International Journal of Control
Vol. 00, No. 00, Month 200x, 1–22

Efficient Mode Enumeration of Compositional Hybrid Systems

Tobias Geyer*, Fabio D. Torrisi and Manfred Morari

Automatic Control Laboratory, ETH Zurich, 8092 Zurich, Switzerland
(*Received 00 Month 200x; final version received 00 Month 200x*)

A hyperplane arrangement is a polyhedral cell complex where the relative position of each cell of the arrangement and the composing hyperplanes are summarized by a sign vector computable in polynomial time. This tool from computational geometry enables the development of a fast and efficient algorithm to translate the composition of discrete-time linear hybrid systems into an equivalent piecewise affine model and to determine if the composition is well-posed. The tool provides also information on the real combinatorial degree of the system which can be used to reduce the size of the search tree and the computation time of the optimization algorithms underlying optimal and model predictive control. Examples are presented illustrating the algorithm and showing its computational effectiveness.

Keywords: Hybrid system, composition, hyperplane arrangement, mode enumeration, discrete hybrid automaton

1 Introduction

A *hybrid system* is a collection of digital finite state machines interacting among each other and with an analog environment. Each *logic state* of the digital part of the hybrid system acts on the analog part inducing a different operational *mode*. On the other hand, the evolution of the analog part triggers switches in the states of the digital part. A mode in a hybrid system, which is sometimes also referred to as a *discrete state* or *location*, is characterized by a combination of logic states, events and logic inputs. A mode is feasible, if and only if its associated set of analog (continuous-valued) states is non-empty, else it is infeasible.

The practical relevance of hybrid systems is twofold. Digital controllers embedded in a continuous environment require adequate modelling, analysis and design tools for instance in the automotive industry (Balluchi, Benvenuti, Di Benedetto, Pinello, and Sangiovanni-Vincentelli 2000). Moreover, many physical phenomena admit a natural hybrid description, like power electronics integrating switched semiconductors and diodes, biomolecular networks and TCP/IP networks (Hespanha, Bohacek, Obraczka, and Lee 2001).

Hybrid systems can be composed to form *compositional hybrid systems* (Alur and Henzinger 1997, Rashid and Lygeros 1999, Benvenuti, Ferrari, Mazzi, and Sangiovanni-Vincentelli 2008). In general, the resulting system is very complex, as the number of possible modes depends exponentially on the number of component systems. The explosion of the size of the logic state leads to computational difficulties as the time and space complexity of many algorithms depends on the number of possible modes.

In some cases, the composition induces a structure that can be exploited, like in hierarchical hybrid systems (Alur et al. 2001). This allows one to break down the problem into pieces and to apply the assume-guarantee approach (Alur and Henzinger 1997, Henzinger, Minea, and Prabhu 2001). Recognizing that a system can be modelled as a hierarchical hybrid system is part of the "art" of model building. In many cases this may not be possible at all, because the ties between the components are too tight. On the other hand, tight interactions — which are effectively

*Corresponding author. Email: t.geyer@ieee.org

ISSN: 0020-7179 print/ISSN 1366-5820 online

DOI: 10.1080/0020717YYxxxxxxxx
http://www.informaworld.com

constraints — often render many modes infeasible (as will be shown later in an example) and the complexity of the system can be reduced by explicitly computing and taking into account only the feasible modes.

This paper focuses on *Discrete Hybrid Automata* (DHA) (Torrisi and Bemporad 2004), a discrete-time linear hybrid system framework where the analog environment evolves according to affine (linear plus offset) difference equations (affine dynamics), while affine threshold conditions on the analog variables drive the digital part (a finite state machine). DHAs are a mathematical abstraction of the features provided by other computation oriented and domain specific discrete-time linear hybrid system frameworks including *Mixed Logical Dynamical* (MLD) systems (Bemporad and Morari 1999), polyhedral *Piecewise Affine* (PWA) systems (Sontag 1981), *Linear Complementarity* (LC) systems (van der Schaft and Schumacher 1998), *Extended Linear Complementarity* (ELC) systems (Heemels, De Schutter, and Bemporad 2001), and *Max-Min-Plus-Scaling* (MMPS) systems (Heemels et al. 2001). As shown first in Sontag (1996) for PWA and a class of hybrid systems and then — with different arguments — in Heemels et al. (2001) and Torrisi and Bemporad (2004) all those modelling frameworks are equivalent and it is possible to represent the same system using any of these frameworks. Thus DHAs generalize many computation oriented discrete-time models for hybrid systems and therefore represent a universal starting point for solving complex analysis and synthesis problems for hybrid systems.

DHAs can be connected in an arbitrary way to form *compositions of hybrid systems*. Specifically, parallel, cascaded and (nested) feedback structures with algebraic loops are addressed in this paper. Connections encompass both real as well as binary variables. Since the thresholds (or guards) in a DHA are defined by hyperplanes, the enumeration of the DHA's feasible modes is easily solvable by using algorithms that compute the cells of a hyperplane arrangement (Ferrez, Fukuda, and Liebling 2001). This is a classical problem in computational geometry (Buck 1943) and admits optimal (Edelsbrunner 1987) and efficient (Avis and Fukuda 1996, Ferrez et al. 2001) algorithms.

With regards to compositions, an algorithm is proposed to sequentially enumerate the modes of a compositional hybrid system according to the interactions and dependencies of the DHAs in the composition. This algorithm is *efficient* for two reasons. First, to enumerate the modes of one DHA, the notion of cells in a hyperplane arrangement and reverse search are used, which scales polynomially rather than exponentially with the number of possible modes. Second, when considering multiple DHAs forming a composition, the mode enumeration is run sequentially according to a computational order in which the DHAs are arranged. This is in contrast to a brute-force approach, where every possible mode in a composition is investigated, out of which the majority is infeasible due to the interactions amongst the DHAs and additional constraints. Exploiting the structure of the composition, the sequential enumeration greatly reduces the number of modes considered by pruning branches with infeasible modes from the search tree.

The impact of the mode enumeration on applications is threefold. First, at the modelling stage, the enumeration of modes allows the designer to understand the real complexity of the compound model. Second, after the modelling stage, the model can be efficiently translated into a PWA representation. This operation is trivial once all modes have been enumerated. Furthermore, the PWA representation of the compound DHA model allows one to determine if the model is well-posed, i.e. if for all initial conditions and all inputs, the state and output trajectories are uniquely defined for all time-steps. In fact, a composition of well-posed DHAs is not necessarily well-posed as a whole. Third, during the computational stage (i.e. analysis and control), the explicit computation of the set of feasible modes of the compound system allows one to prune unnecessary modes from the resulting system and to reduce the combinatorial explosion of the underlying (optimization) algorithms. This is of particular importance for *Model Predictive Control* (MPC) of hybrid models (Bemporad and Morari 1999), where the aim is to compute the next N control inputs that optimize a given performance index defined on the variables of a hybrid *prediction model*. The prediction model is the series connection of N identical single-step prediction models, where each model uses the state predicted by the previous model as

initial state. Using the mode enumeration technique, cuts in the form of integer constraints can be added to the prediction model. In general, these cuts greatly reduce the size of the integer search tree of the underlying optimization problem.

With regards to translating hybrid systems into an equivalent PWA form, two related approaches (Bemporad 2002) and (Villa, Duque, Gauthier, and Rakoto-Ravalontsalama 2004) are available in the literature. Both algorithms transform MLD models into equivalent PWA descriptions, while the approach presented here is applicable to DHAs. Yet, as shown in Torrisi and Bemporad (2004), DHAs can be automatically translated into MLD models using the tool HYSDEL (Torrisi and Bemporad 2004) and most of the MLD models presented in the literature were derived from DHA descriptions using HYSDEL (Borrelli, Bemporad, Fodor, and Hrovat 2001, Earl and D'Andrea 2002). The main difference is that the approach in Bemporad (2002) is based on multi-parametric programming and mixed integer linear programming. Similarly, the technique described in Villa et al. (2004) relies on linear programming. In contrast to that, the approach presented in this paper computes the cells of a hyperplane arrangement and is applicable to DHAs rather than MLD models. Most importantly, using the structural information only available in the HYSDEL code (and not in the corresponding MLD model) allowed us to design a conversion tool that is faster by at least an order of magnitude compared with its two counterparts (Bemporad 2002, Villa et al. 2004).

Neither of the two papers above nor this one here aim to derive PWA models of minimal complexity, i.e. with the minimal number of polyhedra. Since this problem is $\mathcal{NP}$-hard, such an algorithm would not be efficient. Yet, as shown in Geyer, Torrisi, and Morari (2008), optimal complexity reduction algorithms can be designed based on cells in an hyperplane arrangement.

This paper extends preliminary results that appeared in Geyer, Torrisi, and Morari (2003) by presenting the algorithms in greater detail and elaborating on algebraic loops in compositions of DHAs. Also Potocnik et al. (2004) re-state the results of Geyer et al. (2003) claiming some minor improvements in the way the initial algorithm was implemented without adding new or original insight.

The paper is organized as follows: Section 2 presents the notion of cell enumeration in hyperplane arrangements. In Section 3, the classes of DHAs and PWA systems are recapitulated and the equivalence of both representations is shown in a constructive way by leveraging the tools presented in Section 2. Based on this, Section 4 presents an algorithm that enumerates the modes of a composition of DHAs, transforms it into an equivalent PWA representation and checks its well-posedness. In Section 5, this mode enumeration algorithm is applied to two examples described in HYSDEL. It is shown that the information collected during the mode enumeration step allows one to either build efficiently an equivalent PWA model or to shorten the computation time for solving optimal control problems by adding cuts (integer constraints) to the optimization problem. Section 6 summarizes the results and points out some future research directions.

The mode enumeration algorithm is implemented in MATLAB and assumes that the composition of DHAs is given as HYSDEL code. The latest version is integrated in the Multi-Parametric Toolbox (Kvasnica, Grieder, Baotić, and Morari 2004), while an older version of the code can also be downloaded from `http://control.ethz.ch/~hybrid/hysdel`.

2 Cell Enumeration in Hyperplane Arrangements

Let $\mathcal{A}$ be a collection of n distinct hyperplanes $\{\mathcal{H}_i\}_{i=1,\dots,n}$ in the d-dimensional Euclidian space $\mathbb{R}^d$, where each hyperplane is given by the linear equality $\mathcal{H}_i = \{z \in \mathbb{R}^d \mid a_i^T z = b_i\}$. We say that the hyperplanes of $\mathcal{A}$ are in *general position*, if there exists no pair of parallel hyperplanes, and if any point of $\mathbb{R}^d$ belongs at most to d hyperplanes. Let $\mathrm{SV} : \mathbb{R}^d \to \{-,+\}^n$ be the simplified

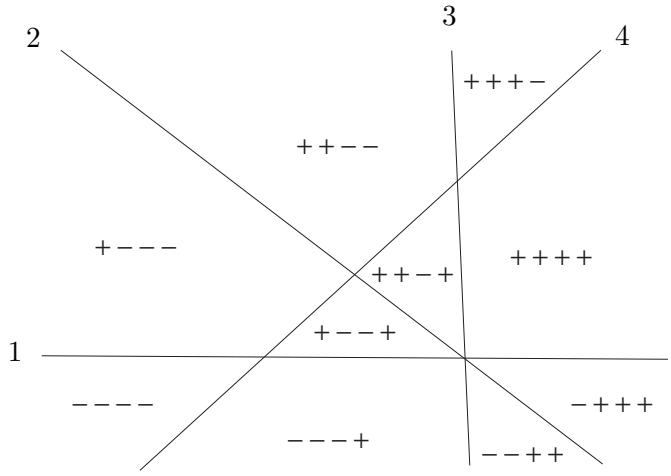


Figure 1. Arrangement of four hyperplanes (lines) in $\mathcal{R} = \mathbb{R}^2$ with markings $m \in M(\mathcal{R})$

sign vector[1] defined as

$$\mathrm{SV}_i(z) = \begin{cases} - & \text{if } a_i^T z \leq b_i, \\ + & \text{if } a_i^T z > b_i \end{cases} \quad \text{for } i \in \{1, 2, \ldots, n\}. \tag{1}$$

Consider the set $\mathcal{P}_m = \{z \in \mathbb{R}^d \mid \mathrm{SV}(z) = m\}$ for a given sign vector m. This set is called a *cell* of the arrangement and is a polyhedron as it is defined by linear inequalities. We will refer to m as the *marking* of the polyhedron (or cell) $\mathcal{P}_m$ in the *hyperplane arrangement* $\mathcal{A}$ (see Fig. 1). Let $M(\mathcal{R})$ be the image of the function $\mathrm{SV}(z)$ for $z \in \mathcal{R} \subseteq \mathbb{R}^d$, namely the collection of all the possible markings of all the points in $\mathcal{R}$.

