DISC School - April 2003

Modeling, Control, and Reachability Analysis of Discrete-Time Hybrid Systems

Alberto Bemporad

Dept. of Information Engineering
University of Siena, Italy

bemporad@unisi.it
http://www.dii.unisi.it/~bemporad

March 30, 2003

Abstract

These notes describe a computational framework for modeling, controlling, and analyzing hybrid systems in discrete-time. We describe three classes of hybrid models in detail: discrete hybrid automata (DHA), piecewise affine (PWA) systems, and mixed logical dynamical (MLD) systems. We show the relations among such model classes and other existing model paradigms, such as (extended) linear complementarity ((E)LC) systems and min-max-plus-scaling (MMPS) systems. We present the language HYSDEL (HYbrid Systems DEscription Language), a high level modeling language for describing discrete-time hybrid systems, and a set of tools for translating DHA into any of the former hybrid models.

We describe controller synthesis strategies based on the receding horizon solution to finite-time optimal control problems via mixed-integer programming and via multiparametric programming, by extending to hybrid systems ideas and results that exist for Model Predictive Control (MPC) of linear systems.

We also present a methodology for computing the set of states that a discrete-time affine hybrid system can reach by starting from a given set of initial conditions and under the effect of exogenous inputs to the system, such as disturbances, within a prescribed range. We show how to use reachability analysis to assess robust stability, safety, and liveness properties of the hybrid system, and how the reach-set computation machinery can be embedded in an optimization procedure to determine solve quite general hybrid optimal control problems.

Finally, we present a complete automotive case study showing the modeling capabilities of HYSDEL and how the different models allow to use several computational tools.

Contents

Chapter 1

Introduction

The mathematical model of a system is traditionally associated with differential or difference equations, typically derived from physical laws governing the dynamics of the system under consideration. Consequently, most of the control theory and tools have been developed for such systems, in particular for systems whose evolution is described by smooth linear or nonlinear state transition functions. On the other hand, in many applications the system to be controlled is also constituted by parts described by *logic*, such as for instance on/off switches or valves, gears or speed selectors, and evolutions dependent on if-then-else rules. Often, the control of these systems is left to schemes based on heuristic rules inferred from practical plant operation.

Recent technological innovations have caused a considerable interest in the study of dynamical processes of a heterogeneous continuous and discrete nature, denoted as *hybrid systems*. The peculiarity of hybrid systems is the interaction between continuous-time dynamics (governed by differential or difference equations), and discrete dynamics and logic rules (described by temporal logic, finite state machines, if-then-else conditions, discrete events, etc.) and discrete components (on/off switches, selectors, digital circuitry, software code, etc.).

Hybrid systems switch among many operating modes, where each mode is governed by its own characteristic dynamical laws. Mode transitions are triggered by variables crossing specific thresholds (state events), by the elapse of certain time periods (time events), or by external inputs (input events) [4]. A typical example of hybrid systems are embedded systems, constituted by dynamical components governed by logical/discrete decision components. Complex systems organized in hierachical way, where for instance discrete planning algorithms at the higher level interact with continuous control algorithms and processes at the lower level, are another example of hybrid systems. In these systems, a hierarchical organization helps managing the complexity of the system, as higher levels in the hierarchy require less detailed models (=abstractions) of the functioning of the lower levels. Two main categories of hybrid systems were successfully adopted for analysis and synthesis purposes [44]: *hybrid control systems* [3, 5, 31, 101, 103], which consist of the interaction between continuous dynamical systems and discrete/logic automata (Fig. 1.1), and *switched systems* [45, 93, 128, 138], where the state-space is partitioned into regions, each one being associated with a different continuous dynamics (Fig. 1.2).

Hybrid systems arise in a large number of application areas and are attracting increasing attention in both academic theory-oriented circles as well as in industry, for instance the automotive industry [9, 10, 14, 23, 94]. Moreover, many physical phenomena admit a natural hybrid description, like circuits integrating relays or diodes, biomolecular networks [2], and TCP/IP networks in [90].

As an example of hybrid control problem consider the design of a cruise control system that commands the gear shift, the engine torque, and the braking force in order to

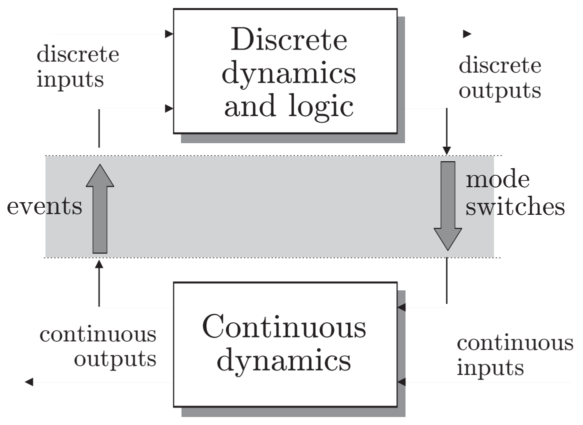


Figure 1.1: Hybrid systems. Logic-based discrete dynamics and continuous dynamics interact through events and mode switches

track a desired vehicle speed while minimizing fuel consumption and emissions. Designing a control law that optimally selects both the discrete inputs (gears) and continuous inputs (torque and brakes) requires a hybrid model that includes the continuous dynamics of the power train, the discrete logic of the gearbox, and consumption/emission maps [14]. A simplified version of this problem will be dealt with in Chapter 5.

A gasoline engine has also a natural hybrid representation: the power train, gas flow, and thermal dynamics are continuous processes, while the pistons have four modes of operation which can be described as a discrete event process or a finite state machine [10]. These two heterogeneous processes interact tightly, as the timing of the transitions between two phases of the pistons is determined by the continuous dynamics of the power train, which, in turn, depends on the torque produced by each piston.

In most cases, the synthesis of control schemes for systems having a discrete and continuous dynamical nature is still approached with heuristic rules, usually driven by engineering insight and experience, with a consequently long design and verification process. The interest of the control community is motivated by several clearly discernible trends in industry which point toward an extended need for new tools to design control/supervisory schemes for hybrid systems and to analyze their stability, safety, and performance.

In the theory of hybrid systems, several problems are investigated, such as: definition and computation of trajectories, stability and safety analysis, control, state estimation, etc.

The definition of trajectories is usually associated with a *simulator*, a tool able to compute the time evolution of the variables of the system. This may seem straightforward at first, however some hybrid formalisms introduce extra behavior like Zeno effects [95], that complicate the definition of trajectories. Although simulation allows to probe the model, it certainly does not permit structural properties of the model to be assessed. In fact any analysis based on simulation is likely to miss the subtle phenomena that a model may generate, especially in the case of hybrid models.

Tools like *reachability analysis* and *piecewise quadratic Lyapunov stability* are becoming a standard in analysis of hybrid systems. Reachability analysis (or *safety analysis* or *formal verification*), which will be the topic of Chapter 4, aims at detecting if a hybrid model will eventually reach an unsafe state configuration or satisfy a temporal logic formula [3]. Reachability analysis relies on a reach set computation algorithm, which is strongly related to the mathematical model of the system [127].

Piecewise quadratic Lyapunov stability [69, 93], is a deductive way to prove the stability of an equilibrium point of a subclass of hybrid systems (piecewise linear systems), the computational burden is usually low, at the price of a convex relaxation of the prob-

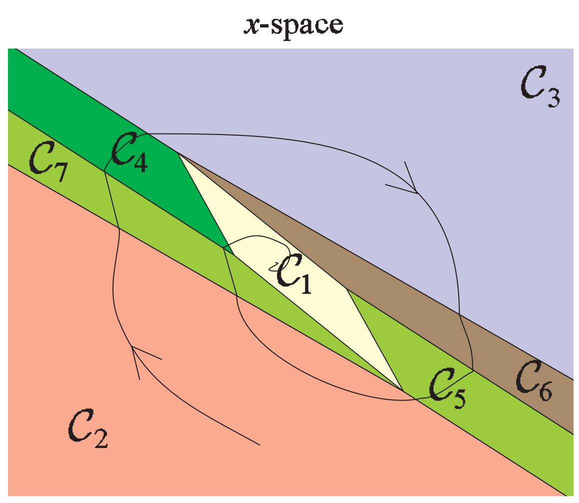


Figure 1.2: Piecewise affine systems. Mode switches are triggered by threshold events

lem which leads to conservative results. While for pure linear systems there exists a complete theory for the *identification* of unknown system parameters, the extension to general hybrid systems is still under investigation.

Controlling a model (and therefore a process) means choosing the input such that the output tracks some desired reference. The control (or *scheduling*) problem can be tackled in several ways, according to the model type and control objective. Most of the control approaches are based on optimal control ideas [139]. The dual problem of control is *state estimation*, which amounts to compute the value of unmeasurable state variables based on the measurements of output variables. The main applicative relevance of state estimation is for control, when direct measurements of the state vector are not possible, and for problems of monitoring and fault detection.

After the seminal work [137], where a class of hybrid-state continuous-time dynamical systems was formulated and an optimal control problem examined, several modelling frameworks for hybrid systems have appeared in the literature, we refer the interested reader to [4, 44, 51, 102] and references therein. Each class is usually tailored to solve a particular problem, and many of them look largely dissimilar, at least at first sight.

Timed automata and hybrid automata have proved to be a successful modeling framework for formal verification (see [127] and the references contained therein) and have been widely used in the literature. The starting point for both models is a finite state machine equipped with continuous dynamics. In the theory of *timed automata*, the dynamic part is the continuous-time flow $\dot{x} = 1$. Efficient computational tools complete the theory of timed automata and allow one to perform verification and scheduling of such models. Timed automata were extended to *linear hybrid automata* [3], where the dynamics is modeled by the differential inclusion $a \leq \dot{x} \leq b$. Specific tools allow one to verify such models against safety and liveness requirements. Linear hybrid automata were further extended to *hybrid automata* where the continuous dynamics is governed by differential equations. Tools exist to model and analyze those systems, either directly or by approximating the model with timed automata or linear hybrid automata [127].

In these notes we will focus on discrete-time hybrid systems, that we will call *discrete hybrid automata* (DHA), whose continuous dynamics is described by linear difference equations and whose discrete dynamics is described by finite state machines, both synchronized by the same clock. This will be the topic of Chapter 2.

A particular case of DHA is the popular class of *piecewise affine* (PWA) systems first introduced by Sontag [128]. Essentially, PWA are switched affine systems whose mode only depends on the current location of the state vector. More precisely, the state space is partitioned into polyhedral regions, as depicted in Figure 1.2, and each region is associated with a different affine state-update equation (more generally, the partition is defined in the combined space of state and input vectors). We will actually show that DHA and PWA systems are equivalent model classes, and hence, in particular, that generic DHA systems can be converted to equivalent PWA systems.

Another popular class of hybrid systems is the class of *linear complementarity* (LC) systems [53, 87, 88, 126, 134]. LC systems were mainly investigated in continuous-time, and applications include constrained mechanical systems, electrical networks with ideal diodes or other dynamical systems with piecewise linear relations, variable structure systems, constrained optimal control problems, projected dynamical systems and so on [87, Ch. 2]. Issues related to modeling, well-posedness (existence and uniqueness of solution trajectories), simulation and discretization have been of particular interest.

