions, and is computed in 72.66 s in Matlab 5.3 on a Pentium III 650 MHz machine.

In order to compare the simulation time required by the original MLD system and its PWA equivalent, we simulate both systems over 300 steps. While the MLD system takes 20.20 s by using the MILP solver [7], the same simulation takes 0.94 s by using the PWA equivalent system.

5 Conclusions

We have described an efficient algorithm for translating hybrid systems expressed as mixed-logical dynamical system into an equivalent piecewise affine system, where equivalence means that the same initial conditions and inputs produce the identical trajectories. We believe that the result is very useful to apply several techniques available for PWA systems (stability analysis via piecewise quadratic Lyapunov functions, controller synthesis, verification, simulation) to relatively complex hybrid systems composed by linear dynamics, automata, propositional logic, linear threshold conditions, and if-then-else rules, such as those described by the modeling language HYSDEL [26].

Acknowledgements

The author would like to thank Fabio D. Torrisi and Maurice Heemels for stimulating discussions about some of the ideas reported in this paper.

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. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 20–31. Springer-Verlag, 2000.

[3] A. Bemporad, F. Borrelli, and M. Morari. Piecewise linear optimal controllers for hybrid systems. In *Proc. American Contr. Conf.*, pages 1190–1194, Chicago, IL, June 2000.

[4] A. Bemporad, F. Borrelli, and M. Morari. On the optimal control law for linear discrete time hybrid systems. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer Verlag, 2002.

[5] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Trans. Automatic Control*, 45(10):1864–1876, 2000.

[6] A. Bemporad, K. Fukuda, and F.D. Torrisi. Convexity recognition of the union of polyhedra. *Computational Geometry*, 18:141–154, 2001.

[7] A. Bemporad and D. Mignone. *MIQP.M: A Matlab function for solving mixed integer quadratic programs.* ETH Zurich, 2000. Code available at `http://control.ethz.ch/~hybrid/miqp`.

[8] A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proc. American Contr. Conf.*, pages 2471–2475, Chicago, IL, June 1999.

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

[10] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[11] A. Bemporad, F.D. Torrisi, and M. Morari. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 45–58. Springer Verlag, 2000.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization.* To be published in 2003. `http://www.stanford.edu/class/ee364/index.html#book`.

[13] M.S. Branicky. *Studies in hybrid systems: modeling, analysis, and control.* PhD thesis, LIDS-TH 2304, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[14] A. Chutinan and B. H. Krogh. Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations. In F. Vaandrager and J. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 76–90. Springer-Verlag, 1999.

[15] R. DeCarlo, M. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.

[16] T. Gal. *Postoptimal Analyses, Parametric Programming, and Related Topics.* de Gruyter, Berlin, 2nd edition, 1995.

[17] K. Gokbayrak and C.G. Cassandras. A hierarchical decomposition method for optimal control of hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 1816–1821, Phoenix, AZ, December 1999.

[18] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 3972–3976, Phoenix, AZ, December 1999.

[19] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000.

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

[21] ILOG, Inc. *CPLEX 7.0 User Manual*. Gentilly Cedex, France, 2000.

[22] M. Johannson and A. Rantzer. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. *IEEE Trans. Automatic Control*, 43(4):555–559, 1998.

[23] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[24] E.D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automatic Control*, 26(2):346–358, April 1981.

[25] F.D. Torrisi and A. Bemporad. Discrete-time hybrid modeling and verification. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 2899–2904, Orlando, Florida, 2001.

[26] F.D. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. Technical Report AUT02-03, ETH Zurich, Submitted for publication, 2002. http://control.ethz.ch/~hybrid/hysdel.

[27] A.J. van der Schaft and J.M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Trans. Automatic Control*, 43:483–490, 1998.