The cell enumeration problem in a hyperplane arrangement amounts to enumerating all the elements of the set $M(\mathcal{R})$. Let $\#M(\mathcal{R})$ denote the number of cells identified by $M(\mathcal{R})$. Buck's formula (Buck 1943) defines the upper bound

$$\#M(\mathcal{R}) \leq \sum_{i=0}^{d} \binom{n}{i} = O(n^d), \tag{2}$$

with the equality satisfied if the hyperplanes are in general position and $\mathcal{R} = \mathbb{R}^d$.

The cell enumeration problem admits an optimal solution with time and space complexity $O(n^d)$ (Edelsbrunner 1987). An alternative approach based on reverse search was presented in Avis and Fukuda (1996) and improved in Ferrez et al. (2001). Reverse search is an exhaustive search technique that can be considered as a special graph search.

Proposition 2.1: *(Ferrez et al. 2001, Theorem 4.1) There exists a reverse search algorithm for enumerating hyperplane arrangements that runs in $O(n\,\mathrm{lp}(n,d)\,\#M(\mathcal{R}))$ time and $O(n,d)$ space, where $\mathrm{lp}(n,d)$ denotes the complexity of solving a Linear Program (LP) with n constraints and d variables.*

Note that in many cases of interest, the hyperplanes are not in general position and $\#M(\mathcal{R})$ is considerably smaller than the theoretical upper bound.

The following proposition follows directly from the definition of $\mathcal{P}_m$ and (1).

[1]Note that in general, the sign vector is defined such that its image is $\{-, 0, +\}$, where the '0' element corresponds to $a_i^T z = b_i$. Cells with '0' markings are lower-dimensional and not meaningful in the context of PWA systems.

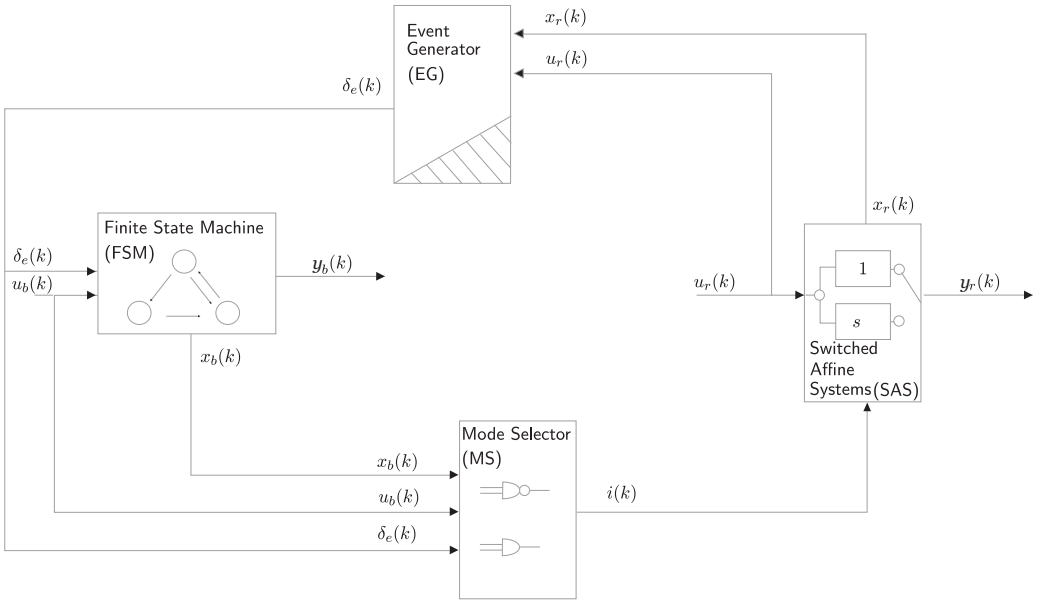


Figure 2. A Discrete Hybrid Automaton (DHA) is the connection of a Finite State Machine (FSM) and a Switched Affine System (SAS) through a Mode Selector (MS) and an Event Generator (EG)

Proposition 2.2: *The collection of polyhedral sets* $\{\mathcal{P}_m\}_{m\in M(\mathcal{R})}$ *satisfies:*

$$(i)\quad \bigcup_{m\in M(\mathcal{R})} \mathcal{P}_m = \mathcal{R}, \qquad (ii)\quad \mathcal{P}_i \cap \mathcal{P}_j = \emptyset,\ \forall i,j \in M(\mathcal{R}),\ i \neq j$$

Definition 2.3: A collection of polyhedral sets that satisfies points *(i)* and *(ii)* in Proposition 2.2 is a *polyhedral partition* of the polyhedral set $\mathcal{R}$.

3 Discrete-Time Linear Hybrid Models

3.1 *Discrete Hybrid Automata*

As depicted in Fig. 2 and shown in Torrisi and Bemporad (2004), Discrete Hybrid Automata (DHA) result from the interconnection of a *Finite State Machine* (FSM), which provides the discrete part of the hybrid system, with a *Switched Affine System* (SAS) providing the continuous part of the system. The interaction between the two is based on two connecting elements: The *Event Generator* (EG) and the *Mode Selector* (MS). The EG extracts binary signals from the continuous part. Those binary events and the binary inputs trigger switches of the FSM states. The MS combines all binary variables (states, inputs and events) to choose the mode and thus the corresponding continuous dynamic of the SAS. Next, we define each of the four components.
Switched Affine System (SAS). A Switched Affine System is a collection of affine systems

$$x_r(k+1) = A_{i(k)}x_r(k) + B_{i(k)}u_r(k) + f_{i(k)} \tag{3a}$$
$$y_r(k) = C_{i(k)}x_r(k) + D_{i(k)}u_r(k) + g_{i(k)}\,, \tag{3b}$$

where $k \in \mathbb{N}_0$ is the discrete time-instant, $x_r \in \mathcal{X}_r \subseteq \mathbb{R}^{n_r}$ is the real state, $u_r \in \mathcal{U}_r \subseteq \mathbb{R}^{m_r}$ is the real input, $y_r \in \mathcal{Y}_r \subseteq \mathbb{R}^{p_r}$ is the real output, $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathcal{I}}$ is a collection of matrices of appropriate dimensions, and the mode $i \in \mathcal{I} \subset \mathbb{N}$ selects the affine state-update and output function.
Event Generator (EG). An Event Generator generates the binary event signal δ_e according

to the fulfillment of affine constraints or thresholds

$$\delta_e(k) = f_{\mathrm{H}}(x_r(k), u_r(k)), \tag{4}$$

where $f_{\mathrm{H}} : \mathbb{R}^{n_r} \times \mathbb{R}^{m_r} \to \mathcal{D} \subseteq \{0,1\}^{n_e}$ is a vector of descriptive functions of a set of affine constraints.

Finite State Machine (FSM). A Finite State Machine (or automaton) is a discrete dynamic process that evolves according to the binary state-update function

$$x_b(k+1) = f_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)) \tag{5a}$$

$$y_b(k) = g_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)), \tag{5b}$$

where $x_b \in \mathcal{X}_b \subseteq \{0,1\}^{n_b}$ is the binary state, $u_b \in \mathcal{U}_b \subseteq \{0,1\}^{m_b}$ the binary input, $y_b \in \mathcal{Y}_b \subseteq \{0,1\}^{p_b}$ the binary output, δ_e the event and $f_{\mathrm{B}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{X}_b$, $g_{\mathrm{B}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{Y}_b$ are deterministic binary functions.

Mode Selector (MS). The binary state x_b, the binary input u_b and the event δ_e select the mode i of the SAS through a deterministic binary function $f_{\mathrm{M}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{I}$, which is therefore called *Mode Selector*. The output of the function

$$i(k) = f_{\mathrm{M}}(x_b(k), u_b(k), \delta_e(k)) \tag{6}$$

is called *active mode*. We say that a *mode switch* occurs at time-instant k if $i(k-1) \neq i(k)$. Note that one may associate with a state x_b of the FSM more than one mode i according to the event δ_e. Moreover, in the above definitions, $\mathcal{I}$ denotes the set of (feasible) modes, while the infeasible modes are given by the complement of $\mathcal{I}$ in the set of binary state-input-event combinations, i.e. $2^{(n_b+m_b+n_e)} \setminus \mathcal{I}$.

To shorten the notation, we will use the definitions $x \triangleq \left[\begin{smallmatrix} x_r \\ x_b \end{smallmatrix}\right]$, $u \triangleq \left[\begin{smallmatrix} u_r \\ u_b \end{smallmatrix}\right]$, $y \triangleq \left[\begin{smallmatrix} y_r \\ y_b \end{smallmatrix}\right]$, $\mathcal{X} \triangleq \mathcal{X}_r \times \mathcal{X}_b$, $\mathcal{U} \triangleq \mathcal{U}_r \times \mathcal{U}_b$ and $\mathcal{Y} \triangleq \mathcal{Y}_r \times \mathcal{Y}_b$ throughout the rest of the paper.

Definition 3.1: A DHA is *well-posed* on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$, if for all initial conditions $x(0) \in \mathcal{X}$ and for all inputs $u(k) \in \mathcal{U}$ the state trajectory $x(k) \in \mathcal{X}$ and the output trajectory $y(k) \in \mathcal{Y}$ are uniquely defined for all $k \in \mathbb{N}_0$.

3.2 *Piecewise Affine Systems*

Polyhedral piecewise affine (PWA) systems (Sontag 1981, Heemels et al. 2001) are defined by partitioning the state-input space into polyhedra and associating with each polyhedron an affine state-update and output function

$$x(k+1) = A_{j(k)}x(k) + B_{j(k)}u(k) + f_{j(k)} \tag{7a}$$

$$y(k) = C_{j(k)}x(k) + D_{j(k)}u(k) + g_{j(k)} \tag{7b}$$

$$\text{with } j(k) \text{ such that } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{Q}_{j(k)}, \tag{7c}$$

where $k \in \mathbb{N}_0$ is the discrete time-instant, and $x \in \mathcal{X}$ denotes the states, $u \in \mathcal{U}$ the inputs and $y \in \mathcal{Y}$ the outputs, with both real and binary components as defined in the previous section. The polyhedra $\mathcal{Q}_{j(k)}$ define a set of polyhedra $\{\mathcal{Q}_j\}_{j\in\mathcal{J}}$ on the state-input space $\mathcal{X} \times \mathcal{U}$. The real matrices $A_{j(k)}$, $B_{j(k)}$, $C_{j(k)}$, $D_{j(k)}$ and real vectors $f_{j(k)}$, $g_{j(k)}$ with $j(k) \in \mathcal{J}$, $\mathcal{J}$ finite, are constant and of suitable dimensions. We refer to $j(k)$ as the *mode* of the system and to $\#\mathcal{J}$ as the number of modes.