In Chapter 2 we will show that DHA models are a mathematical abstraction of the features provided by other computational oriented and domain specific hybrid frameworks: *Mixed logical dynamical* (MLD) *models* [31], the aforementioned piecewise affine (PWA) systems [128] and linear complementarity (LC), *extended linear complementarity* (ELC) *systems* [62, 63], and *max-min-plus-scaling* (MMPS) *systems* [65]. In particular, as shown in [86] all those modeling frameworks are equivalent (possibly under some hypothesis) and it is possible to represent the same system with models of each class.

As we already mentioned, we will work with discrete-time hybrid models. Despite the fact that the effects of sampling can be neglected in most applications, we note, however, that interesting mathematical phenomena occurring in hybrid systems, such as Zeno behaviors [95] do not exist in discrete-time. On the other hand, most of these phenomena are usually a consequence of the continuous-time switching model, rather than the real natural behavior. Our main motivation for concentrating on discrete-time models stems from the need to analyze these systems and to solve optimization problems, such as optimal control or scheduling problems, for which the continuous-time counterpart would not be easily computable. Although it is possible to consider hybrid automata in continuous-time, several computational tools profit from the discretization of time[1]. As anticipated DHA generalize many computational oriented models for hybrid systems and therefore represent the starting point for solving complex analysis and synthesis problems for hybrid systems.

In particular the MLD and PWA frameworks allow one to recast reachability/observability analysis, optimal control, and estimation as mathematical programming problems. Reachability analysis algorithms were developed in [37] for stability and performance analysis of hybrid PWA systems. In [25, 132] the authors also presented a novel approach for solving scheduling problems using combined reachability analysis and quadratic optimization for MLD and PWA models. For feedback control, in [31] the authors propose a model predictive control scheme which is able to stabilize MLD systems on desired reference trajectories while fulfilling operating constraints, and possibly take into account previous qualitative knowledge in the form of heuristic rules. This will be the topic of Chapter 3. Similarly, the dual problem of state estimation admits a receding horizon solution scheme [30, 71, 107].

Several authors focused on the problem of solving optimal control problems for hybrid systems. For continuous-time hybrid systems, most of the literature either studied necessary conditions for a trajectory to be optimal [116, 129], or focused on the computation of optimal/suboptimal solutions by means of dynamic programming or the maximum principle [46, 47, 82, 85, 100, 121, 138, 139].

The hybrid optimal control problem becomes less complex when the dynamics is

[1]Also some tools for continuous-time hybrid models perform internally a time discretization of the model in order to execute the computations [127].

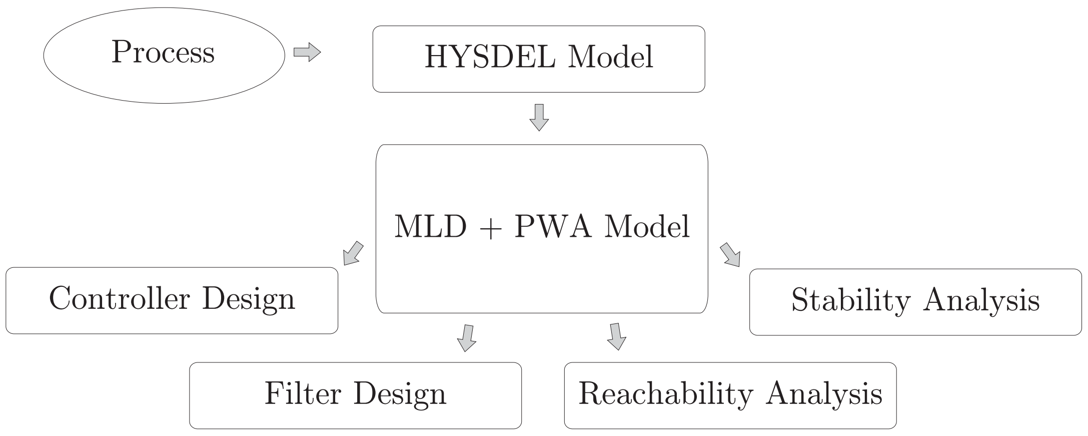


Figure 1.3: Design flow: The process is modeled in HYSDEL. The model is automatically translated into MLD and PWA form, and used for analysis/synthesis

expressed in discrete-time, as the main source of complexity becomes the combinatorial (yet finite) number of possible switching sequences. In particular, in [16, 31, 131] the authors have solved optimal control problems for discrete-time hybrid systems by transforming the hybrid model into a set of linear equalities and inequalities involving both real and (0-1) variables, so that the optimal control problem can be solved by a mixed-integer programming (MIP) solver.

In these notes we will refer to the tool HYSDEL (HYbrid Systems DEscription Language), a high level language for modeling and simulating DHA, and for automatically translating DHA into MLD and PWA models. The idea is to model hybrid systems as DHA using HYSDEL, then use MLD and PWA models as a computationally convenient model — defined by a collection of equalities and inequalities and therefore often hard to determine by hand – for analysis and synthesis purposes. The idea is depicted in Figure 1.3/

Finally, we mention that identification techniques for piecewise affine systems were recently developed [20, 34, 72, 97], that allow one to derive models (or parts of models) from input/output data.

Chapter 2

Modeling

In this chapter we introduce the basic discrete time hybrid models considered in this notes: piecewise affine (PWA) models, mixed logical dynamical (MLD) models, and discrete hybrid automata (DHA).

After introducing PWA systems, we will go through the steps needed for modeling a system as a DHA. We will first detail the process of translating propositional logic involving Boolean variables and linear threshold events over continuous variables into mixed-integer linear inequalities, generalizing several results available in the literature [31, 119, 136], in order to get an equivalent MLD form of a DHA system. We will also recall the key equivalence results among several classes of discrete-time hybrid systems, so that existing analysis and synthesis tools developed for a particular class can be easily transferred to the other classes.

We will present the tool HYSDEL (=HYbrid Systems DEscription Language), that allows describing the hybrid dynamics in a textual form, and a related compiler which provides different model representations of the given hybrid dynamics. The HYSDEL compiler is available at `http://control.ee.ethz.ch/~hybrid/hysdel`.

A complete automotive case study is reported in Chapter 5, where we will derive a model of the car engine and power train, and, using that model, solve a controller synthesis and a safety analysis problem.

2.1 Piecewise Affine (PWA) Systems

PWA systems [86, 128] are defined by partitioning the space of states and inputs into polyhedral regions (cf. Figure 1.2) and associating with each region a different linear state-update equation

$$\begin{aligned}
x'(k) &= A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)} && \text{(2.1a)}\\
y(k) &= C_{i(k)}x(k) + D_{i(k)}u(k) + g_{i(k)} && \text{(2.1b)}\\
i(k) &\quad \text{such that} && \\
& H_{i(k)}x(k) + J_{i(k)}u(k) \leq K_{i(k)}, && \text{(2.1c)}\\
& \tilde{H}_{i(k)}x(k) + \tilde{J}_{i(k)}u(k) < \tilde{K}_{i(k)}, && \text{(2.1d)}
\end{aligned}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$, $y \in \mathcal{Y} \subseteq \mathbb{R}^p$, the matrices $A_{i(k)}$, $B_{i(k)}$, $f_{i(k)}$, $C_{i(k)}$, $D_{i(k)}$, $g_{i(k)}$, $H_{i(k)}$, $J_{i(k)}$, $K_{i(k)}$, $\tilde{H}_{i(k)}$, $\tilde{J}_{i(k)}$, $\tilde{K}_{i(k)}$ are constant and have suitable dimensions, $x'(k)$ denotes the successor $x(k+1)$ of $x(k)$, $i(k) \in \mathcal{I}_s \triangleq \{1, \ldots, s\}$, the inequalities in (2.1c) and (2.1d) should be interpreted component-wise and the constraints (2.1c) and (2.1d) define a polyhedral partition $\{\mathcal{P}_i\}_{i=1,\ldots,s}$ of the set $\mathcal{X} \times \mathcal{U}$. In the sequel, we denote by $\mathcal{X}_{i(k)}$ the subset of $\mathcal{X} \times \mathcal{U}$ defined by (2.1c)–(2.1d). When only numerical aspects are of interest, we can consider all the inequalities in (2.1c)–(2.1d) to be non strict, as discussed in [131].

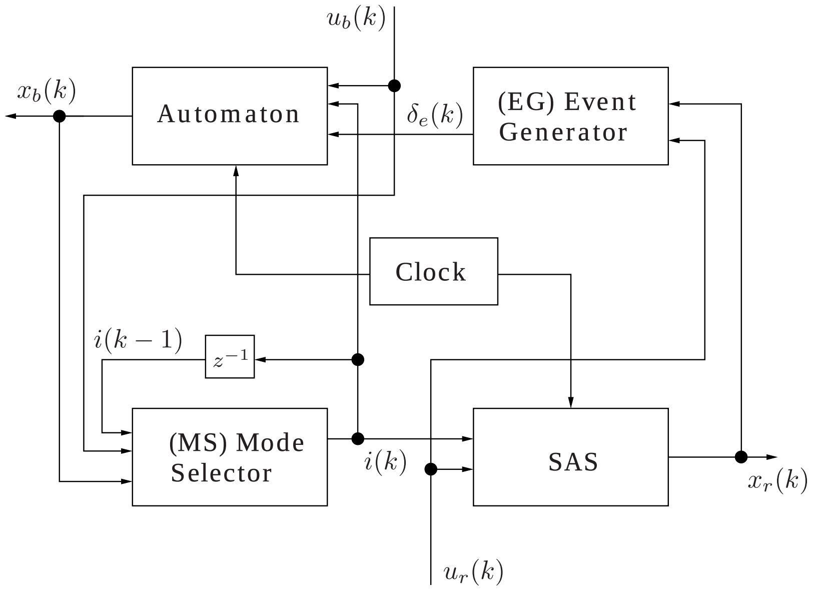


Figure 2.1: A discrete hybrid automaton (DHA) is the connection of a finite state machine (FSM) and a switched affine system (SAS), through a mode selector (MS) and an event generator (EG). The output signals are omitted for clarity

Note that the multiple definition of the state-update function over common boundaries of sets $\mathcal{X}_i$ (the boundaries will also be referred to as *guardlines*) is a technical issue that arises only when the PWA mapping is discontinuous.

PWA systems can model a large number of physical processes, such as systems with static nonlinearities, and can approximate nonlinear dynamics via multiple linearizations at different operating points.

For PWA systems, well-posedness is defined as follows

Definition 1 *A PWA system is* well-posed *on* $(\mathcal{X}, \mathcal{U}, \mathcal{Y})$*, if for all initial conditions* $x(0) = \mathcal{X}$ *and for all inputs* $u(k) \in \mathcal{U}$*, for all* $k \in \mathbb{N}$*, the state trajectory* $x(k) \in \mathcal{X}$ *and the output trajectory* $y(k) \in \mathcal{Y}$ *are uniquely defined.*

When the mode $i(k)$ is an exogenous variable, the condition in (2.1c)–(2.1d) disappears and we refer to (2.1) as a *switched affine system* (SAS).