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

A Hysdel Model

```
/* Hybrid model  - Renault Clio 1900 */

SYSTEM car {

INTERFACE {
    STATE {
        REAL position [-1000, 1000];                /* position      */
        REAL speed    [-50*1000/3600, 220*1000/3600]; /* velocity (m/s)*/
        }

    INPUT {
        REAL torque  [-300,300]; /* Nm */
        REAL F_brake [0,9000];   /* N  */
        BOOL gear1, gear2, gear3, gear4, gear5, gearR;
        }

    PARAMETER {
        REAL mass = 1020; /* kg */
        REAL beta_friction = 25; /* W/m*s */
        REAL Rgear1 = 3.7271;
        REAL Rgear2 = 2.048;
        REAL Rgear3 = 1.321;
        REAL Rgear4 = 0.971;
        REAL Rgear5 = 0.756;
        REAL RgearR = -3.545;

        REAL wheel_rim = 14;        /* in */
        REAL tire_width = 175;      /* mm */
        REAL tire_height_perc = 65; /* % */
        REAL R_final = 3.294;
        REAL loss = 0.925; /* correction term for losses */

        REAL pi = 3.1415; REAL inch = 2.54;

        /* wheel radius: */
        REAL wheel_radius = (wheel_rim/2*inch+(tire_width/10)*
                             (tire_height_perc/100))/100;

        /* speed=speed_factor*w_engine/Rgear */
        REAL speed_factor = loss /R_final * wheel_radius;
        REAL max_brake = 8.53; /* max acceleration (m/s^2) */
        REAL max_brake_force = mass*max_brake; /* max braking force */
        REAL wemin = -100*2*pi/60;
        REAL wemax = 6000*2*pi/60;
        REAL Ts = 0.5; /* sampling time, seconds */
        /* torque nonlinearity:
           C(w)=aPWL(i)+bPWL(i)*w, w\in\[wPWL(i),wPWL(i+1)] rad/s */
        REAL aPWL1 = 0;       REAL aPWL2 = 58.1070;
        REAL aPWL3= 151.7613; REAL aPWL4 =192.8526; REAL aPWL5=259.9484;
        REAL bPWL1 = 1.3281;  REAL bPWL2 = 0.6344;
        REAL bPWL3 =  0.0755; REAL bPWL4 = -0.0880; REAL bPWL5=-0.2883;
        /* breakpoints */
        REAL wPWL1 = 83.7733; REAL wPWL2 =  167.5467;
        REAL wPWL3 = 251.32;  REAL wPWL4=335.0933;

        /* Engine brake torque */
        REAL alpha1 = 10;
        REAL beta1 = 0.3;}
}

IMPLEMENTATION {
   AUX {REAL Fe1, Fe2, Fe3, Fe4, Fe5, FeR;
        REAL w1, w2, w3, w4, w5, wR;
        BOOL dPWL1,dPWL2,dPWL3,dPWL4;
        REAL DCe1,DCe2,DCe3,DCe4; }

   AD { dPWL1 = wPWL1-(w1+w2+w3+w4+w5+wR)<=0;
        dPWL2 = wPWL2-(w1+w2+w3+w4+w5+wR)<=0;
        dPWL3 = wPWL3-(w1+w2+w3+w4+w5+wR)<=0;
        dPWL4 = wPWL4-(w1+w2+w3+w4+w5+wR)<=0;}


   DA {Fe1 = {IF gear1  THEN torque/speed_factor*Rgear1};
       Fe2 = {IF gear2  THEN torque/speed_factor*Rgear2};
       Fe3 = {IF gear3  THEN torque/speed_factor*Rgear3};
       Fe4 = {IF gear4  THEN torque/speed_factor*Rgear4};
       Fe5 = {IF gear5  THEN torque/speed_factor*Rgear5};
       FeR = {IF gearR  THEN torque/speed_factor*RgearR};

       w1 = {IF gear1  THEN speed/speed_factor*Rgear1};
       w2 = {IF gear2  THEN speed/speed_factor*Rgear2};
       w3 = {IF gear3  THEN speed/speed_factor*Rgear3};
       w4 = {IF gear4  THEN speed/speed_factor*Rgear4};
       w5 = {IF gear5  THEN speed/speed_factor*Rgear5};
       wR = {IF gearR  THEN speed/speed_factor*RgearR};

       DCe1 = {IF dPWL1 THEN (aPWL2-aPWL1)+(bPWL2-bPWL1)*
                                 (w1+w2+w3+w4+w5+wR)};
       DCe2 = {IF dPWL2 THEN (aPWL3-aPWL2)+(bPWL3-bPWL2)*
                                 (w1+w2+w3+w4+w5+wR)};
       DCe3 = {IF dPWL3 THEN (aPWL4-aPWL3)+(bPWL4-bPWL3)*
                                 (w1+w2+w3+w4+w5+wR)};
       DCe4 = {IF dPWL4 THEN (aPWL5-aPWL4)+(bPWL5-bPWL4)*
                                 (w1+w2+w3+w4+w5+wR)};}

    CONTINUOUS { position = position+Ts*speed;
      speed = speed+Ts/mass*(Fe1+Fe2+Fe3+Fe4+Fe5+FeR-
                          F_brake-beta_friction*speed);}

    MUST   {
      -w1 <= -wemin;
      w1 <= wemax;
      -w2 <= -wemin;
      w2 <= wemax;
      -w3 <= -wemin;
      w3 <= wemax;
      -w4 <= -wemin;
      w4 <= wemax;
      -w5 <= -wemin;
      w5 <= wemax;
      -wR <= -wemin;
      wR <= wemax;

      -F_brake <=0; /* brakes cannot accelerate ! */
      F_brake <= max_brake_force;

      /* Commanded torque between Cb(we) and Ce(we)*/
      -torque-(alpha1+beta1*(w1+w2+w3+w4+w5+wR)) <=0;
      torque-(aPWL1+bPWL1*(w1+w2+w3+w4+w5+wR)+DCe1+DCe2+DCe3+DCe4)-1<=0;

      -((REAL gear1)+(REAL gear2)+(REAL gear3)+(REAL gear4)+
                 (REAL gear5)+(REAL gearR))<=-0.9999;
      (REAL gear1)+(REAL gear2)+(REAL gear3)+(REAL gear4)+
                 (REAL gear5)+(REAL gearR)<=1.0001;

       dPWL4 -> dPWL3;
       dPWL4 -> dPWL2;
       dPWL4 -> dPWL1;
       dPWL3 -> dPWL2;
       dPWL3 -> dPWL1;
       dPWL2 -> dPWL1;}
  }
}
```