For PWA models, we define well-posedness as in Definition 3.1. The following lemma follows directly from Definition 2.3.

Lemma 3.2: *Let Σ_{PWA} be a PWA system as in (7). If $\{\mathcal{Q}_j\}_{j\in\mathcal{J}}$ is a polyhedral partition of $\mathcal{X}\times\mathcal{U}$, then Σ_{PWA} is well-posed.*

Note however, that the converse statement does not hold in general as a well-posed PWA system may be defined on an overlapping set of polyhedra.

3.3 *Equivalence of DHAs and PWA Systems*

This section states that any well-posed DHA can be transformed into an equivalent PWA representation. The constructive proof serves as a basis for the mode enumeration algorithm proposed in the next section.

Definition 3.3: Let Σ_1, Σ_2 be well-posed hybrid models with states x_1, $x_2 \in \mathcal{X}$, inputs u_1, $u_2 \in \mathcal{U}$ and outputs y_1, $y_2 \in \mathcal{Y}$. The hybrid models Σ_1 and Σ_2 are *equivalent* on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$, if for all initial conditions $x_1(0) = x_2(0) \in \mathcal{X}$ and for all $u_1(k) = u_2(k) \in \mathcal{U}$ the state and output trajectories coincide, i.e. $x_1(k) = x_2(k)$ and $y_1(k) = y_2(k)$ for all discrete time steps $k \in \mathbb{N}_0$.

Lemma 3.4: *(Torrisi and Bemporad 2004, Lemma 1) Let Σ_{PWA} be a well-posed PWA system with states $x \in \mathcal{X}$, inputs $u \in \mathcal{U}$ and outputs $y \in \mathcal{Y}$. Then there exists a well-posed DHA Σ_{DHA} equivalent to Σ_{PWA} on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$.*

Lemma 3.5: *Let Σ_{DHA} be a well-posed DHA with states $x \in \mathcal{X}$, inputs $u \in \mathcal{U}$ and outputs $y \in \mathcal{Y}$. Then there exists a well-posed PWA system Σ_{PWA} equivalent to Σ_{DHA} on $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$.*

Proof Consider the affine thresholds of the Event Generator (4) that define a hyperplane arrangement, which forms by Proposition 2.2 a polyhedral partition. Let $\mathcal{P}_m$ be a polyhedron of this partition. By construction, $\bar{\delta}_e(m) = f_{\mathrm{H}}(x_r, u_r)$ holds for any point $\left[\begin{smallmatrix} x_r \\ u_r \end{smallmatrix}\right] \in \mathcal{P}_m$, namely all points in $\mathcal{P}_m$ trigger the same event $\bar{\delta}_e(m)$. Given a marking m, the associated event $\bar{\delta}_e(m)$, a binary state $\bar{x}_b \in \mathcal{X}_b$ and a binary input $\bar{u}_b \in \mathcal{U}_b$, the Mode Selector determines the mode $\bar{\imath} = f_{\mathrm{M}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_e(m))$ using the binary function (6). The $\bar{\imath}$-th dynamic in the Switched Affine System given by (3) is the corresponding affine dynamic. The Finite State Machine yields the binary state-update as well as the binary output according to (5). Therefore, for each $m \in M$, $\bar{x}_b \in \mathcal{X}_b$ and $\bar{u}_b \in \mathcal{U}_b$, the system

$$x_r(k+1) = A_{\bar{\imath}} x_r(k) + B_{\bar{\imath}} u_r(k) + f_{\bar{\imath}}, \tag{8a}$$

$$x_b(k+1) = f_{\mathrm{B}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_{e(m)}), \tag{8b}$$

$$y_r(k) = C_{\bar{\imath}} x_r(k) + D_{\bar{\imath}} u_r(k) + g_{\bar{\imath}}, \tag{8c}$$

$$y_b(k) = g_{\mathrm{B}}(\bar{x}_b, \bar{u}_b, \bar{\delta}_{e(m)}), \tag{8d}$$

$$\text{if } \begin{bmatrix} x_r(k) \\ u_r(k) \end{bmatrix} \in \mathcal{P}_m, \quad x_b(k) = \bar{x}_b, \quad u_b(k) = \bar{u}_b, \tag{8e}$$

defines a PWA system. In fact, by collecting $x = \left[\begin{smallmatrix} x_r \\ x_b \end{smallmatrix}\right]$, $u = \left[\begin{smallmatrix} u_r \\ u_b \end{smallmatrix}\right]$ and $y = \left[\begin{smallmatrix} y_r \\ y_b \end{smallmatrix}\right]$, by performing the substitutions $A_j = \left[\begin{smallmatrix} A_{\bar{\imath}} & 0 \\ 0 & 0 \end{smallmatrix}\right]$, $B_j = \left[\begin{smallmatrix} B_{\bar{\imath}} & 0 \\ 0 & 0 \end{smallmatrix}\right]$, $f_j = \begin{bmatrix} f_{\bar{\imath}} \\ f_{\mathrm{B}(\cdot)} \end{bmatrix}$ and similarly for C_j, D_j and g_j, and by defining $\mathcal{Q}_j \triangleq \mathcal{P}_m \times \bar{x}_b \times \bar{u}_b \in \mathcal{X} \times \mathcal{U}$, (8a)–(8d) are formally equivalent to (7a)–(7b) and (8e) is formally equivalent to (7c). The well-posedness of the PWA system follows from Proposition 2.2 and Lemma 3.2. □

4 Mode Enumeration Algorithm

Based on the cell enumeration in hyperplane arrangements summarized in Section 2 and the equivalence of DHAs and PWA models shown in Section 3.3, we present in this section an

algorithm that enumerates efficiently the feasible modes of a composition of DHAs and derives an equivalent PWA system.

4.1 *Single DHA*

Consider the DHA Σ as in Section 3.1 and let $\mathcal{X}\times\mathcal{U}$ denote the state-input space of the DHA, for which we want to solve the following problem. Given a binary state $x_b \in \mathcal{X}_b$ and a binary input $u_b \in \mathcal{U}_b$ find the set of feasible modes $\mathcal{J} \subseteq \mathcal{I}$[1], the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}} \in \mathcal{R}$, $\mathcal{R} = \mathcal{X}_r \times \mathcal{U}_r$, and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$, where $\mathcal{S}_j = \{A_j, B_j, f_j, C_j, D_j, g_j\}$. As this is the same problem as in Lemma 3.5, we derive an algorithm from its constructive proof. Note that $\mathcal{I}$ is the image of the Mode Selector and can be computed once the set $M(\mathcal{R})$ has been enumerated.

Algorithm 4.1

function $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ] = \mathsf{SingleDHA}\ (\ \Sigma,\ \mathcal{R},\ x_b,\ u_b\ )$

$j = 0,\ \mathcal{J} = \emptyset$

for $m \in M(\mathcal{R})$

$j = j + 1,\ \mathcal{J} = \mathcal{J} \cup \{j\}$

$\mathcal{P}_j = \mathcal{P}_m = \{z \in \mathcal{R} : \mathrm{SV}(z) = m\}$

get δ_e based on m

$i = f_{\mathrm{M}}(x_b, u_b, \delta_e)$

$\mathcal{S}_j = \left\{ \begin{bmatrix} A_i & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} B_i & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} f_i \\ f_{\mathrm{B}}(x_b,u_b,\delta_e) \end{bmatrix}, \begin{bmatrix} C_i & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} D_i & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} g_i \\ g_{\mathrm{B}}(x_b,u_b,\delta_e) \end{bmatrix} \right\}$

return $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ]$

For a given binary state $x_b \in \mathcal{X}_b$ and binary input $u_b \in \mathcal{U}_b$ Algorithm 4.1 enumerates the feasible modes of the DHA on the real state-input space $\mathcal{R} = \mathcal{X}_r \times \mathcal{U}_r$. With regards to Proposition 2.1, n is equal to the number of hyperplanes generated by the Event Generator, i.e. $n = n_e$, and d is the dimension of the real state-input space $n_r + m_r$. If the binary state-input combination entails no feasible modes, or if the corresponding $\mathcal{X}_r \times \mathcal{U}_r = \emptyset$, the algorithm returns empty sets of polyhedra and dynamics. Repeated calls of Algorithm 4.1 lead to a set of PWA models defined on $\mathcal{X}_r \times \mathcal{U}_r$, where each model is associated with a feasible binary state-input combination. This representation is advantageous if determining the state-update and the outputs for a given state and input is the main purpose, as choosing the respective PWA model can be done by binary search. However, the model can be transformed easily into a PWA model defined on $\mathcal{X} \times \mathcal{U}$ as shown in the proof of Lemma 3.5.

Remark 1 : If the DHA Σ is well-posed, the resulting PWA model is well-posed, too, as shown in Lemma 3.5. Furthermore, by Proposition 2.2, the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ forms a polyhedral partition of $\mathcal{X}_r \times \mathcal{U}_r$.

4.2 *Composition of DHAs*

The algorithm proposed above can be extended in a natural way to deal with composition of DHAs. Consider s DHAs denoted as Σ_i, $i \in \{1, 2, \ldots, s\}$ with states $x_i \in \mathcal{X}_i$, inputs $u_i \in \mathcal{U}_i$ and outputs $y_i \in \mathcal{Y}_i$[2]. Let $\mathcal{I}_i$ be the set of (feasible) modes of the DHA Σ_i. The composition has the exogenous input $u \in \mathcal{U}$ and the exogenous output $y \in \mathcal{Y}$, where *exogenous inputs* are signals coming from outside the composition. Accordingly, *exogenous outputs* are outputs of the composition. We define the real and binary state spaces of the composition $\mathcal{X}_r \triangleq \mathcal{X}_r^1 \times \ldots \times \mathcal{X}_r^s$ and

[1] $\mathcal{J} = \mathcal{I}$ holds, if all modes $\mathcal{I}$ of the Switched Affine System are feasible.

[2] As in the last section x_i, u_i and y_i encompass both real and binary components. Furthermore, $\mathcal{X}_i \triangleq \mathcal{X}_r^i \times \mathcal{X}_b^i$ and accordingly for $\mathcal{U}_i$ and $\mathcal{Y}_i$.

$\mathcal{X}_b \triangleq \mathcal{X}_b^1 \times \ldots \times \mathcal{X}_b^s$, respectively. Accordingly, the compound vectors $x_r \triangleq [(x_r^1)^T, \ldots, (x_r^s)^T]^T \in \mathcal{X}_r$ and $x_b \triangleq [(x_b^1)^T, \ldots, (x_b^s)^T]^T \in \mathcal{X}_b$ are the sorted aggregation of the real and binary states of the s DHAs. Summing up, the compound system has the compound state vector $x = \left[\begin{smallmatrix} x_r \\ x_b \end{smallmatrix}\right] \in \mathcal{X}_r \times \mathcal{X}_b$, the exogenous input u and the exogenous output y.