2.2 Discrete Hybrid Automata

A discrete hybrid automaton [131] is formed by generating the mode $i(k)$ of a switched affine system through a *mode selector* function that depends on (1) the discrete state of a *finite state machine*, (2) *discrete events* generated by the continuous variables of the SAS over-passing given linear-thresholds (hyperplanes), (3) exogenous discrete inputs, and (4) time events (see Fig. 2.1).

In the following we will use the fact that any discrete variable $\alpha \in \{\alpha_1, \ldots, \alpha_j\}$, admits a Boolean encoding $a \in \{0,1\}^{d(j)}$, where $d(j)$ is the number of bits used to represent α_1, ..., α_j. From now on we will refer to either the variable or its encoding with the same name.

2.2.1 Switched Affine System *(SAS)*

A switched affine system is a collection of linear affine systems:

$$\begin{aligned} x_r'(k) &= A_{i(k)}x_r(k) + B_{i(k)}u_r(k) + f_{i(k)}, && (2.2a) \\ y_r(k) &= C_{i(k)}x_r(k) + D_{i(k)}u_r(k) + g_{i(k)}, && (2.2b) \end{aligned}$$

where $k \in \mathbb{Z}^+$ is the time indicator, $'$ denotes the successor operator ($x_r'(k) = x_r(k+1)$), $x_r \in \mathcal{X}_r \subseteq \mathbb{R}^{n_r}$ is the continuous state vector, $u_r \in \mathcal{U}_r \subseteq \mathbb{R}^{m_r}$ is the exogenous continuous input vector, $y_r \in \mathcal{Y}_r \subseteq \mathbb{R}^{p_r}$ is the continuous output vector, $\{A_i, B_i, f_i, C_i, D_i, g_i\}_{i\in\mathcal{I}}$ is a collection of matrices of opportune dimensions, and the mode $i(k) \in \mathcal{I} \triangleq \{1,\ldots,s\}$ is an input signal that chooses the affine state update dynamics. A SAS of the form (2.2) preserves the value of the state when a switch occurs, however it is possible to implement reset maps on a SAS, as shown in [131]. A SAS can be rewritten as the combination of linear terms and *if-then-else* rules: The state-update equation (2.2a) is equivalent to

$$z_1(k) = \begin{cases} A_1x_r(k) + B_1u_r(k) + f_1, & \text{if } (i(k) = 1), \\ 0, & \text{otherwise}, \end{cases} \tag{2.3a}$$

$$\vdots$$

$$z_s(k) = \begin{cases} A_sx_r(k) + B_su_r(k) + f_s, & \text{if } (i(k) = s), \\ 0, & \text{otherwise}, \end{cases} \tag{2.3b}$$

$$x_r'(k) = \sum_{i=1}^{s} z_i(k), \tag{2.3c}$$

where $z_i(k) \in \mathbb{R}^{n_r}, i = 1,\ldots,s$, and (2.2b) admits a similar transformation.

2.2.2 Event Generator *(EG)*

An event generator is a mathematical object that generates a logic signal according to the satisfaction of a linear (or affine) constraint:

$$\delta_e(k) = f_{\mathrm{H}}(x_r(k), u_r(k), k), \tag{2.4}$$

where $f_{\mathrm{H}} : \mathcal{X}_r \times \mathcal{U}_r \times \mathbb{Z}_{\geq 0} \to \mathcal{D} \subseteq \{0,1\}^{n_e}$ is a vector of descriptive functions of a linear hyperplane, and $\mathbb{Z}_{\geq 0} \triangleq \{0,1,\ldots\}$ is the set of nonnegative integers. In particular, *time events* are modeled as: $[\delta_e^i(k) = 1] \leftrightarrow [kT_s \geq t_0]$, where T_s is the sampling time and t_0 is a given time, while *threshold events* are modeled as: $[\delta_e^i(k) = 1] \leftrightarrow [a^T x_r(k) + b^T u_r(k) \leq c]$, where ^{i} denotes the i-th component of a vector.

2.2.3 Finite State Machine (*FSM*)

A finite state machine[1] (or automaton) is a discrete dynamic process that evolves according to a logic state update function:

$$x_b'(k) = f_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)), \tag{2.5a}$$

where $x_b \in \mathcal{X}_b \subseteq \{0,1\}^{n_b}$ is the Boolean state, $u_b \in \mathcal{U}_b \subseteq \{0,1\}^{m_b}$ is the exogenous Boolean input, $\delta_e(k)$ is the endogenous input coming from the EG, and $f_{\mathrm{B}} : \mathcal{X}_b \times \mathcal{U}_b \times \mathcal{D} \to \mathcal{X}_b$ is a deterministic logic function. A FSM can be conveniently represented using an oriented graph. A FSM may also have an associated Boolean output

$$y_b(k) = g_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k)), \tag{2.5b}$$

[1] In this notes we will only refer to synchronous finite state machines, where the transitions may happen only at sampling times. The adjective synchronous will be omitted for brevity.

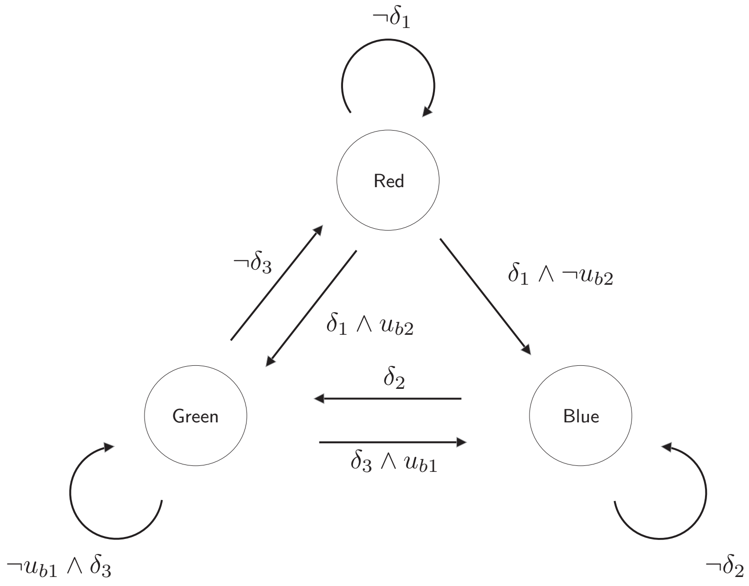


Figure 2.2: Example of finite state machine

where $y_b \in \mathcal{Y}_b \subseteq \{0,1\}^{p_b}$ and $g_b : \mathcal{X}_r \times \mathcal{U}_r \times D \mapsto \mathcal{Y}_b$. The idea of transforming a well-posed FSM into a set of Boolean equalities was already presented in [115] where the authors performed model checking using (mixed) integer optimization on an equivalent set of integer inequalities.

Example 2.2.1 Figure 2.2 shows a finite state machine where $u_b = [u_{b1}\ u_{b2}]^T$ is the input vector, and $\delta = [\delta_1 \ldots \delta_3]^T$ is a vector of signals coming from the event generator. The logic state update function or *state transition function* is:

$$x_b'(k) = \begin{cases} \textsf{Red if } ((x_b(k) = \textsf{Green}) \wedge \neg\delta_3)\vee \\ \quad ((x_b(k) = \textsf{Red}) \wedge \neg\delta_1), \\ \textsf{Green if } ((x_b(k) = \textsf{Red}) \wedge \delta_1 \wedge u_{b2})\vee \\ \quad ((x_b(k) = \textsf{Blue}) \wedge \delta_2)\vee \\ \quad ((x_b(k) = \textsf{Green}) \wedge \neg u_{b1} \wedge \delta_3), \\ \textsf{Blue if } ((x_b(k) = \textsf{Red}) \wedge \delta_1 \wedge \neg u_{b2})\vee \\ \quad ((x_b(k) = \textsf{Green}) \wedge (\delta_3 \wedge u_{b1}))\vee \\ \quad ((x_b(k) = \textsf{Blue}) \wedge \neg\delta_2)). \end{cases} \tag{2.6}$$

By associating a Boolean vector $x_b = \left[\begin{smallmatrix} x_{b1} \\ x_{b2} \end{smallmatrix}\right]$ to each state (Red $= \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$, Green $= \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$, and Blue $= \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$), one can rewrite (2.6) as:

$$\begin{aligned} x_{b1}' &= (\neg x_{b1} \wedge \neg x_{b2} \wedge \delta_1 \wedge \neg u_{b2}) \vee \\ &\quad (x_{b1} \wedge \neg\delta_2) \vee (x_{b2} \wedge \delta_3 \wedge u_{b1}), \\ x_{b2}' &= (\neg x_{b1} \wedge \neg x_{b2} \wedge \delta_1 \wedge u_{b2}) \vee \\ &\quad (x_{b1} \wedge \delta_2) \vee (x_{b2} \wedge \delta_3 \wedge \neg u_{b1}), \end{aligned}$$

where the time index (k) was omitted for brevity.

□

Note that since the logic state update function is deterministic, for each state the conditions associated to all the outgoing arcs are mutually exclusive.

2.2.4 Mode Selector (*MS*)

The logic state $x_b(k)$, the Boolean inputs $u_b(k)$, and the events $\delta_e(k)$ select the dynamic mode $i(k)$ of the SAS through a Boolean 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 this function

$$i(k) = f_{\mathrm{M}}(x_b(k), u_b(k), \delta_e(k)) \tag{2.7}$$

is called *active mode*. We say that a *mode switch* occurs at step k if $i(k) \neq i(k-1)$. Note that, in contrast to continuous-time hybrid models, where switches can occur at any time, in our discrete-time setting a mode switch can only occur at sampling instants.

2.2.5 DHA Trajectories

For a given initial condition $\left[\begin{smallmatrix} x_r(0) \\ x_b(0) \end{smallmatrix}\right] \in \mathcal{X}_r \times \mathcal{X}_b$, and input $\left[\begin{smallmatrix} u_r(k) \\ u_b(k) \end{smallmatrix}\right] \in \mathcal{U}_r \times \mathcal{U}_b$, $k \in \mathbb{Z}_{\geq 0}$, the state trajectory $x(k)$, $k \in \mathcal{Z}_{\geq 0}$ of the system is recursively computed as follows:

1. Initialization: $x(0) = \left[\begin{smallmatrix} x_r(0) \\ x_b(0) \end{smallmatrix}\right]$;

2. Recursion:

   (a) $\delta_e(k) = f_{\mathrm{H}}(x_r(k), u_r(k), k)$;

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

   (c) $y_r(k) = C_{i(k)} x_r(k) + D_{i(k)} u_r(k) + g_{i(k)}$;

   (d) $y_b(k) = g_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k))$;

   (e) $x_r'(k) = A_{i(k)} x_r(k) + B_{i(k)} u_r(k) + f_{i(k)}$;

   (f) $x_b'(k) = f_{\mathrm{B}}(x_b(k), u_b(k), \delta_e(k))$.