The connections between the DHAs are equivalent to linear equality constraints between the DHA outputs and inputs. The input of the i-th DHA Σ_i is a linear combination of the DHA outputs and the exogenous input

$$u_i = \sum_{l=1}^{s} L_{li}\, y_l + L_i\, u\,, \tag{9}$$

where L_{li} and L_i are (0,1)-matrices of appropriate dimension and $i, l \in \{1, 2, \ldots, s\}$. Specifically, the L_{li} have dimension $(p_r^l + p_b^l) \times (m_r^i + m_b^i)$. Note that, in general, the inputs and outputs are vectors that include both real and binary components. Since each component of an input (vector) is connected either with an DHA output or an exogenous input, the sum of each row of the matrix $[L_{1i}\ L_{2i} \ldots L_{si}\ L_i]$ is equal to one.

Before describing the algorithm, we recall some definitions and results from graph theory (Deo 1974) to describe the topology of the composition. A *directed graph* or *digraph* $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ consists of a set of *vertices* $\mathscr{V}$, a set of *edges* $\mathscr{E}$ and a mapping that maps every edge onto some ordered pair of vertices. A *directed closed walk* is an alternating sequence of vertices and edges, beginning and ending with the same vertex, such that each edge is oriented from the vertex preceding it to the vertex following it. If additionally, no vertices except the initial and terminal one appear more than once, the directed closed walk is called a *directed circuit*. If a digraph has no directed circuits, it is called *acyclic*, otherwise it is *cyclic*.

The definitions above can be applied directly to the composition of DHAs by defining the DHAs as vertices and the connections from outputs to inputs as directed edges. In general, one edge can represent several connections between two DHAs.

Note that directed circuits in $\mathscr{G}$ correspond to (algebraic) loops in the composition. Conversely, an acyclic directed graph implies the lack of such loops. Besides that, since the DHAs are functions of the compound state $x(k)$ and exogenous input $u(k)$ at time-instant k, the state-update functions providing $x(k+1)$ cannot be part of loops, since the integrating feature of the state-update (difference) equation breaks algebraic loops.

We define the topology of the connections among the DHAs by an adjacency matrix, which can be easily determined based on the connections.

Definition 4.2: Let $\mathscr{G}$ be a digraph with s vertices containing at most one edge per pair of vertices (no parallel edges). Then the adjacency matrix $W = [w_{ij}]$ of the digraph $\mathscr{G}$ is a $s \times s$ $(0,1)$-matrix with $w_{ij} = 1$ if there is an edge directed from the i-th vertex to the j-th vertex and $w_{ij} = 0$ otherwise. The sequence of indices of W is given by $\{1, 2, \ldots, s\}$.

Theorem 4.3: *(Deo 1974, Theorem 9.17) The digraph $\mathscr{G}$ is* acyclic *iff* $\det(I - W) \neq 0$*, where I is the identity matrix.*

Theorem 4.4: *(Deo 1974, Theorem 9.16) If the digraph $\mathscr{G}$ is acyclic, then its vertices can be ordered such that the adjacency matrix of the reordered graph is an upper (or lower) triangular matrix.*

As defined in Geyer et al. (2003), the sequence of indices of the reordered adjacency matrix implies a computational order along which the algorithm will proceed.

Example 4.5

Figure 3 depicts four DHAs ($s = 4$) after reordering the corresponding graph. The computational order is given by $\{1, 2, 3, 4\}$. DHAs Σ_1 and Σ_4 have exogenous inputs, while Σ_3 and Σ_4

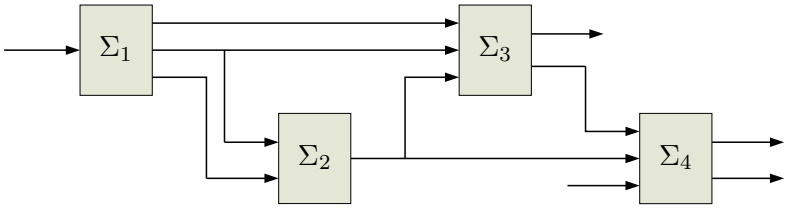


Figure 3. Composition of DHAs of Example 4.5 after reordering

have exogenous outputs. The connection matrices for the input of Σ_3, for example, are given by

$$L_{13} = \begin{bmatrix} 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,0 \end{bmatrix}, \quad L_{23} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad L_{33} = L_{43} = L_3 = \begin{bmatrix} 0\,0 \\ 0\,0 \\ 0\,0 \end{bmatrix}.$$

4.2.1 Compositions Without Loops

In a first step, we assume that the connections do not form loops. This can be easily determined by Theorem 4.3. From Theorem 4.4 follows, that the adjacency matrix can be transformed into an upper triangular matrix employing for example topological sorting (Deo 1974) or matrix permutation. Furthermore, we assume that the indices of the DHAs are such that the adjacency matrix is upper triangular. This implies that the computational order is given by $\{1, 2, \ldots, s\}$. Consequently, Σ_i depends only on exogenous inputs and on outputs of Σ_j, $j < i$ with $i, j \in \{1, \ldots, s\}$.

Similar to the single DHA case, our aim is the following. For the composition of DHAs $\{\Sigma_i\}_{i=1,\ldots,s}$ defined on $\mathcal{X} \times \mathcal{U}$, the *compound* state *exogenous* input space, and for a given binary compound state $x_b \in \mathcal{X}_b$ and a given exogenous binary input $u_b \in \mathcal{U}_b$ determine the set of (feasible) modes $\mathcal{J} \subseteq \mathcal{I}_1 \times \ldots \times \mathcal{I}_s$, the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}} \in \mathcal{X}_r \times \mathcal{U}_r$ and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$. The hereafter described Algorithm 4.6 partitions the real compound state exogenous input space $\mathcal{X}_r \times \mathcal{U}_r$ sequentially (for a given combination of x_b and u_b).

Consider the first DHA Σ_l, $l = 1$. As mentioned before, its input vector is a subset of the exogenous input vector (and independent from other DHAs). Therefore, since the sets of real states and inputs are available together with the binary states and inputs, Algorithm 4.1 can be used to determine the modes $\mathcal{J}_l$, the polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}_l}$ and the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}_l}$ of Σ_l. Clearly, the polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}_l}$ partition the $\mathcal{X}_r \times \mathcal{U}_r$ space.

For each mode $j \in \mathcal{J}_l$, the tuple $\mathcal{S}_j$ defines the output vector of Σ_l as an affine function of the compound state vector x and the exogenous input vector u. According to the connections in the compound model (9) these output functions define the inputs of DHAs with higher computational order, namely Σ_i, $i \in \{l+1, \ldots, s\}$. Thus before proceeding, the algorithm replaces their inputs by the affine output function of Σ_l using the function Subst(). More specifically, for mode $j \in \mathcal{J}_l$, the function Subst() cycles through all DHAs $\{\Sigma_i\}_{i=l+1,\ldots,s}$. If the adjacency matrix indicates a connection from an output of Σ_l to an input of Σ_i, $i \in \{l+1, \ldots, s\}$, i.e. $w_{li} = 1$, the input u_i is replaced by $u_{i,j} = L_{li}\, y_{l,j} + L_i\, u$, where $y_{l,j} = C_j\, x + D_j\, u + g_j$. Note that C_j, D_j, g_j are elements of $\mathcal{S}_j$ of the DHA Σ_l. The index j underlines the fact that the input equation is specific to the mode $j \in \mathcal{J}_l$. The substitution operation assures that Σ_{l+1} solely depends on compound states and exogenous inputs.

Next, for a given mode $j \in \mathcal{J}_l$, l is increased by one and the algorithm is called again to partition the polyhedron $\mathcal{P}_j$ into a set of polyhedra using the hyperplanes of the DHAs with computational order greater than l. This is repeated for all the remaining $j \in \mathcal{J}_l$. If l reaches its maximum s, the current branch terminates and the set of polyhedra of Σ_s, which are part of the overall set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ of the compound DHA system, are added to it. Stepping sequentially through the composition of DHAs according to their computational order leads to the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the corresponding PWA dynamic $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$.

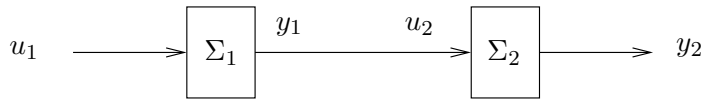


Figure 4. Composition of DHAs of Example 4.8

Remark 2: During the execution of the algorithm, the set of polyhedra is always defined on the complete state-input space — or more precisely, on the real compound state and exogenous input space — $\mathcal{X}_r \times \mathcal{U}_r$. In particular, the dimensionality of the problem (i.e. the number of inputs, states and outputs) is always the same. As the algorithm proceeds, additional hyperplanes are added cutting the existing polyhedra into smaller ones.

Recapitulating the above and given the binary compound state vector $x_b \in \mathcal{X}_b$ and the binary exogenous binary input vector $u_b \in \mathcal{U}_b$, Algorithm 4.6 is summarized as follows. The matrices W, L_{li} and L_i are omitted in the argument to simplify the exposition.

Algorithm 4.6

reorder $\{\Sigma_i\}_{i=1,\dots,s}$ such that W is upper triangular
$[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ] = \mathsf{CompDHA}\ (\ \{\Sigma_i\}_{i=1,\dots,s},\ \mathcal{X}_r \times \mathcal{U}_r,\ x_b,\ u_b,\ 1\ )$

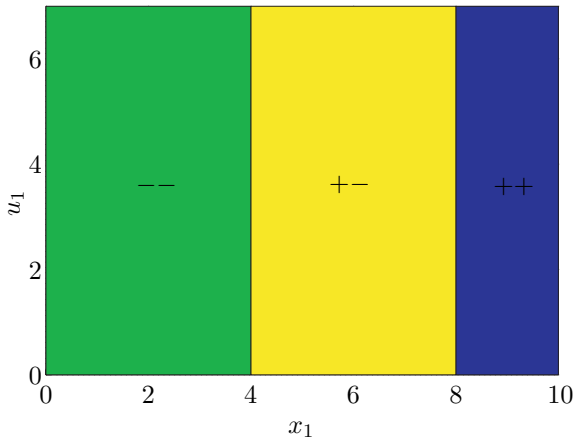
function $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ] = \mathsf{CompDHA}\ (\ \{\Sigma_i\}_{i=l,\dots,s},\ \mathcal{R},\ x_b,\ u_b,\ l\ )$
  $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_l},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_l}\ ] = \mathsf{SingleDHA}\ (\ \Sigma_l,\ \mathcal{R},\ x_b,\ u_b\ )$
  if $l \leq s$ **then**
    $\mathcal{P} = \emptyset,\ \ \mathcal{S} = \emptyset,\ \ \mathcal{J} = \emptyset$
    for $j \in \mathcal{J}_l$
      $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_{\text{new}}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_{\text{new}}}\ ] = \mathsf{CompDHA}\ (\ \mathsf{Subst}\ (\ \{\Sigma_i\}_{i=l+1,\dots,s},\ \mathcal{S}_j,\ l\ ),\ \mathcal{P}_j,\ x_b,\ u_b,\ l+1\ )$
      $\{\mathcal{P}_j\}_{j\in\mathcal{J}} = \{\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_{\text{new}}}\ \}$
      $\{\mathcal{S}_j\}_{j\in\mathcal{J}} = \{\ \{\mathcal{S}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_{\text{new}}}\ \}$
      $\mathcal{J} = \mathcal{J} \cup \mathcal{J}_{\text{new}}$
    return $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}}\ ]$
  else
    return $[\ \{\mathcal{P}_j\}_{j\in\mathcal{J}_l},\ \{\mathcal{S}_j\}_{j\in\mathcal{J}_l}\ ]$

function $\{\Sigma_i\}_{i=l+1,\dots,s} = \mathsf{Subst}\ (\ \{\Sigma_i\}_{i=l+1,\dots,s},\ \mathcal{S}_j,\ l\ )$
  for $i \in \{l+1,\dots,s\}$
    if $w_{li} = 1$ **then**
      substitute u_i in Σ_i by $L_{li}\,(C_j x + D_j u + g_j) + L_i\,u$
  return $\{\Sigma_i\}_{i=l+1,\dots,s}$

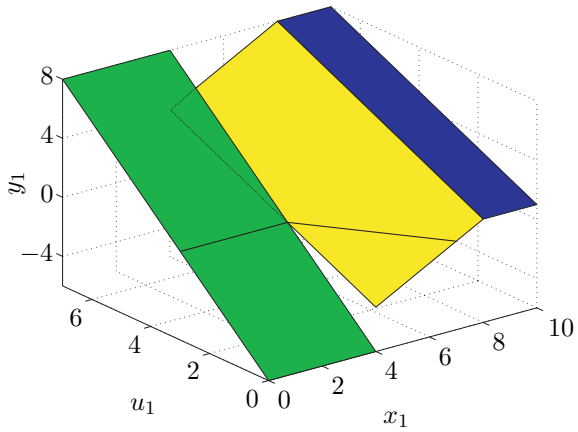
As for the single DHA case, the algorithm yields a PWA model defined on $\mathcal{X}_r \times \mathcal{U}_r$ for every feasible combination of $x_b \in \mathcal{X}_b$ and $u_b \in \mathcal{U}_b$. The following corollary extends Remark **1**. It follows in a constructive way from Algorithm 4.6, Proposition 2.2 and Lemma 3.5.

Corollary 4.7: *Given a composition of DHAs $\{\Sigma_i\}_{i=1,\dots,s}$ without loops, where each Σ_i is well-posed, the resulting PWA model is well-posed, too, and its set of polyhedra forms a polyhedral partition.*

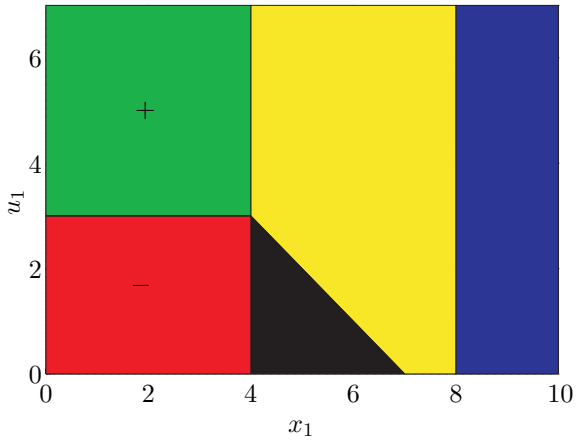
Example 4.8
Consider now the composition of DHAs shown in Fig. 4 with the state $x_1 \in \mathcal{X} = [0, 10]$, the



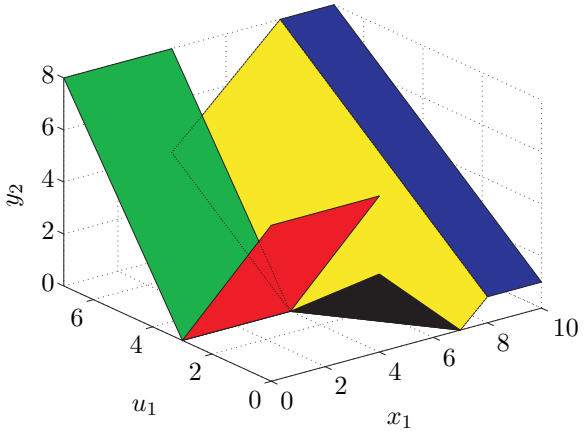
(a) Polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}_1}$



(b) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}_1}$. Bold lines indicate the intersections of the output functions with $y_1 = 0$



(c) Polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}_2}$



(d) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}_2}$

Figure 5. Polyhedral partitions and PWA output functions of the composition of DHAs of Example 4.8

exogenous input $u_1 \in \mathcal{U} = [0, 7]$, the exogenous output $y_2 \in \mathbb{R}$ and the connection $u_2 = y_1$. The DHA Σ_1 is given by[1]

$$\mathrm{EG}_1 : \begin{cases} \delta_1 = [x_1 \geq 4], \\ \delta_2 = [x_1 \geq 8] \end{cases}$$

$$\mathrm{SAS}_1 : y_1 = \begin{cases} 2u_1 - 6 & \text{if } i_1 = 1, \\ x_1 + u_1 - 7 & \text{if } i_1 = 2, \\ u_1 + 1 & \text{if } i_1 = 3 \end{cases}$$

$$\mathrm{MS}_1 : i_1 = \begin{cases} 1 & \text{if } \bar{\delta}_1 \wedge \bar{\delta}_2, \\ 2 & \text{if } \delta_1 \wedge \bar{\delta}_2, \\ 3 & \text{if } \delta_1 \wedge \delta_2 \end{cases}$$

and Σ_2 yields as output the 1-norm of its input which amounts to

$$\mathrm{EG}_2 : \delta_3 = [u_2 \geq 0] \qquad \mathrm{SAS}_2 : y_2 = \begin{cases} -u_2 & \text{if } i_2 = 1, \\ u_2 & \text{if } i_2 = 2 \end{cases} \qquad \mathrm{MS}_2 : i_2 = \begin{cases} 1 & \text{if } \bar{\delta}_3, \\ 2 & \text{if } \delta_3 \end{cases}$$

Clearly, the state-input space is given by $\mathcal{X} \times \mathcal{U}$, the corresponding digraph is acyclic and the indices of the DHAs are already ordered such that the adjacency matrix is upper triangular.

[1]Since the state-update functions do not influence the polyhedral partition they are omitted here for brevity.

In a first step, the algorithm determines the hyperplane arrangement of Σ_1, which contains the hyperplanes of the EG_1, namely $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 = 4\}$ and $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 = 8\}$. The enumeration of the cells in the arrangement leads to the markings and the polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}_1}$, $\mathcal{J}_1 = \{1, 2, 3\}$, as shown in Fig. 5(a). The corresponding PWA output functions are depicted in Fig. 5(b).

In a second step, $\{\mathcal{P}_j\}_{j\in\mathcal{J}_1}$ is further partitioned by the hyperplane defined in the EG_2. Starting with mode $j = 1 \in \mathcal{J}_1$ this is done in the following way. The function Subst() replaces the expressions for u_2 in the EG_2 and in the SAS_2 by $2u_1 - 6$. Thus, for this particular mode, the hyperplane arrangement of Σ_2 is defined within the polyhedron $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, 0 \le x_1 \le 4\}$ and holds the single hyperplane $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, u_1 = 3\}$. The corresponding markings are shown in Fig. 5(c). As both modes of the EG_2 are feasible, $\mathcal{J}_2$ contains two modes and the polyhedron $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, 0 \le x_1 \le 4\}$ is partitioned into two. Accordingly, the mode $j = 2 \in \mathcal{J}_1$ leads to the hyperplane arrangement $\{[\begin{smallmatrix} x_1 \\ u_1 \end{smallmatrix}] \in \mathcal{X} \times \mathcal{U} \,|\, x_1 + u_1 = 7\}$ and also to two modes, whereas the hyperplane arrangement corresponding to $j = 3 \in \mathcal{J}_1$ is empty and thus no additional mode is added.

The final polyhedral partition $\{\mathcal{P}_j\}_{j\in\mathcal{J}} = \{\mathcal{P}_j\}_{j\in\mathcal{J}_2}$ and the PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}} = \{\mathcal{S}_j\}_{j\in\mathcal{J}_2}$ are shown in Fig. 5(c) and 5(d), respectively.

4.2.2 Compositions With Loops

The algorithm is now generalized to compositions of DHAs containing algebraic loops. Having determined the adjacency matrix W and verified that the digraph is cyclic, one has to identify the connections whose removal breaks all loops and renders the corresponding digraph acyclic. These connections correspond to the feedback arc set.

Definition 4.9: Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be a digraph. A set $\mathscr{F} \subseteq \mathscr{E}$ is a *feedback arc set* (FAS) for $\mathscr{G}$, if $\mathscr{G}' = (\mathscr{V}, \mathscr{E} - \mathscr{F})$ is acyclic. The set $\mathscr{F}$ is a *minimum* FAS if the number of edges in $\mathscr{F}$ is minimum.

Finding the minimum FAS is $\mathcal{NP}$-hard. However, our algorithm does not require the FAS to be minimal. For a given digraph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ fast and effective heuristics exist (Eades, Lin, and Smyth 1993) with time complexity $O(\#\mathscr{E})$ that yield an FAS $\mathscr{F}$ with upper bounded cardinality $\#\mathscr{F} \le \#\mathscr{E}/2 - \#\mathscr{V}/6$.

Removing the loops in the composition of DHAs is equivalent to replacing the connections corresponding to feedback arcs by newly created auxiliary inputs. In general, a feedback arc f corresponds to more than one connection between two DHAs and encompasses therefore real as well as binary variables. Thus, we add a real auxiliary input for every connection from a real output to a real input and accordingly a binary auxiliary input for every connection corresponding to binary variables. We denote the vector of auxiliary real and binary inputs by $v_r \in \mathcal{V}_r$ and $v_b \in \mathcal{V}_b$, respectively, and define $v \triangleq [\begin{smallmatrix} v_r \\ v_b \end{smallmatrix}]$. The removed connections are kept as equality constraints in

$$v_i = \sum_{l=1}^{s} \Lambda_{li}\, y_l \,, \tag{10}$$

where Λ_{li} are (0,1)-matrices of appropriate dimension and $i, l \in \{1, 2, \ldots, s\}$. Specifically, to remove a connection, the corresponding entries in the connection matrices L_{li} in (9) are set to zero, while the corresponding entry in Λ_{li} is set to one. Repeating this for all $f \in \mathscr{F}$ yields a composition of DHAs without loops defined on the augmented exogenous input space $\mathcal{U}_r \times \mathcal{V}_r \times \mathcal{U}_b \times \mathcal{V}_b$.

Assume again that the indices of the DHAs are ordered such that the adjacency matrix is upper triangular. Given a binary state $x_b \in \mathcal{X}_b$ and an augmented binary input $[\begin{smallmatrix} u_b \\ v_b \end{smallmatrix}] \in \mathcal{U}_b \times \mathcal{V}_b$, this assumption allows us to use Algorithm 4.6 to derive the set of feasible modes $\mathcal{J}'$, the set of polyhedra $\{\mathcal{P}'_j\}_{j\in\mathcal{J}'}$ and the corresponding PWA dynamics $\{\mathcal{S}'_j\}_{j\in\mathcal{J}'}$ defined on the augmented

real state-input space $\mathcal{X}_r \times \mathcal{U}_r \times \mathcal{V}_r$. As the algorithm proceeds, the outputs of the DHAs are replaced step by step by affine combinations of states and exogenous inputs. Therefore, the constraints (10) storing the removed connections must be updated simultaneously by replacing outputs by affine combinations of states and exogenous inputs according to the respective PWA output function. This yields the set of constraints $\{\mathcal{C}'_j\}_{j\in\mathcal{J}'}$, where $\mathcal{C}'_j$ denotes the updated constraints corresponding to the mode $j \in \mathcal{J}'$. Specifically, $\mathcal{C}'_j$ is of the form

$$[H^j_{x_r}\ H^j_{x_b}]\begin{bmatrix}x_r\\x_b\end{bmatrix} + [H^j_{u_r}\ H^j_{u_b}]\begin{bmatrix}u_r\\u_b\end{bmatrix} + [H^j_{v_r}\ H^j_{v_b}]\begin{bmatrix}v_r\\v_b\end{bmatrix} = K^j\,, \tag{11}$$

where $H^j_{x_r}$, $H^j_{x_b}$, $H^j_{u_r}$, $H^j_{u_b}$, $H^j_{v_r}$ and $H^j_{v_b}$ are matrices with $\#\mathscr{F}$ rows and an appropriate number of columns, and K^j is a column vector with $\#\mathscr{F}$ components.