Definition 2 *A DHA is* well-posed *on $\mathcal{X}_r \times \mathcal{X}_b$, $\mathcal{U}_r \times \mathcal{U}_b$, $\mathcal{Y}_r \times \mathcal{Y}_b$, if for all initial conditions $x(0) = \left[\begin{smallmatrix} x_r(0) \\ x_b(0) \end{smallmatrix}\right] \in \mathcal{X}_r \times \mathcal{X}_b$, and for all inputs $u(k) = \left[\begin{smallmatrix} u_r(k) \\ u_b(k) \end{smallmatrix}\right] \in \mathcal{U}_r \times \mathcal{U}_b$, for all $k \in \mathbb{Z}_{\geq 0}$, the state trajectory $x(k) \in \mathcal{X}_r \times \mathcal{X}_b$ and output trajectory $y(k) = \left[\begin{smallmatrix} y_r(k) \\ y_b(k) \end{smallmatrix}\right] \in \mathcal{Y}_r \times \mathcal{Y}_b$ are uniquely defined.*

Definition 2 will be used for other types of hybrid models that we will introduce later. In general a hybrid model may not be well-posed, either because the trajectories stop after a finite time (for instance, the state vector leaves the set $\mathcal{X}_r \times \mathcal{X}_b$) or because of non-determinism (the successor $x_r'(k)$, $x_b'(k)$ may be multiply defined). Note that trajectories of DHA are deterministic.

DHA modes are a subclass of *Hybrid Automata* (HA) [3], the main difference is in the time model, DHA admit time in the natural numbers, while in HA the time is a real number. Moreover DHA models do not allow instantaneous transitions, and are deterministic, opposed to HA where any enabled transition may occur in zero time. This has two consequences (i) DHA do not admit live-locks (infinite switches in zero time), (ii) DHA do not admit Zeno behaviors (infinite switches in finite time). Finally in DHA models, guards, reset maps and continuous dynamics are limited to linear (or affine) functions. However, working with discrete-time models allows the development of several analysis and synthesis tools, as later reported in Chapters 3 and 4.

2.3 Discrete Hybrid Automata and Piecewise Affine Systems

This section highlights the relationships between the classes of DHA and PWA systems introduced in the previous sections.

Definition 3 *Let Σ_1, Σ_2 be hybrid models, whose inputs are $u_1(k) \in \mathcal{U}_1 \subseteq \mathcal{U}$, $u_2(k) \in \mathcal{U}_2 \subseteq \mathcal{U}$ and outputs $y_1(k) \in \mathcal{Y}_1 \subseteq \mathcal{Y}$, $y_2(k) \in \mathcal{Y}_2 \subseteq \mathcal{Y}$, $k \in \mathbb{Z}_{\geq 0}$. Let $x_1(k) \in \mathcal{X}_1 \subseteq \mathcal{X}$ be the state of Σ_1 and $x_2(k) \in \mathcal{X}_2 \subseteq \mathcal{X}$ the state of Σ_2, $k \in \mathbb{Z}_{\geq 0}$. The hybrid models Σ_1 and Σ_2 are* equivalent *on $\bar{\mathcal{X}}$,$\bar{\mathcal{U}}$,$\bar{\mathcal{Y}}$, $\bar{\mathcal{X}} \subseteq \mathcal{X}_1 \cap \mathcal{X}_2$, $\bar{\mathcal{U}} \subseteq \mathcal{U}_1 \cap \mathcal{U}_2$, $\bar{\mathcal{Y}} \subseteq \mathcal{Y}_1 \cap \mathcal{Y}_2$ if for all initial conditions $x_1(0) = x_2(0) \in \bar{\mathcal{X}}$, and for all $u_1(k) = u_2(k) \in \bar{\mathcal{U}}$, the output trajectories coincides, i.e. $y_1(k) = y_2(k)$ and $x_2(k) = x_1(k)$ at all steps $k \in \mathbb{Z}_{\geq 0}$.*

Lemma 1 *Let Σ_{PWA} be a well-posed PWA model defined on a set of states $\mathcal{X} \subseteq \mathbb{R}^n$, a set of inputs $\mathcal{U} \subseteq \mathbb{R}^m$, and a set of outputs $\mathcal{Y} \subseteq \mathbb{R}^p$. Then it can be rewritten as an equivalent well-posed DHA model Σ_{DHA} on $\mathcal{U}$,$\mathcal{X}$,$\mathcal{Y}$.*

Proof. Equations (2.1a)–(2.1b) are the modes of the SAS, the constraints $H_i x + J_i u \leq K_i$, $i = 1, \ldots, s$ are the defining hyperplanes $f_{\text{H}}(\cdot)$ of the EG, and the MS is defined by (2.1c), namely if all the events associated to the hyperplanes of $H_j x + J_j u \leq K_j$ are satisfied then $i(k) = j$. □

2.4 Logic and Mixed-Integer Inequalities

Despite the fact that DHA are rich of expressiveness and are therefore quite suitable for modeling and simulating hybrid dynamical systems, they are not directly suitable for solving synthesis and analysis problems, due to their heterogeneous discrete and continuous nature. In this section we want to describe how DHA can be translated into different hybrid models that are more suitable for computations. We highlight the main techniques of the translation process, by generalizing several results appeared in the literature [31, 52, 54, 81, 91, 109, 112, 119, 133, 135, 136].

2.4.1 Logical Functions

Boolean functions can be equivalently expressed by inequalities [54].

In order to introduce our notation, we recall here some basic definitions of Boolean algebra. A variable X is a *Boolean variable* if $X \in \{0, 1\}$. A *Boolean expression* is inductively defined[2] by the grammar

$$\begin{array}{rcl} \phi & ::= & X | \neg\phi_1 | \phi_1 \vee \phi_2 | \phi_1 \oplus \phi_2 | \phi_1 \wedge \phi_2 | \\ & & \phi_1 \leftarrow \phi_2 | \phi_1 \rightarrow \phi_2 | \phi_1 \leftrightarrow \phi_2 | (\phi_1), \end{array} \tag{2.8}$$

where X is a Boolean variable, and the logic operators $\neg$ (not), $\vee$ (or), $\wedge$ (and), $\leftarrow$ (implied by), $\rightarrow$ (implies), $\leftrightarrow$ (iff) have the usual semantics. A Boolean expression is in *conjunctive normal form* (CNF) or *product of sums* if it can be written according to the following grammar:

$$\phi \quad ::= \quad \psi | \phi \wedge \psi, \tag{2.9}$$

$$\psi \quad ::= \quad \psi_1 \vee \psi_2 | \neg X | X, \tag{2.10}$$

where ψ are called *terms of the product*, and X are the *terms of the sum* ψ. A CNF is minimal if it has the minimum number of terms of product and each term has the

[2]For the sake of simplicity, we are neglecting precedence.

	Relation	Logic	(In)equalities
L1	**AND ($\wedge$)**	$X_1 \wedge X_2$	$d_1 = 1$, $d_2 = 1$ **or** $d_1 + d_2 \geq 2$
L2	**OR ($\vee$)**	$X_1 \vee X_2$	$d_1 + d_2 \geq 1$
L3	**NOT ($\neg$)**	$\neg X_1$	$d_1 = 0$
L4	**XOR ($\oplus$)**	$X_1 \oplus X_2$	$d_1 + d_2 = 1$
L5	**IMPLY ($\rightarrow$)**	$X_1 \rightarrow X_2$	$d_1 - d_2 \leq 0$
L6	**IFF ($\leftrightarrow$)**	$X_1 \leftrightarrow X_2$	$d_1 - d_2 = 0$
L7	**ASSIGNMENT ($=$, $\leftrightarrow$)**	$X_3 = X_1 \wedge X_2$ $X_3 \leftrightarrow X_1 \wedge X_2$	$d_1 + (1 - d_3) \geq 1$ $d_2 + (1 - d_3) \geq 1$ $(1 - d_1) + (1 - d_2) + d_3 \geq 1$

Table 2.1: Basic conversion of logic relations into mixed-integer inequalities. Relations involving the inverted literals $\neg X$ can be obtained by substituting $(1 - d)$ for d in the corresponding inequalities. More conversions are reported in [107], or can be derived by (2.13)–(2.14)

minimum number of terms of sum. Every Boolean expression can be rewritten as a minimal CNF.

A Boolean expression f will be also called *Boolean function* when is used to define a literal X_n as a function of $X_1, \ldots, X_{n-1}$:

$$X_n = f(X_1, X_2, \ldots, X_{n-1}). \tag{2.11}$$

In general, we can define relations among Boolean variables $X_1, \ldots, X_n$ through a *Boolean formula*

$$F(X_1, \ldots, X_n) = 1, \tag{2.12}$$

where $X_i \in \{0, 1\}$, $i = 1, \ldots, n$. Note that each Boolean function is also a Boolean formula, but not vice versa. Boolean formulas can be equivalently translated into a set of integer linear inequalities. For instance, $X_1 \vee X_2 = 1$ is equivalent to $X_1 + X_2 \geq 1$ [136]. Some recurrent equivalences are reported in Table 2.1.

The translation can be performed automatically either using an *symbolical* method or a *geometrical* method, that we describe here below.

Symbolical Method

The symbolical method consists of first converting (2.11) or (2.12) into CNF, a task that can be performed automatically by using one of the several standard techniques available [54, 55]. Let the CNF have the form

$$\begin{array}{ll} \text{(CNF)} & \displaystyle\bigwedge_{j=1}^{m} \left(\bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \neg X_i \right) \\ & N_j, P_j \subseteq \{1, \ldots, n\},\ \forall j = 1, \ldots, m. \end{array} \tag{2.13}$$

Then, the corresponding set of integer linear inequalities is

$$\begin{cases} 1 \leq \sum_{i \in P_1} X_i + \sum_{i \in N_1} (1 - X_i), \\ \vdots \\ 1 \leq \sum_{i \in P_m} X_i + \sum_{i \in N_m} (1 - X_i). \end{cases} \tag{2.14}$$

With these inequalities we can define the set P_{CNF} for any Boolean formula F as:

$$\begin{array}{rcl} P_{\text{CNF}} & = & \{x \in [0, 1]^n : \text{(2.14) are satisfied} \\ & & \text{with } x = [X_1, \ldots, X_n]^T\}. \end{array} \tag{2.15}$$

An alternative symbolical method is described in [54].

Geometrical Method

The geometrical method consists of two steps (see e.g. [111]). First, the set of points in $[0,1]^n$ satisfying (2.11) or (2.12) is computed (for this reason, the method was also called *truth table* method in [111]). Each row of the truth table is associated with a vertex of the hypercube $\{0,1\}^n$. The vertices are collected in a set V of *valid* points, all the other points $\{0,1\}^n \setminus V$ are called *invalid*. The inequalities representing the Boolean formula are obtained by computing the convex hull of V, for which several tools are available (see e.g. [77]). We therefore define

$$P_{\mathrm{CH}} = \{x \in [0,1]^n : x \in \mathrm{conv}(V)\}. \tag{2.16}$$

Although P_{CH} and P_{CNF} contain the same integer points, i.e. $(P_{\mathrm{CH}} \cap \{0,1\}^n) = (P_{\mathrm{CNF}} \cap \{0,1\}^n)$, in general the set $P_{\mathrm{CH}} \subseteq P_{\mathrm{CNF}}$, since $\mathrm{conv}(V)$ is the smallest set containing all integer feasible points. However, there exist Boolean formulas, for which $P_{\mathrm{CH}} \neq P_{\mathrm{CNF}}$[3]. Conditions for which $P_{\mathrm{CH}} = P_{\mathrm{CNF}}$ are currently a topic of research.