In a last step, the algorithm cycles through all modes $j \in \mathcal{J}'$ and imposes the updated constraints $\mathcal{C}'_j$ to remove the auxiliary inputs. This yields the set of modes $\mathcal{J}$ of the original composition containing loops and the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ with the corresponding PWA dynamics $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$ defined on the original space $\mathcal{X}_r \times \mathcal{U}_r$. In general, some of the modes in the set $\mathcal{J}'$ will prove to be infeasible and thus $\mathcal{J} \subseteq \mathcal{J}'$.

Summing up, after identifying and removing the algebraic loops, the above algorithm translates the mode enumeration problem into a higher-dimensional space, while the constraints (10) preserve the information about the loops. In a last step, the algorithm imposes these constraints and projects the problem back into the original state-input space. Note that a non-minimal FAS does not increase the number of modes. This is due to the fact that a non-minimal FAS leads to the removal of some surplus connections and thus to an auxiliary space of higher dimension than strictly necessary. When imposing the constraints and projecting the problem down onto the original space, the same number of modes results regardless of whether the FAS was minimal or not. However, the higher dimensionality of the auxiliary space tends to lead to a slight increase of the overall computation time.

Consider now the mode $j \in \mathcal{J}'$ with the associated polyhedron $\mathcal{P}'_j$ and the constraint $\mathcal{C}'_j$. The following three cases may occur when imposing the constraint.

(i) If $\det(H^j_{v_r}) \neq 0$, one can express the auxiliary real input as a function of the real state and the real exogenous input, and we substitute v_r in $\mathcal{P}'_j$. This is the same as intersecting $\mathcal{P}'_j$ with the (hyperplane defined by the) constraint $\mathcal{C}'_j$ and projecting the result on the original state-input space $\mathcal{X}_r \times \mathcal{U}_r$. The associated PWA dynamic $\mathcal{S}_j$ is derived by substituting v_r in $\mathcal{S}'_j$. If the polyhedron $\mathcal{P}_j$ is non-empty, the auxiliary input has been removed successfully and $\mathcal{P}_j$ is now solely defined on $\mathcal{X}_r \times \mathcal{U}_r$. Therefore, we add the mode j to $\mathcal{J}$, the polyhedron $\mathcal{P}_j$ to $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ and the PWA dynamic $\mathcal{S}_j$ to $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$. If $\mathcal{P}_j$ is empty, the corresponding mode j is infeasible and thus discarded.

(ii) However, if $\det(H^j_{v_r}) = 0$, there exists either no or an infinite number of solutions for v_r. This is the case when the intersection of $\mathcal{P}'_j$ with the (hyperplane defined by the) constraint $\mathcal{C}'_j$ is lower-dimensional or empty. In both cases, the corresponding mode and polyhedron are infeasible and thus removed.

(iii) In the presence of auxiliary binary inputs v_b, the algorithm is called for all combinations of binary states, exogenous binary inputs and auxiliary binary inputs. In general, based on the constraints (10), which include binary variables, some of these combinations will prove to be infeasible and the algorithm needs to discard the associated modes and polyhedra.

The first two cases, in which loops have either zero, one or an infinite number of solutions are well-known from linear systems theory with the only difference, that in our case, $\det(H^j_{v_r})$ is a local property that holds only for a given polyhedron and not for the whole state-input space. However, when dealing with loops in hybrid systems, additional difficulties may arise. As the next example will show, even if for all modes $\det(H^j_{v_r}) \neq 0$ holds, the resulting polyhedra do not necessarily form a polyhedral partition of $\mathcal{X}_r \times \mathcal{U}_r$ and the composition of DHAs is thus not

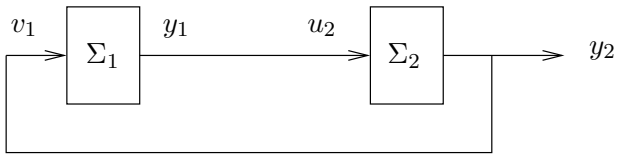
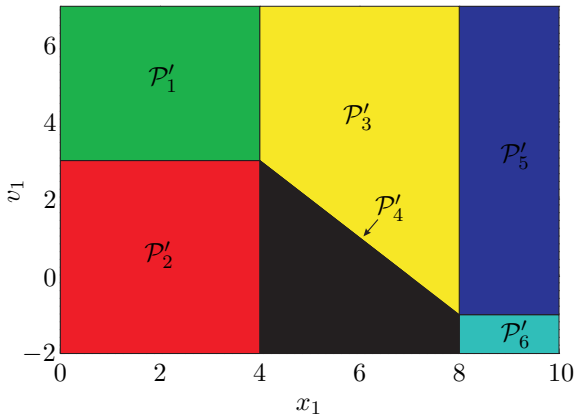
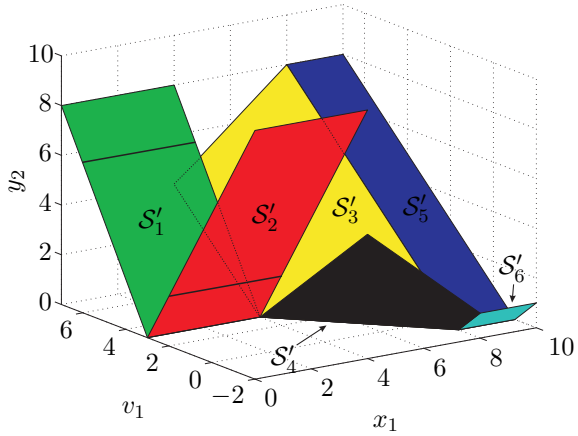


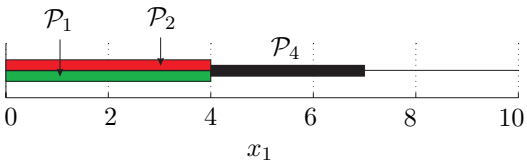
Figure 6. Composition of DHAs of Example 4.10 containing a feedback loop



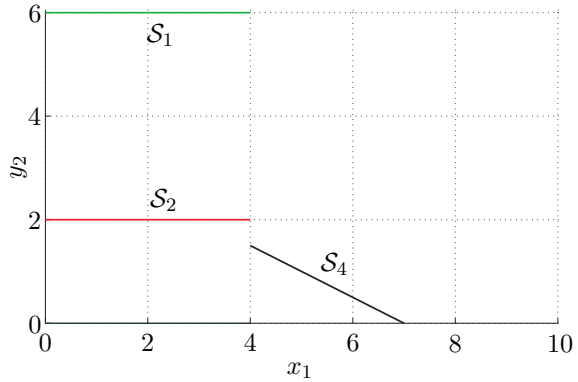
(a) Set of polyhedra $\{\mathcal{P}'_j\}_{j\in\mathcal{J}'}$ defined on the augmented state-input space $\mathcal{X}\times\mathcal{V}$



(b) PWA output functions given in $\{\mathcal{S}'_j\}_{j\in\mathcal{J}'}$ and defined on the augmented state-input space $\mathcal{X}\times\mathcal{V}$. Bold lines indicate the intersections of the output functions with $y_2 = v_1$



(c) Set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ defined on the original state-space $\mathcal{X}$



(d) PWA output functions given in $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$ and defined on the original state-space $\mathcal{X}$

Figure 7. Set of polyhedra and PWA output functions of the composition of DHAs in Example 4.10

well-posed in general.

Example 4.10 Reconsider the composition of DHAs in Example 4.8 to which we add a feedback loop as shown in Fig. 6. This removes the exogenous input u_1 and reduces the state-input space to $\mathcal{X} = [0, 10]$. The corresponding digraph is cyclic and contains the feedback arc $f = (\Sigma_2, \Sigma_1)$. Introducing the auxiliary real variable $v_1 \in \mathcal{V} = \mathbb{R}$ and storing the removed connection $v_1 = y_2$ in the form of $\Lambda_{11} = 0$, $\Lambda_{21} = 1$ allows one to break the loop. As a result, the composition is defined on the augmented state-input space $\mathcal{X}\times\mathcal{V}$, and Algorithm 4.6 leads to the set of polyhedra $\{\mathcal{P}'_j\}_{j\in\mathcal{J}'}$ and the PWA dynamics $\{\mathcal{S}'_j\}_{j\in\mathcal{J}'}$ shown in Fig. 7(a) and Fig. 7(b), respectively[1].

Next we impose the constraint $v_1 = y_2$. Consider the mode $j = 1 \in \mathcal{J}'$ with the polyhedron $\mathcal{P}'_1 = \{\left[\begin{smallmatrix} x_1 \\ v_1 \end{smallmatrix}\right] \in \mathcal{X}\times\mathcal{V} \mid \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right] \left[\begin{smallmatrix} x_1 \\ v_1 \end{smallmatrix}\right] \leq \left[\begin{smallmatrix} 4 \\ -3 \end{smallmatrix}\right]\}$, the output function $y_2 = 2v_1 - 6$ and the updated constraint $v_1 = 6$. The fact, that $\det(H^1_{v_r})$ is different from zero allows us to derive $\mathcal{P}_1 = \{x_1 \in$

[1]Note that the v_1-axis has been artificially restricted to $-2 \leq v_1 \leq 7$ to facilitate the plotting.

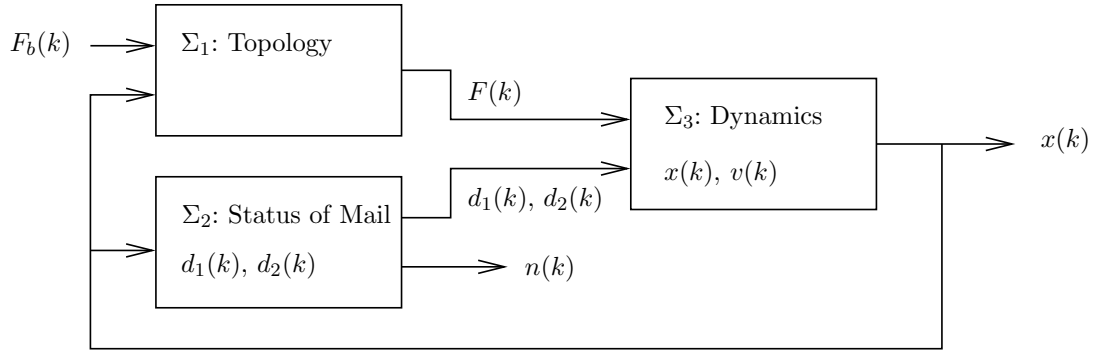


Figure 8. Paperboy example consisting of three DHAs with the respective states

$\mathcal{X} \,|\, x_1 \leq 4\}$. As $\mathcal{P}_1$ is non-empty, the mode $j = 1$ is feasible. The corresponding output function is given by $y_2 = 6$. The modes $j = 2$ and $j = 4$ are handled in a similar way. For the modes $j = 3$ and $j = 5$, the updated constraints are $x_1 = 7$ and $0 = 1$, respectively. In both cases, $\det(H^j_{v_r})$ is zero. For $j = 3$, this leads to an infinite number of solutions and a lower dimensional polyhedron containing only the single point $x_1 = 7$, whereas for $j = 5$ no solution exists. This can be also seen from Fig. 7(b), where the intersection of the hyperplane $v_1 = y_2$ (not depicted) with $\mathcal{S}_3$ projected on the $\mathcal{X}$ space leads to $x_1 = 7$. On the other hand, the hyperplanes $v_1 = y_2$ and $\mathcal{S}_5$ are parallel. Both modes together with the associated polyhedra and dynamics are therefore infeasible and thus removed.