2.4.2 Continuous-Logic Interfaces

Events of the form (2.4) can be equivalently expressed as

$$f^i_{\mathrm{H}}(x_r(k), u_r(k), k) \leq M^i(1-\delta^i_e), \tag{2.17a}$$
$$f^i_{\mathrm{H}}(x_r(k), u_r(k), k) > m^i\delta^i_e, \qquad i = 1,\dots,n_e, \tag{2.17b}$$

where M^i, m^i are upper and lower bounds, respectively, on $f^i_{\mathrm{H}}(x_r(k), u_r(k), k)$. As we will point out in Section 2.5, sometimes from a computational point of view, it may be convenient to have a system of inequalities without strict inequalities. In this case we will follow the common practice [136] to replace the strict inequality (2.17b) as

$$f^i_{\mathrm{H}}(x_r(k), u_r(k), k) \geq \epsilon + (m^i - \epsilon)\delta^i_e, \tag{2.17c}$$

where ϵ is a small positive scalar, e.g., the machine precision, although the equivalence does not hold for $0 < f^i_{\mathrm{H}}(x_r(k), u_r(k), k) < \epsilon$, i.e., for the numbers in the interval $(0,\epsilon)$ that cannot be represented in a computer.

The most common *logic to continuous* interface is the if-then-else construct

$$\text{IF } \delta \text{ THEN } z = a_1'x + b_1'u + f_1 \text{ ELSE } z = a_2'x + b_2'u + f_2, \tag{2.18}$$

where $\delta \in \{0,1\}$, $z \in \mathbb{R}$, $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, and $a_1, b_1, f_1, a_2, b_2, f_2$ are constants of suitable dimensions. The if-then-else construct (2.18) can be translated into [37]

$$\begin{aligned}
(m_2 - M_1)\delta + z &\leq a_2'x + b_2'u + f_2, && (2.19a)\\
(m_1 - M_2)\delta - z &\leq -a_2'x - b_2'u - f_2, && (2.19b)\\
(m_1 - M_2)(1-\delta) + z &\leq a_1'x + b_1'u + f_1, && (2.19c)\\
(m_2 - M_1)(1-\delta) - z &\leq -a_1'x - b_1'u - f_1, && (2.19d)
\end{aligned}$$

where M_i, m_i are upper and lower bounds on $a_ix + b_iu + f_i$, $i = 1,2$. Note that when $a_2 = 0$, $b_2 = 0$, $f_2 = 0$, relations (2.18)–(2.19) model the real product $z = \delta \cdot (a_1'x + b_1'u + f)$ described in [136].

2.5 Mixed Logical Dynamical Systems

Mixed logical dynamical (MLD) systems are computationally oriented representations of hybrid systems that consist of a collection of linear difference equations involving

[3]For example $(X_1 \vee X_2) \wedge (X_1 \vee X_3) \wedge (X_2 \vee X_3)$.

both real and (0-1) variables, subject to a set of linear inequalities [31]. Typically, MLD models are obtained by starting with a DHA representation of a given hybrid process, according to the techniques described in the previous section.

An MLD system is described by the following relations:

$$
\begin{aligned}
x'(k) &= Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) + B_5, && (2.20a)\\
y(k) &= Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) + D_5, && (2.20b)\\
E_2\delta(k) + E_3z(k) &\le E_1u(k) + E_4x(k) + E_5, && (2.20c)\\
\overline{E}_2\delta(k) + \overline{E}_3z(k) &= \overline{E}_1u(k) + \overline{E}_4x(k) + \overline{E}_5. && (2.20d)
\end{aligned}
$$

where $x \in \mathbb{R}^{n_r} \times \{0,1\}^{n_b}$ is a vector of continuous and binary states, $u \in \mathbb{R}^{m_r} \times \{0,1\}^{m_b}$ are the inputs, $y \in \mathbb{R}^{p_r} \times \{0,1\}^{p_b}$ the outputs, $\delta \in \{0,1\}^{r_b}$, $z \in \mathbb{R}^{r_r}$ represent auxiliary binary and continuous variables, respectively, and A, B_1, B_2, B_3, C, D_1, D_2, D_3, E_1,...,E_5 and $\overline{E}_1$,...,$\overline{E}_5$ are matrices of suitable dimensions. Given the current state $x(k)$ and input $u(k)$, the time-evolution of (2.20) is determined by solving $\delta(k)$ and $z(k)$ from (2.20c)–(2.20d), and then updating $x'(k)$ and $y(k)$ from (2.20a)–(2.20b).

We assume that system (2.20) is *completely well-posed* [31], which means that for all $x(k)$, $u(k)$ within a given bounded set the variables $\delta(k)$, $z(k)$ are defined by (2.20c)–(2.20d) in a unique way[4]. This allows assuming that $x(k+1)$ and $y(k)$ are uniquely defined once $x(k)$, $u(k)$ are given, and therefore that x- and y-trajectories exist and are uniquely determined by the initial state $x(0)$ and input signal $u(0)$, $u(1)$, ... It is clear that the well-posedness assumption stated above is usually guaranteed by the procedure used to generate the linear inequalities (2.20c), and therefore this hypothesis is typically fulfilled by MLD relations derived from modeling real-world plants. Nevertheless, a numerical test for well-posedness is reported in [31, Appendix 1].

The equations and inequalities obtained with the methods presented in Sections 2.4.1 and 2.4.2 typically contribute to defining the MLD model (2.20). Since the problems of synthesis and analysis of MLD models are tackled by optimization techniques, we have replaced strict inequalities as in (2.17b) by non-strict inequalities as in (2.17c)[5]. As observed before, indeed strict inequalities of the form $a'x > b$ can be approximated by $a'x \ge b + \epsilon$ where ϵ is a small positive scalar, e.g., the machine precision (which depends on the number of bits used for representing real numbers), although the equivalence does not hold for $0 < a'x - b < \epsilon$, that is for the numbers in the interval $(0, \epsilon)$ that cannot be represented in the machine).

Note that the constraints (2.20c) allow one to specify additional linear constraints on continuous variables (e.g., constraints over physical variables of the system), and logic constraints over Boolean variables. The ability to include constraints, constraint prioritization, and heuristics adds to the expressiveness and generality of the MLD framework. Note also that despite the fact that the description (2.20) seems to be linear, clearly the nonlinearity is concentrated in the integrality constraints over binary variables. The following simple example illustrates the technique.

Example 2.5.1 Consider the following simple switched scalar system [31]

$$
x(k+1) = \begin{cases} 0.8x(k) + u(k) & \text{if} \quad x(k) \ge 0 \\ -0.8x(k) + u(k) & \text{if} \quad x(k) < 0 \end{cases} \tag{2.21}
$$

where $x(k) \in [-10, 10]$, and $u(k) \in [-1, 1]$. The condition $x(k) \ge 0$ can be associated to a binary variable $\delta(k)$ such that

$$
[\delta(k) = 1] \leftrightarrow [x(k) \ge 0]. \tag{2.22}
$$

[4]For a more general definition of well-posedness, see [31].

[5]One may also explicitly include in (2.20) strict inequality constraints $\tilde{E}_2\delta(k) + \tilde{E}_3z(k) < \tilde{E}_1u(k) + \tilde{E}_4x(k) + \tilde{E}_5$.

By using the transformations (2.17a)–(2.17c), equation (2.22) can be expressed by the inequalities

$$-m\delta(k) \le x(k) - m \tag{2.23a}$$
$$-(M+\epsilon)\delta(k) \le -x(k) - \epsilon \tag{2.23b}$$

where $M = -m = 10$, and ϵ is a small positive scalar. Then (2.21) can be rewritten as

$$x(t+1) = 1.6\delta(k)x(k) - 0.8x(k) + u(k). \tag{2.24}$$

By defining a new variable $z(k) = \delta(k)x(k)$ which, by (2.18)–(2.19) can be expressed as

$$z(k) \le M\delta(k) \tag{2.25a}$$
$$z(k) \ge m\delta(k) \tag{2.25b}$$
$$z(k) \le x(k) - m(1-\delta(k)) \tag{2.25c}$$
$$z(k) \ge x(k) - M(1-\delta(k)), \tag{2.25d}$$

the evolution of system (2.21) is ruled by the linear equation

$$x(t+1) = 1.6z(k) - 0.8x(k) + u(k)$$

subject to the linear constraints (2.23) and (2.25).

Lemma 2 *Let $\Sigma_{\rm DHA}$ be a well-posed DHA model defined on a set of states $\mathcal{X} \subseteq \mathbb{R}^n$, a set of inputs $\mathcal{U} \subseteq \mathbb{R}^m$, and a set of outputs $\mathcal{Y} \subseteq \mathbb{R}^p$. Then for any bounded $\bar{\mathcal{X}}, \bar{\mathcal{U}}$, there exists a well posed MLD model $\Sigma_{\rm MLD}$ equivalent to $\Sigma_{\rm DHA}$ on $\bar{\mathcal{X}}, \bar{\mathcal{U}}, \mathcal{Y}$.*

Proof. Directly follows from Sections 2.4.1, 2.4.2, (2.3). □

Finally we recall that the MLD model is similar to the model presented in [67] for verification of safety properties as they both aim at translating a hybrid system in a set of mixed integer linear equalities and inequalities using similar techniques.

2.6 Other Computational Models and Further Equivalences

In the previous section we showed the equivalence relations between DHA, PWA and MLD systems. In this section, we review other existing models of linear hybrid systems and show further relationships with DHA.

2.6.1 Linear Complementarity Systems

Linear complementarity (LC) systems are given in discrete-time by the equations

$$x'(k) = Ax(k) + B_1u(k) + B_2w(k), \tag{2.26a}$$
$$y(k) = Cx(k) + D_1u(k) + D_2w(k), \tag{2.26b}$$
$$v(k) = E_1x(k) + E_2u(k) + E_3w(k) + E_4, \tag{2.26c}$$
$$0 \le v(k) \perp w(k) \ge 0, \tag{2.26d}$$

with $v(k), w(k) \in \mathbb{R}^q$ and where $\perp$ denotes the orthogonality of vectors (i.e. $v(k) \perp w(k)$ means that $v^T(k)w(k) = 0$). We call $v(k)$ and $w(k)$ the complementarity variables. A, B_i, C, D_i and E_i are real matrices [53, 87, 88, 126, 134].

2.6.2 Extended Linear Complementarity (ELC) Systems

In [62, 63, 65] it has been shown that several types of hybrid systems can be modeled as extended linear complementarity (ELC) systems:

$$x(k+1) = Ax(k) + B_1u(k) + B_2w(k) \tag{2.27a}$$

$$y(k) = Cx(k) + D_1u(k) + D_2w(k) \tag{2.27b}$$

$$E_1x(k) + E_2u(k) + E_3w(k) \leqslant g_4 \tag{2.27c}$$

$$\sum_{i=1}^{p} \prod_{j\in\phi_i} \big(g_4 - E_1x(k) - E_2u(k) - E_3w(k)\big)_j \;=\; 0, \tag{2.27d}$$

where $w(k) \in \mathbb{R}^r$ is a vector of auxiliary variables. Condition (2.27d) is equivalent to

$$\begin{array}{l}\prod_{j\in\phi_i} \big(g_4 - E_1x(k) - E_2u(k) - E_3w(k)\big)_j = 0 \\ \quad \forall i \in \{1, 2, \dots, p\}\end{array} \tag{2.28}$$

due to the inequality conditions (2.27c). This implies that (2.27c)–(2.27d) can be considered as a system of linear inequalities (i.e. (2.27c)), where there are p groups of linear inequalities (one group for each index set ϕ_i) such that in each group at least one inequality should hold with equality. Note that LC systems are a particular case of ELC systems.