The resulting set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$, $\mathcal{J} = \{1, 2, 4\}$, and the PWA output functions $\{\mathcal{S}_j\}_{j\in\mathcal{J}}$, which are defined on the one-dimensional state-space $\mathcal{X}$, are shown in Fig. 7(c) and 7(d), respectively. Observe that $\mathcal{P}_1 = \mathcal{P}_2$ and that two different output functions are associated to them. Because of that and as a part of the state-space, namely $\{x_1 \in \mathcal{X} \,|\, x_1 \geq 7\}$, is not covered, $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ does not form a polyhedral partition of $\mathcal{X} = [0, 10]$. Therefore, the resulting PWA system and consequently also the corresponding composition of DHAs is not well-posed.

This example demonstrates how tight interactions render modes infeasible. When compared to Example 4.8, the additional algebraic loop ("tight interaction") in Example 4.10 renders three out of the six modes infeasible. Moreover, this example shows that in the presence of loops the polyhedra of the resulting PWA system do not necessarily form a polyhedral partition. The reason for this is twofold. First, modes might be infeasible either because the intersection of the associated polyhedron with the updated constraint is empty or because $\det(H^j_{v_r})$ of the constraint is zero. In general, infeasible modes result in gaps in the state-input space. Second, polyhedra (associated with different dynamics) may overlap.

As a result, using well-posed DHAs to form a composition of DHAs with loops does not guaranty well-posedness of the overall composition. However, as well-posedness of the composition of DHAs relates directly to well-posedness of the corresponding PWA model, we can conclude that the composition of DHAs is well-posed if and only if the corresponding PWA system is well-posed. Based on the polyhedra and the dynamics, one can easily evaluate well-posedness of the PWA model. If the set of polyhedra $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ of the PWA model forms a polyhedral partition, well-posedness is assured by Lemma 3.2. On the other hand, if the union of $\{\mathcal{P}_j\}_{j\in\mathcal{J}}$ covers the state-input space completely, and if all pairs of overlapping polyhedra are associated to the same PWA dynamic, the PWA model is well-posed, too.

5 Examples and Applications

This final section presents two examples showing how the mode enumeration algorithm can be used to efficiently derive the PWA representation of a given hybrid system. Moreover, it is demonstrated how the knowledge about the set of (feasible) modes can be exploited to reduce

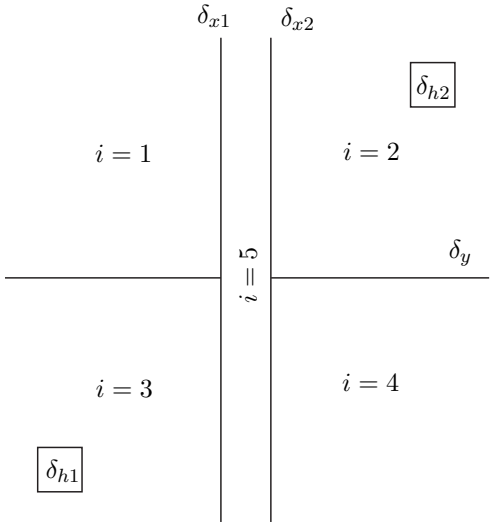


Figure 9. Topology of the neighborhood in the paperboy example as defined by DHA Σ_1. The four properties are associated with the modes $i = 1, ..., 4$. The road has the mode $i = 5$. The road and the properties are separated by three thresholds, while the two houses are defined by two thresholds

the computation time of Model Predictive Control.

5.1 *Car Example*

In Torrisi and Bemporad (2001), the authors proposed a hybrid model of a car with a robotized gear shift. This example was adopted in Bemporad (2002), where the author computes the MLD model using HYSDEL and the PWA model equivalent to the MLD model using multi-parametric and mixed integer linear programming. As the model is given in HYSDEL, the algorithm in Section 4 starts from this description to translate the car example into a PWA model. The resulting PWA model encompasses 30 polyhedra and six different modes. Using MATLAB 5.3 on a Pentium III 650 MHz machine, the model is computed in 1.9 s. This is 40 times faster than the algorithm reported in Bemporad (2002) on a similar machine and 50 faster than the conversion approach of Villa et al. (2004).

The reason for the shorter computation time of our algorithm is twofold. First, the algorithm presented here exploits the structure of the composition of DHAs, while the algorithms presented in Bemporad (2002) and Villa et al. (2004) deal with MLD models concealing that structural information. Second, the approach in Bemporad (2002) needs to remove redundant inequalities at each iteration of the exploration algorithm. This operation may dominate the total computation time in Bemporad (2002).

Apart from this, both Bemporad (2002) and Villa et al. (2004) lack the compositional capability and can only handle the transformation from a single MLD to a single PWA model. In particular, models with loops cannot be tackled.

5.2 *Paperboy Example*

A paperboy delivers by bike two heavy and bulky mail items to two different houses within a neighborhood consisting of four properties and one road. The properties and the road have different slopes and different friction coefficients.

The (exogenous) system input at time-instant k is given by the force $F_b(k) \in \mathcal{U} \subset \mathbb{R}^2$, $\mathcal{U} = [-F_{\max}, F_{\max}]^2$, $F_{\max} = 162\,\mathrm{N}$ that the paperboy applies to his bike in order to accelerate and brake. Driven by F_b, the paperboy cycles in the two-dimensional neighborhood $\mathcal{X}_1 = [-s_n, s_n]^2$ with $s_n = 1000\,\mathrm{m}$. His position is given by $x(k) = [x_1(k), x_2(k)]^T \in \mathcal{X}_1 \subset \mathbb{R}^2$ and his speed $v(k) \in \mathcal{X}_2 \subset \mathbb{R}^2$ is limited by $\mathcal{X}_2 = [-v_{\max}, v_{\max}]^2$, where $v_{\max} = 15\,\mathrm{m/s}$. Two binary states

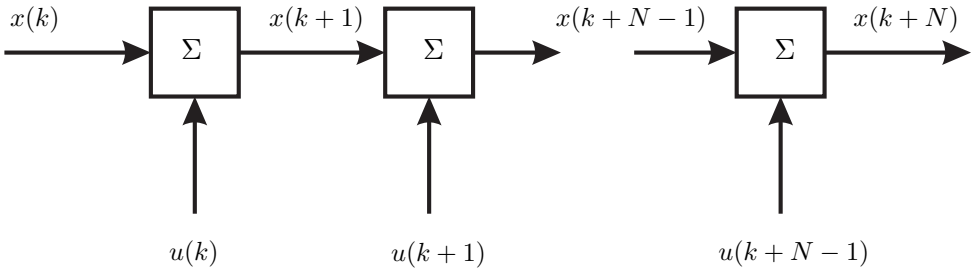


Figure 10. Conceptual scheme of the N-step prediction model

$d_1(k)$, $d_2(k) \in \{0, 1\}$ denote the status of the mail delivery. The (exogenous) outputs of the model are the position $x(k)$ and the number of delivered mail items $n(k) \in \{0, 1, 2\}$.

As depicted in Fig. 8, the paperboy problem can be decomposed into three DHAs, which are described in detail in Geyer et al. (2003). The corresponding HYSDEL code can be found in Geyer (2005, Appendix A.1). As shown in Geyer et al. (2003), the paperboy example is defined on an eight-dimensional state-input space. The composition contains no algebraic loops, since the third DHA's state-update function for $x(k)$ introduces a delay of one sampling interval that breaks the loop. Algorithm 4.6 yields the equivalent PWA model within 6.5 s on a 2.8 GHz Pentium IV PC. It encompasses 168 feasible modes and polyhedra, what is by far below the upper bound of 7099 given by Buck's formula (2).

The paperboy starts the mail delivery at a random position $x(0)$ with speed $v(0) = 0$. His objective is to first deliver one mail item to House 1 centered around x_{h1} and then to move on to House 2 at position x_{h2} to deliver the second mail item. Using the force that the paperboy applies as manipulated variable, namely $u(k) = F_b(k)$, this control objective can be expressed by the cost function

$$J(x(k), v(k), U(k)) = \sum_{\ell=0}^{N-1} \|x(k+\ell|k) - x_{ref}(k+\ell|k)\|_1 + \epsilon\|u(k+\ell|k)\|_1 \,, \tag{12}$$

which penalizes the predicted deviation of the position from its reference over the horizon N using the 1-norm for the sequence of manipulated variables $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T$. The reference x_{ref} is switched from x_{h1} to x_{h2} when the paperboy reaches House 1. Additionally, the very small penalty term $\epsilon = 10^{-6}$ is imposed on the manipulated variable.

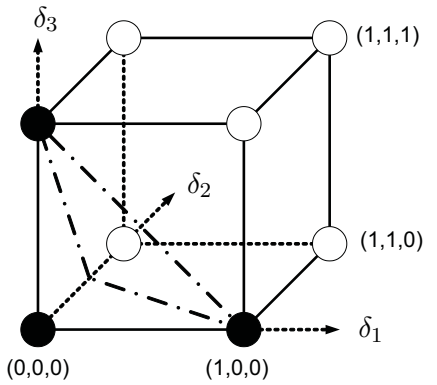
In the next section, we will use the paperboy example to evaluate the potential of the mode enumeration algorithm to reduce the computation time of Model Predictive Control (MPC), see e.g. Maciejowski (2002). The MPC control problem amounts to minimizing the cost function (12) subject to the evolution of the paperboy model over the prediction horizon and subject to constraints on F_b, x and v as given above. The solution of this optimization problem, which is a *Mixed-Integer Linear Program* (MILP), yields the control input, namely the force F_b.

5.3 *Cuts for Model Predictive Control*

MPC of discrete-time linear hybrid systems uses an internal hybrid model, which is usually in Mixed Logical Dynamical (MLD) form (Bemporad and Morari 1999). In the great majority of cases, the MLD model is derived starting from a composition of DHAs described textually in HYSDEL. When translating a composition of DHAs into an MLD model, information about the structure of the hybrid model is lost. However, the explicit computation of the set of feasible modes of the composition of DHAs allows one to add this structural information to the MLD model in the form of cuts. These cuts, which are formulated as integer inequality constraints on the binary inputs, binary states and binary variables in the MLD model, prune infeasible combinations of these binary variables, or equivalently modes, from the MLD model by restricting

δ_1	δ_2	δ_3	i	feasible
0	0	1	1	yes
0	0	0	2	yes
1	0	1	3	no
1	0	0	4	yes
1	1	1	5	no
1	1	0	6	no
0	1	1	7	no
0	1	0	8	no

(a) Truth table for the binary variables



(b) Representation of the binary variables in $\mathbb{R}^3$

Figure 11. Revisited Example 4.10 with the truth table for the example's binary variables indicating whether a mode is feasible or not along with the mode number i. Furthermore, the rows of the truth table are shown in $\mathbb{R}^3$, where the black (white) nodes refer to feasible (infeasible) modes. The intersections of the unit cube with the hyperplane defined by the cut are indicated by dash dotted lines

the solver to only consider combinations of binary variables (modes) that are feasible. Infeasible modes are thus removed explicitly from the solver's search tree.