2.6.3 Max-Min-Plus-Scaling (MMPS) Systems

In [65] a class of discrete event systems has been introduced that can be modelled using the operations maximization, minimization, addition and scalar multiplication. Expressions that are built using these operations are called max-min-plus-scaling (MMPS) expressions.

Definition 4 (Max-min-plus-scaling expression) *A max-min-plus-scaling expression f of the variables $x_1, \dots, x_n$ is defined by the grammar*[6]

$$f := x_i | \alpha | \max(f_k, f_l) | \min(f_k, f_l) | f_k + f_l | \beta f_k \tag{2.29}$$

with $i \in \{1, 2, \dots, n\}$, α, $\beta \in \mathbb{R}$, and where f_k, f_l are again MMPS expressions.

A MMPS expression is e.g. $2x_1+4x_2-3$, $\min(\max(-x_1, 2x_2), 3x_2+x_3)$. Note that the $\min$ operation is in fact not explicitly needed in (2.29) since we have $\min(f_k, f_l) = -\max(-f_k, -f_l)$.

MMPS systems are defined by the relations

$$x(k+1) = \mathcal{M}_x(x(k), u(k), w(k)) \tag{2.30a}$$

$$y(k) = \mathcal{M}_y(x(k), u(k), w(k)) \tag{2.30b}$$

$$\mathcal{M}_c(x(k), u(k), w(k)) \leqslant c, \tag{2.30c}$$

where $\mathcal{M}_x$, $\mathcal{M}_y$ and $\mathcal{M}_c$ are MMPS expressions in terms of the components of $x(k)$, the input $u(k)$ and the auxiliary variables $w(k)$, which are all real-valued.

Despite the fact that MMPS functions are continuous, discontinuous functions can be modeled through MMPS inequalities. For instance, $w_i(k) \in \{0, 1\}$ can be represented by the two inequalities $\max(w_i(k)-1, -w_i(k)) \geq 0$, $-\max(w_i(k)-1, -w_i(k)) \geq 0$. Similarly, in LC models it can be represented by $w_i(k)(1-w_i(k)) \leq 0$, $w_i(k) \geq 0$, $1-w_i(k) \geq 0$.

[6]The symbol | stands for OR and the definition is recursive.

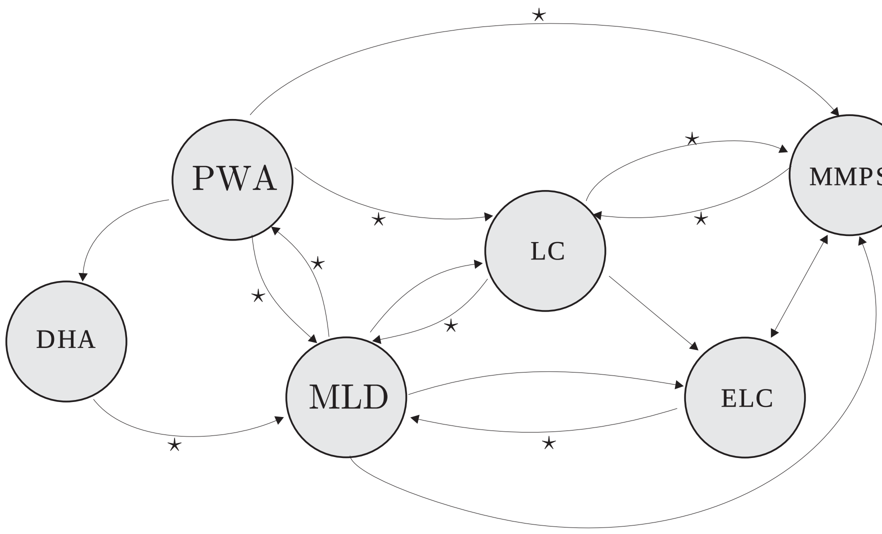


Figure 2.3: Graphical representation of the links between different classes of discrete-time hybrid systems. An arrow going from class A to class B means that A is a subset of B. Arrows with a star ($\star$) require conditions to establish the indicated inclusion.

2.6.4 Equivalence Results

In [13, 18, 86] equivalence relationships among the model classes mentioned above were proved. We summarize here below the main result.

Fact 1 *PWA, MLD, LC, ELC, and MMPS models are equivalent classes of hybrid models (certain equivalences require assumptions on the boundedness of input, state, and auxiliary variables or on well-posedness).*

Proof. See [86] for full details on assumptions, relationships, and a constructive proof. $\square$

In [131] Fact 1 was extended to also include DHA models, to obtain the relationships depicted in Figure 2.3.

Theorem 1 *Let $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$ be sets of states, inputs, and outputs respectively, and assume that $\mathcal{X}$, $\mathcal{U}$ are bounded. Then DHA, PWA, MLD, LC, ELC, and MMPS well-posed models are equivalent to each other on $\mathcal{X}, \mathcal{U}, \mathcal{Y}$.*

Proof. Mutual equivalences among PWA, MLD, LC, ELC, and MMPS on bounded $\mathcal{X}$, $\mathcal{U}$, $\mathcal{Y}$ follows from Fact 1. By Lemma 1, any PWA model Σ_{PWA} can be rewritten as an equivalent DHA model Σ_{DHA}, while any Σ_{DHA} can be rewritten as an MLD model Σ_{MLD} by Lemma 2. Therefore, any equivalence relation can be stated for any ordered pairs of models. $\square$

While there is no difference in modeling capability among the models, the same task can be solved substantially more efficiently by picking the proper model.

Each modeling framework has its advantages. For instance, stability criteria were formulated for PWA systems [93, 108]. In [87, 134] (linear) complementarity systems in *continuous* time have been studied. Applications include constrained mechanical systems, electrical networks with ideal diodes or other dynamical systems with piecewise linear relations, variable structure systems, constrained optimal control problems, projected dynamical systems and so on [87, Ch. 2]. Issues related to modeling, well-posedness (existence and uniqueness of solution trajectories), simulation and discretization have been of particular interest. Table 2.2 summarizes the advised model for several typical engineering tasks, according to the authors' knowledge of the state of the art.

Table 2.2: Advised model for each task

Task	Model
Modeling	DHA
Simulation	DHA
Control	MLD,PWA,MMPS
Stability	PWA
Verification	PWA
Identification	PWA
Fault Detection	MLD
Estimation	MLD

2.7 HYSDEL Models

A modeling language was proposed in [131] to describe DHA models, called HYbrid System DEscription Language (HYSDEL). The HYSDEL description of a DHA is an abstract modeling step. The associated HYSDEL compiler then translates the description into several computational models, in particular into a MLD using the technique presented in Section 2.4, and PWA form using either the direct approach of [79] or the indirect approach that translates the MLD into a PWA of [13]. HYSDEL can generate also a simulator that runs as a function in Matlab.

In this section we show how a DHA system can be modeled in HYSDEL by illustrating the HYSDEL description of the following DHA system:

$$\text{SAS: } x_r'(k) = \begin{cases} x_r(k) + u_r(k) - 1, & \text{if} \quad i(k) = 1, \\ 2x_r(k), & \text{if} \quad i(k) = 2, \\ 2, & \text{if} \quad i(k) = 3, \end{cases} \tag{2.31a}$$

$$\text{EG: } \begin{cases} \delta_e(k) = [x_r(k) \geq 0], \\ \delta_f(k) = [x_r(k) + u_r(k) - 1 \geq 0], \end{cases} \tag{2.31b}$$

$$\text{MS: } i(k) = \begin{cases} 1, & \text{if} \quad \begin{bmatrix} \delta_e(k) \\ \delta_f(k) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ 2, & \text{if} \quad \delta_e(k) = 1, \\ 3, & \text{if} \quad \begin{bmatrix} \delta_e(k) \\ \delta_f(k) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{cases} \tag{2.31c}$$

A complete description of the syntax of HYSDEL is available in the manual accompanying the compiler [130], and an example of realistic size is presented in Section 5.1.

Consider the HYSDEL list of Table 2.3. As any HYSDEL list, it is composed of two parts. The first one, called INTERFACE, contains the declaration of all variables and parameters, so that it is possible to make the proper type checks. The second part, IMPLEMENTATION, is composed of specialized sections where the relations among the variables are described. These sections are described next.

AUX SECTION The HYSDEL section AUX contains the declaration of the auxiliary variables used in the model. These variables will become the δ and z variables in the MLD model (2.20).

AD SECTION The HYSDEL section AD allows one to define Boolean variables from continuous ones, and is based exactly on the same semantics of the event generator (EG) described earlier. HYSDEL does not provide explicit access to the time instance, however this limitation can be easily overcome by adding a continuous state variable t such that $t' = t + T_s$, where T_s is the sampling time.

LOGIC SECTION The section LOGIC allows one to specify arbitrary functions of Boolean variables: In particular the mode selector is a Boolean function and therefore it can be modeled in this section.

DA SECTION The HYSDEL section DA defines continuous variables according to if-then-else conditions on Boolean variables. This section models part of the switched affine

Table 2.3: Sample HYSDEL list of system (2.31)

```
SYSTEM sample {
INTERFACE {
  STATE {
    REAL xr [-10, 10]; }
  INPUT {
    REAL ur [-2, 2]; }
}
IMPLEMENTATION {
  AUX {
    REAL z1, z2, z3;
    BOOL de, df, d1, d2, d3; }
  AD {
    de = xr >= 0;
    df = xr + ur - 1 >= 0; }
  LOGIC {
    d1 = ~de & ~df;
    d2 = de;
    d3 = ~de & df; }
  DA {
    z1 = {IF d1 THEN xr + ur - 1 };
    z2 = {IF d2 THEN 2 * xr };
    z3 = {IF (~de & df)  THEN 2 }; }
  CONTINUOUS {
    xr = z1 + z2 + z3; }
}}
```

system (SAS), namely the variables z_i defined in (2.3a)–(2.3b). Note that, as the definition of z3 suggests, HYSDEL can handle compound logic formulas in the DA section, therefore there is no need to explicitly define a Boolean variable for each mode.

CONTINUOUS SECTION The CONTINUOUS section describes the linear dynamics, expressed as difference equations. This section models (2.3c).

An HYSDEL description may have additional sections that are not part of the sample code of Table 2.3, that we describe below. For examples and the detailed syntax we refer the interested reader to [130].

LINEAR SECTION HYSDEL allows also one to define a continuous variable as an affine function of continuous variables in the LINEAR section. This section, together with the CONTINUOUS and AD sections allows more flexibility when modeling the SAS. This extra flexibility allows algebraic loops that may render undefined the trajectories of the model. The HYSDEL compiler integrates a semantic checker that is able to detect and report such abnormal situations.

AUTOMATA SECTION The AUTOMATA section specifies the state transition equations of the finite state machine (FSM) as a collection of Boolean functions $x'_{bi}(k) = f_{\mathrm{B}i}(x_b(k), u_b(k), \delta_e(k))$, $i = 1, \ldots, n_b$.

OUTPUT SECTION The OUTPUT section allows one to specify static linear and logic relations for the output vector $y = \left[\begin{smallmatrix} y_r \\ y_b \end{smallmatrix}\right]$.

Finally HYSDEL allows one more section:

MUST SECTION This section specifies arbitrary linear and logic constraints on continuous and Boolean variables, and therefore it allows for defining the sets $\mathcal{X}_r$, $\mathcal{X}_b$, $\mathcal{U}_r$, $\mathcal{U}_b$, $\mathcal{Y}_r$, $\mathcal{Y}_b$ (more generally, the MUST section allows also mixed constraints on states, inputs, and outputs).

Thanks to the equivalences mentioned in the previous section, it is clear that HYSDEL is a tool that allows generating several different hybrid models of a given hybrid system. In particular, HYSDEL generates MLD models, which can be immediately (and efficiently) translated into PWA systems [13, 79], or LC/ELC/MMPS systems using the constructive methods reported in [86].

2.8 A Simple Example

Consider the following simple PWA system [31]

$$\left\{ \begin{array}{rcl} x(k+1) & = & 0.8 \begin{bmatrix} \cos\alpha(k) & -\sin\alpha(k) \\ \sin\alpha(k) & \cos\alpha(k) \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \\ y(k) & = & \begin{bmatrix} 0 & 1 \end{bmatrix} x(k) \\ \alpha(k) & = & \left\{ \begin{array}{ll} \frac{\pi}{3} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \geq 0 \\ -\frac{\pi}{3} & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) < 0 \end{array} \right. \end{array} \right. \tag{2.32}$$

Assuming that $[-5, 5] \times [-5, 5]$ is the set of states $x(k)$ of interest, using HYSDEL we describe (2.32) as

```
/* 2x2 PWA system */

SYSTEM pwa {

INTERFACE {
  STATE { REAL x1,x2; }

  INPUT { REAL u; }

  OUTPUT{ REAL y; }

  PARAMETER {
      REAL Ts = 1; /* sampling time, seconds */
      REAL alpha = 1.0472; /* radiants */
      REAL C = cos(alpha);
      REAL S = sin(alpha);

      REAL umax = 1;
      REAL xmax = 10;
      REAL e = 1e-6; /* precision for strict inequalities */ }
  }

IMPLEMENTATION {
  AUX { REAL z1,z2;
        BOOL sign; }
  AD  { sign = x1<=0 [xmax,-xmax,e]; }

  DA  { z1 = {IF sign THEN 0.8*(C*x1+S*x2) [2*xmax,-2*xmax,e]
              ELSE 0.8*(C*x1-S*x2) [2*xmax,-2*xmax,e]};
        z2 = {IF sign THEN 0.8*(-S*x1+C*x2) [2*xmax,-2*xmax,e]
              ELSE 0.8*(S*x1+C*x2) [2*xmax,-2*xmax,e]};  }

  CONTINUOUS { x1 = z1;
               x2 = z2+u; }
  OUTPUT { y = x2;  }
  }
}
```

and obtain the equivalent MLD form

$$\begin{array}{l} x(k+1) = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] z(k) + \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right] u(k) \\ y(k) = \left[\begin{smallmatrix} 0 & 1 \end{smallmatrix}\right] x(k) \\ \begin{bmatrix} 20 \\ 20 \\ -20 \\ -20 \\ 20 \\ 20 \\ -20 \\ -20 \\ -5.0 \\ 5 \end{bmatrix} \delta(k) + \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} z(k) \leq \begin{bmatrix} -0.4000 & -0.6928 \\ 0.4000 & 0.6928 \\ -0.4000 & 0.6928 \\ 0.4000 & -0.6928 \\ 0.6928 & -0.4000 \\ -0.6928 & 0.4000 \\ -0.6928 & -0.4000 \\ 0.6928 & 0.4000 \\ 1.0000 & 0 \\ -1.0000 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 20 \\ 20 \\ 0 \\ 0 \\ 20 \\ 20 \\ 0 \\ 0 \\ 0.0 \\ 5 \end{bmatrix}. \end{array}$$

2.9 Identification of Hybrid Systems

Finally, we mention that identification techniques for obtaining piecewise affine models (or parts of models) from input/output data were recently developed in [12, 21, 73, 123].

Chapter 3

Controller Synthesis

3.1 Introduction

Controlling a system means to choose the command input signals so that the output signals tracks some desired reference trajectories. The control problem can be tackled in several ways, according to the model type and control objective.

The problem of determining optimal control laws for hybrid systems has been widely investigated in recent years and many results can be found in the control and computer science literature. For continuous-time hybrid systems, most of the literature is focused on the study of necessary conditions for a trajectory to be optimal [116, 129], and on the computation of optimal/suboptimal solutions by means of dynamic programming or the maximum principle [46, 47, 82, 85, 100, 121, 122, 138]. For determining the optimal feedback control law some of these techniques require the discretization of the state space in order to solve the corresponding Hamilton-Jacobi-Bellman equations. In [82] the authors use a hierarchical decomposition approach to break down the overall problem into smaller ones. In so doing, discretization is not involved and the main computational complexity arises from a higher-level nonlinear programming problem.

Optimal quadratic control of piecewise linear and hybrid systems is addressed in [85, 120], where the authors derive bounds on the solution to the associated Hamilton-Jacobi-Bellman inequalities, which are computable by solving convex optimization problems (linear matrix inequalities [120] or finite-dimensional linear programming [85]). In the case of switched linear systems composed by stable autonomous dynamics, in [27] the authors proved that the control law is a state-feedback and there exists a numerical procedure to compute the regions of the state space where the i-th switch should occur.

The hybrid optimal control problem becomes less complex when the dynamics is expressed in discrete-time or as discrete-events [49]. In the area of intelligent manufacturing and queuing systems, for example, one frequently faces *scheduling* problems. The goal of scheduling is to accomplish a given set of tasks (also identified as *jobs*) so as to optimize a meaningful performance criterion. Since the jobs to be scheduled usually involve some dynamics, the problem is hybrid. The dynamics taken into account in scheduling problems are generally very simple (often just of integral type, corresponding to timed events [50, 105]). Optimization of hybrid processes through dynamic simulation is also proposed in [114]. Here, the authors use mixed-integer linear programming (MILP) to obtain a candidate switching sequence. A standard scheduling problem is then solved for the fixed sequence.

For discrete-time linear hybrid systems, in [31] the authors showed how mixed-integer quadratic programming (MIQP) can be efficiently used to determine optimal control sequences. It was also shown that when optimal control is implemented in a receding horizon fashion by repeatedly solving MIQPs on-line, this leads to an asymptotically stabilizing control law. For those cases where on-line optimization is not viable, [16, 17] proposed multiparametric programming as an effective means for synthesizing piecewise affine optimal controllers, that solve in state-feedback form the finite-time hybrid optimal control problem with criteria based on linear (1-norm, ∞-norm) and quadratic (squared Euclidean norm) performance objectives. Such a control design flow for hybrid systems was applied to several industrial case studies, in particular to automotive problems where the simplicity of the control law is essential for its applicability [14, 24, 110].

In the discrete-time case, the main source of complexity is the combinatorial number of possible switching sequences. By combining reachability analysis and quadratic optimization, in Chapter 4 we describe a technique that rules out switching sequences that are either not optimal or simply not compatible with the evolution of the dynamical system. An algorithm to optimize switching sequences that has an arbitrary degree of suboptimality was presented in [100].

Other approaches for synthesizing controllers for piecewise affine systems using LMI relaxations was presented in [59], and for min-max-plus-scaling systems in [64].

3.2 Model Predictive Control for Hybrid Systems

For complex constrained multivariable control problems, *model predictive control* (MPC) has become the accepted standard in the process industries [106, 118]. Here at each sampling time, starting at the current state, an open-loop optimal control problem is solved over a finite horizon. The optimal command signal is applied to the process only during the following sampling interval. At the next time step a new optimal control problem based on new measurements of the state is solved over a shifted horizon. The optimal solution relies on a dynamic model of the process, respects all input and output constraints, and minimizes a performance figure. This is usually expressed as a *quadratic* or a *linear* criterion, so that, for linear prediction models, the resulting optimization problem can be cast as a quadratic program (QP) or linear program (LP), respectively, for which a rich variety of efficient active-set and interior-point solvers are available.

MPC ideas can be applied to control hybrid models. Assume we want the output $y(t)$ to track a reference signal y_e, and let x_e, u_e, δ_e, z_e be a corresponding equilibrium pair for state, input, and auxiliary variables. Let t be the current time, and $x(t)$ the current state. Consider the following optimal control problem

$$\begin{aligned}\min_{\{v,\delta,z\}_0^{T-1}} J(\{v,\delta,z\}_0^{T-1}, x(t)) \triangleq \sum_{k=0}^{T-1} &\|Q_1(v(k)-u_e)\|_p + \\ &\|Q_2(\delta(k|t)-\delta_e)\|_p + \|Q_3(z(k|t)-z_e)\|_p + \\ &\|Q_4(x(k|t)-x_e)\|_p + \|Q_5(y(k|t)-y_e)\|_p \end{aligned} \tag{3.1a}$$

$$\text{s.t.} \left\{ \begin{array}{lcl} x(t|t) & = & x(t) \\ x(k+1|t) & = & Ax(k|t) + B_1 v(k) + B_2\delta(k|t) + B_3 z(k|t) + B_5 \\ y(k|t) & = & Cx(k|t) + D_1 v(k) + D_2\delta(k|t) + D_3 z(k|t) + D_5 \\ \underline{E}_2\delta(k|t) & + & \underline{E}_3 z(k|t) \le \underline{E}_1 v(k) + \underline{E}_4 x(k|t) + \underline{E}_5 \\ \overline{E}_2\delta(k|t) & + & \overline{E}_3 z(k|t) = \overline{E}_1 v(k) + \overline{E}_4 x(k|t) + \overline{E}_5 \\ u_{\min} & \le & v(t+k) \le u_{\max},\ k = 0,1,\ldots,T-1 \\ x_{\min} & \le & x(t+k|t) \le x_{\max},\ k = 1,\ldots,T-1 \\ x(T|t) & = & x_e \end{array} \right. \tag{3.1b}$$

where T is the prediction, $x(k|t)$ is the state predicted at time $t+k$ resulting from the input $u(t+k) = v(k)$ to (2.20) starting from $x(0|t) = x(t)$, $u_{\min}$, $u_{\max}$ and $x_{\min}$, $x_{\max}$ are hard bounds on the inputs and on the states, respectively. In (3.1a), $\|Qx\|_p = x'Qx$ for $p = 2$ and $\|Qx\|_p = \|Qx\|_\infty$ for $p = \infty$, where

$$\begin{array}{lll} Q_{1,4} & = Q'_{1,4} \succ 0,\ Q_{2,3,5} = Q'_{2,3,5} \succeq 0 & (p=2) \\ Q_{1-5} & \text{nonsingular} & (p=\infty). \end{array} \tag{3.2}$$

Assume for the moment that the optimal solution $\{v_t^*(0),\ \ldots,\ v_t^*(T-1),\ \delta_t^*(0),\ \ldots,$ $\delta_t^*(T-1),\ z_t^*(0),\ \ldots,\ z_t^*(T-1)\}$ exists. According to the *receding horizon* philosophy, set

$$u(t) = v_t^*(0), \tag{3.3}$$

disregard the subsequent optimal inputs $v_t^*(1), \ldots, v_t^*(T-1)$, and repeat the whole optimization procedure at time $t+1$. The control law (3.1)–(3.3) provides an extension of MPC to hybrid models, and relies upon the solution of the mixed-integer program (3.1).