Example 5.1

Consider again Example 4.10, whose binary variables are the three event variables δ_1, δ_2 and δ_3. Thus, eight possible modes exist, out of which three are feasible. As shown in Table 11(a), a truth table can be easily built that indicates whether a given combination of binary variables corresponds to a feasible mode or not. Figure 11(b) provides a pictorial description of the truth table's rows in $\mathbb{R}^3$, where the feasible (infeasible) modes are indicate by black (white) nodes. A cut removing the infeasible modes is for example $\delta_1 + 2\delta_2 + \delta_3 \leq 1$. To visualize this cut, the binary variables δ are relaxed to the real variables ρ. This leads to the cutting hyperplane $\rho_1 + 2\rho_2 + \rho_3 = 1$, whose intersection with the unit cube is indicated in Fig. 11(b) by dash dotted lines.

Given the current state and the input, the internal MLD model computes the state at the next time-instant and the output. One might refer to such a model as a single-step prediction model defined on the state-input space $\mathcal{X} \times \mathcal{U}$. When building the optimization problem for MPC with the horizon N, this model is repeated N times. More specifically, the series connection of N identical single-step models is built as shown in Fig. 10, where each model uses the state predicted by the previous model as initial state. We might consider these N models as one single model defined on the state-input space $\mathcal{X} \times \mathcal{U}^N$. Given the initial state $x(k) \in \mathcal{X}$ and the sequence of inputs $U(k) = [(u(k))^T, \ldots, (u(k+N-1))^T]^T \in \mathcal{U}^N$, this models provides the state evolution over N time steps. We thus refer to it as the N-step prediction model. The mode enumeration technique allows us to introduce cuts not only on the modes of the single-step, but also on the N-step prediction model. As a result, additional cuts on the $\mathcal{X} \times \mathcal{U}^N$ space can be added taking into account the interactions between the single-step models.

Cuts can be formulated in terms of additional logic constraints. According to Mignone (2002), two methods can be used to transform logic constraints into mixed integer inequalities, which can be directly added to the MLD model. The *Symbolical Method* converts the constraints into a canonical normal form, which is then translated into integer inequalities, whereas the *Geometrical Method* computes the convex hull of the integer points for which the constraints are fulfilled. In general, the second method is superior to the first one because the convex hull is the smallest set containing all the feasible integer points, and because it introduces fewer additional inequalities.

Example 5.2

As an example for adding cuts to the single-step prediction model, reconsider the paperboy

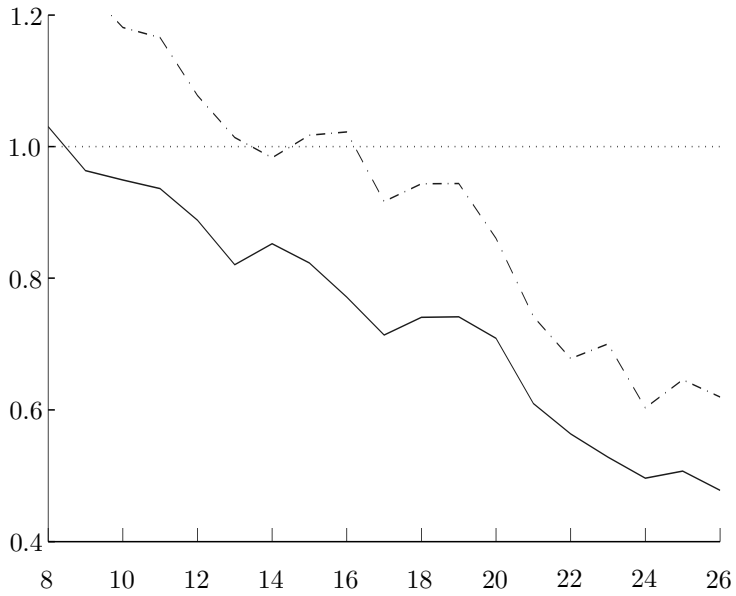


Figure 12. Normalized average computation time versus prediction horizon N, $N = 8, \ldots, 26$, when using single-step prediction models with cuts generated by either the symbolical method (dash-dotted line) or the geometrical method (solid line)

example of the previous section. We used the compiler HYSDEL to transform the paperboy composition of DHAs into an MLD model. The mode enumeration algorithm in Section 4 was used to compute the set of (feasible) modes, which were added as additional constraints to the MLD model using the symbolical method as well as the geometrical method. We solved MPC with the cost function (12) and various prediction horizons. Using CPLEX (ILOG, Inc. 2002) as MILP solver, Fig. 12 reports the average computation times for MPC on the two improved models normalized to the plain model produced by the HYSDEL compiler.

As can be seen, both methods add non-trivial cuts to the model reducing the computation time of CPLEX by a factor of up to two. This improvement is more evident when using less advanced solvers like the one described in Bemporad and Mignone (2000), where for a prediction horizon of three for example, the additional information reduces the computation time by a factor of 210. Figure 12 shows clearly that the cuts introduced by the geometrical method are more effective than the ones of the symbolical method. This is mainly due to the fact that the symbolical method needs much more constraints and consequently memory space to define the feasible modes thus delaying the calling of CPLEX. For the paperboy example, the symbolical method introduces 239 additional constraints, whereas the geometrical method only adds 42. The third conclusion is that both methods become more effective as the prediction horizon is increased, as the benefit of additional cuts grows with the number of binary variables.

6 Conclusions and Future Research

We have presented an effective method to enumerate the set of feasible modes for a given composition of DHAs. The same procedure transforms the compound model into a PWA model. The algorithm is capable of handling algebraic feedback loops in such a composition — in contrast to HYSDEL that is not able to transform compositions containing loops into equivalent MLD models. As a byproduct, the algorithm can also determine whether a composition is well-posed or not. Improving the way data is handled and by storing hyperplane arrangements already computed, led to a reduction of the computation time by a factor of up to 30 when compared to the original code presented in Geyer et al. (2003).

In general, some neighboring polyhedra have the same PWA dynamic and should thus be

joined in order to reduce the model complexity. This problem is addressed in Geyer, Torrisi, and Morari (2008), where optimal complexity reduction algorithms are proposed to derive — based on the markings of a corresponding hyperplane arrangement — an equivalent PWA model that is minimal in the number of polyhedra.

With respect to optimal control, the mode enumeration technique can be exploited to reduce the computation time of MPC by adding integer constraints. Here, we have added such cuts only to the single-step prediction model. Extending this idea to the N-step prediction model should further increase the benefits in terms of a reduction of the on-line computation time. Another promising alternative is to transform the N-step prediction model into an equivalent PWA model, to derive a minimal representation using the optimal complexity reduction scheme of Geyer et al. (2008), and to translate the minimal representation then into a very compact MLD model to be used for MPC.

Acknowledgements

This work was partially supported by the EU research project IST-2001-33520 *Control and Computation*. The work was conceived and largely prepared when the first two authors were with the Automatic Control Laboratory at ETH Zurich. The authors would like to thank Alberto Bemporad and Komei Fukuda for inspiring discussions, and Mato Baotić and Komei Fukuda for sharing their tools.

References

R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In A. Mazurkiewicz and J. Winkowski, editors, *Proc. CONCUR 97*, volume 1243 of *LNCS*, pages 74–88. Springer, 1997.

R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivančić, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky. Hierarchical hybrid modeling of embedded systems. In T.A. Henzinger and C.M. Kirsch, editors, *Embedded Software First International Workshop. Proceedings*, volume 2211 of *LNCS*, pages 14–31. Springer, 2001.

D. Avis and K. Fukuda. Reverse search for enumeration. *Discr. App. Math.*, 65:21–46, 1996.

A. Balluchi, L. Benvenuti, M. Di Benedetto, C. Pinello, and A. Sangiovanni-Vincentelli. Automotive engine control and hybrid systems: Challenges and opportunities. *Proc. IEEE*, 88(7):888–912, 2000.

A. Bemporad. An efficient technique for translating mixed logical dynamical systems into piecewise affine systems. In *Proc. IEEE Conf. on Decision and Control*, Las Vegas, NV, December 2002.

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`.

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

L. Benvenuti, A. Ferrari, E. Mazzi, and A. Sangiovanni-Vincentelli. Composing hybrid systems. pages 4693–4699, Cancun, Mexico, December 2008.

F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. A hybrid approach to traction control. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 162–174. Springer, 2001.

R. C. Buck. Partition of space. *American Math. Monthly*, 50:541–544, 1943.

N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, 1974.

P. Eades, X. Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47:319–323, 1993.

M. G. Earl and R. D'Andrea. Modeling and control of a multi-vehicle system using mixed integer linear programming. In *Proc. IEEE Conf. on Decision and Control*, Las Vegas, NV, December 2002.

H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.

J. A. Ferrez, K. Fukuda, and Th. M. Liebling. Cuts, zonotopes and arrangements. Technical report, EPF Lausanne, Switzerland, November 2001.

T. Geyer. *Low Complexity Model Predictive Control in Power Electronics and Power Systems*. Dr. sc. tech. thesis, Automatic Control Laboratory ETH Zurich, 2005.

T. Geyer, F. D. Torrisi, and M. Morari. Efficient mode enumeration of compositional hybrid systems. In

A. Pnueli and O. Maler, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *LNCS*, pages 216–232. Springer, 2003.

T. Geyer, F.D. Torrisi, and M. Morari. Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, 44(7):1728–1740, 2008.

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

T. A. Henzinger, M. Minea, and V. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034, pages 275–290. Springer, March 2001.

J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee. Hybrid modeling of TCP congestion control. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *LNCS*, pages 291–304. Springer, 2001.

ILOG, Inc. *CPLEX 8.0 User Manual*. Gentilly Cedex, France, 2002.

M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi parametric toolbox (MPT). In R. Alur and G. Pappas, editors, *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 448–462. Springer, 2004. `http://control.ee.ethz.ch/~mpt`.

J.M. Maciejowski. *Predictive Control*. Prentice Hall, 2002.

D. Mignone. *Control and Estimation of Hybrid System with Mathematical Optimization*. Dr. sc. tech. thesis, Automatic Control Laboratory ETH Zurich, Zurich, 2002.

B. Potocnik, G. Music, and B. Zupancic. A new technique for translating discrete hybrid automata into piecewise affine systems. 10(1):41–57, 2004.

S. Rashid and J. Lygeros. Hybrid systems: Modeling, analysis and control – open hybrid automata and composition. UCB/ERL M99/34, Lecture notes of the class EECS 291e, Lecture 8, 1999.

E. D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. Automat. Contr.*, 26 (2):346–358, April 1981.

E. D. Sontag. Interconnected automata and linear systems: A theoretical framework in discrete-time. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III Verification and Control*, volume 1066 of *LNCS*, pages 436–448. Springer, 1996.

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

F. D. Torrisi and A. Bemporad. Hysdel — a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Trans. Contr. Syst. Technol.*, 12(2):235–249, March 2004.

A. J. van der Schaft and J. M. Schumacher. Complementarity modelling of hybrid systems. *IEEE Trans. Automat. Contr.*, 43:483–490, 1998.

J.L. Villa, M. Duque, A. Gauthier, and N. Rakoto-Ravalontsalama. A new algorithm for translating MLD systems into PWA systems. In *Proc. American Control Conf.*, pages 1208–1213, Boston, MA, June 2004.