In principle all the design rules about parameter choices and theoretical results regarding stability developed for MPC over the last two decades can be applied here after some adjustments. For instance, the number of control degrees of freedom can be reduced to $N_u < T$, by setting $u(k) \equiv u(N_u - 1)$, $\forall k = N_u, \ldots, T$. While this choice usually reduces the size of the optimization problem dramatically at the price of inferior performance, here the computational gain is only partial, since all the T $\delta(k|t)$ and $z(k|t)$ variables remain in the optimization.

The end point constraint $x(T|t) = x_e$ can be relaxed by weighting the final state. However from a theoretical point of view, it is not clear how to reformulate an infinite horizon problem for an MLD system, as can be done for linear systems through weights computed from Lyapunov or Riccati algebraic equations. Indeed, infinite horizon formulations are inappropriate for both practical and theoretical reasons. In fact, approximating the infinite horizon with a large T is computationally prohibitive, as the number of 0-1 variables involved in the optimization problem depends linearly on T. Moreover, the quadratic term in δ might oscillate (for example, for a system which approaches the origin in a "switching" manner), and hence "good" (i.e. asymptotically stabilizing) input sequences might be ruled out by a corresponding infinite value of the performance index; it could even happen that no input sequence has finite cost.

3.2.1 Closed-Loop Stability

The following theorem [31] shows that the control law (3.1)–(3.3) stabilizes system (2.20) asymptotically.

Theorem 2 *Let (x_e, u_e) be an equilibrium pair and the corresponding equilibrium auxiliary variables be (δ_e, z_e). Assume that the initial state $x(0)$ is such that a feasible solution of problem (3.1) exists at time $t = 0$. Then for all matrices Q_{1-5} satisfying (3.2) the MPC law (3.1)–(3.3) stabilizes the system in that*

$$\begin{aligned}
\lim_{t\to\infty} x(t) &= x_e \\
\lim_{t\to\infty} u(t) &= u_e \\
\lim_{t\to\infty} \|Q_2(\delta(t) - \delta_e)\|_p &= 0 \\
\lim_{t\to\infty} \|Q_3(z(t) - z_e)\|_p &= 0 \\
\lim_{t\to\infty} \|Q_5(y(t) - y_e)\|_p &= 0
\end{aligned}$$

while fulfilling constraints (2.20c) and the input and state constraints $u_{\min} \le u(t) \le u_{\max}$, $x_{\min} \le x(t) \le \max$.

Note that if Q_2 (or Q_3, Q_5) is non singular, convergence of $\delta(t)$ (or $z(t)$, $y(t)$) follows as well.

Proof. The proof follows easily from standard Lyapunov arguments. Let $\mathcal{U}_t^*$ denote the optimal control sequence $\{v_t^*(0), \ldots, v_t^*(T-1)\}$, let

$$V(t) \triangleq J(\mathcal{U}_t^*, x(t))$$

denote the corresponding value attained by the performance index, and let $\mathcal{U}_1$ be the sequence $\{v_t^*(1), \ldots, v_t^*(T-2), u_e\}$. Then, $\mathcal{U}_1$ is feasible at time $t+1$, along with the vectors

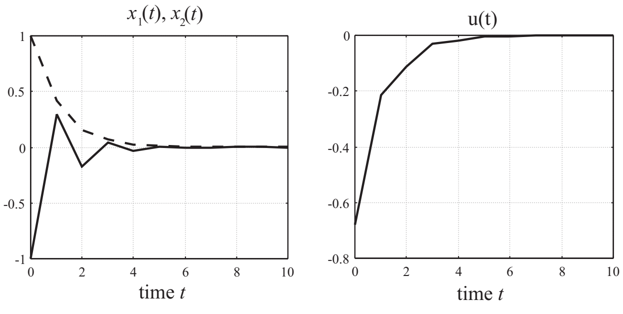


Figure 3.1: MPC closed-loop regulation to the origin of system (2.32)

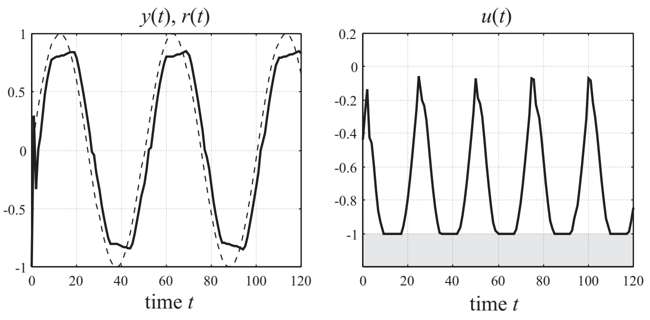


Figure 3.2: Closed-loop tracking problem for system (2.32), with $y(t) = x_2(t)$

$\delta(k|t+1) = \delta(k+1|t)$, $z(k|t+1) = z(k+1|t)$, $k = 0, \ldots, T-2$, $\delta(T-1|t+1) = \delta_e$, $z(T-1|t+1) = z_e$, because $x(T-1|t+1) = x(T|t) = x_e$. Hence,

$$\begin{aligned} V(t+1) \leq J(\mathcal{U}_1, x(t+1)) &= \\ V(t) - \|Q_4(x(t)-x_e)\|_p &+ \\ -\|Q_1(u(t)-u_e)\|_p - \|Q_2(\delta(t)-\delta_e)\|_p &+ \\ -\|Q_3(z(t)-z_e)\|_p - \|Q_5(y(t)-y_e)\|_p & \end{aligned} \tag{3.4}$$

and $V(t)$ is decreasing. Since $V(t)$ is lower-bounded by 0, there exists $V_\infty = \lim_{t\to\infty} V(t)$, which implies $V(t+1) - V(t) \to 0$. Therefore, each term of the sum

$$\begin{aligned} \|Q_4(x(t)-x_e)\|_p + \|Q_1(u(t)-u_e)\|_p &+ \\ \|Q_2(\delta(t)-\delta_e)\|_p + \|Q_3(z(t)-z_e)\|_p &+ \\ \|Q_5(y(t)-y_e)\|_p &\leq \\ V(t) - V(t+1) & \end{aligned} \tag{3.5}$$

converges to zero as well, which proves the theorem. □

Example 3.2.1 In order to stabilize system (2.32) to the origin, the feedback control law (3.1)–(3.3) is adopted, along with the parameters $T = 3$, $u_e = 0$, $\delta_e = 0$, $z_e = [0\ 0]'$, $x_e = [0\ 0]'$, $y_e = 0$, and the weights $Q_1 = 1$, $Q_2 = 0.01$, $Q_3 = 0.01 I_2$, $Q_4 = I_2$, $Q_5 = 0$, with the norm $p = 2$. Fig. 3.1 shows the resulting trajectories. Consider now a desired reference $r(t) = \sin(t/8)$ for the output $y(t)$. We apply the same MPC controller, with the exception of $Q_4 = 10^{-8} I_2$, $Q_5 = 1$. The steady-state parameters are selected as $y_e = r(t)$, and u_e, x_e, δ_e, z_e consistently. Fig. 3.2 shows the resulting closed-loop trajectories. Notice that the constraint $-1 \leq u(t) \leq 1$ prevents the system from tracking the peaks of the sinusoid, and therefore the output trajectory is chopped.

□

3.2.2 MPC Optimization Problem

The MPC formulation (3.1) can be rewritten as a *Mixed Integer Quadratic Program* (MIQP) when the squared Euclidean norm $p = 2$ is used, or as a *Mixed Integer Linear Program* (MILP), when $p = \infty$ or $p = 1$. The construction of the matrices of the mixed-integer linear program associated with MPC for the case $p = \infty$ are reported in Section 3.2.3.

Despite the fact that very effective methods exist to compute the (global) optimal solution of both MIQP and MILP problems (see Section 3.2.4 below), in the worst-case the solution time depends exponentially on the number of integer variables. In principle, this might limit the scope of application of the proposed method to very slow systems, since for real-time implementation the sampling time should be large enough to allow the worst-case computation. However, the stability proof does not require that the evaluated control sequences $\{\mathcal{U}_t^*\}_{t=0}^{\infty}$ are global optima, but only that they lead to a decrease in the objective function. Thus the solver can be interrupted at any intermediate step to obtain a suboptimal solution $\mathcal{U}_{t+1}^*$ which satisfies the decrease condition. For instance, when Branch & Bound methods are used to solve an MIQP problem, the new control sequence $\mathcal{U}_t^*$ can be selected as the solution to a QP subproblem which is integer-feasible and has the lowest value. Obviously in this case the performance deteriorates.

3.2.3 MILP Associated with MPC

The sum of the components of any vector

$$q \triangleq [\varepsilon_0^u, \ldots, \varepsilon_{T-1}^u, \varepsilon_0^\delta, \ldots, \varepsilon_{T-1}^\delta, \varepsilon_0^z, \ldots, \varepsilon_{T-1}^z, \varepsilon_1^x, \ldots, \varepsilon_{T-1}^x, \varepsilon_0^y, \ldots, \varepsilon_{T-1}^y]'$$

that satisfies

$$\begin{array}{l}
-\mathbf{1_m}\varepsilon_k^u \leq Q_1(u(k|t) - u_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_m}\varepsilon_k^u \leq -Q_1(u(k|t) - u_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_{r_\ell}}\varepsilon_k^\delta \leq Q_2(\delta(k|t) - \delta_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_{r_\ell}}\varepsilon_k^\delta \leq -Q_2(\delta(k|t) - \delta_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_{r_c}}\varepsilon_k^z \leq Q_3(z(k|t) - z_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_{r_c}}\varepsilon_k^z \leq -Q_3(z(k|t) - z_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_n}\varepsilon_k^x \leq Q_4(x(k|t) - x_e) \; k = 1, 2, \ldots, T-1 \\
-\mathbf{1_n}\varepsilon_k^x \leq -Q_4(x(k|t) - x_e) \; k = 1, 2, \ldots, T-1 \\
-\mathbf{1_p}\varepsilon_k^y \leq Q_5(y(k|t) - y_e) \; k = 0, 1, \ldots, T-1 \\
-\mathbf{1_p}\varepsilon_k^y \leq -Q_5(y(k|t) - y_e) \; k = 0, 1, \ldots, T-1
\end{array} \tag{3.6}$$

represents an upper bound on $J(v_0^{T-1}, x(t))$, where 1_h is a column vector of ones of length h, and where

$$x(k|t) = A^k x(t) + \sum_{j=0}^{k-1} A^j (B_1 u(k-1-j|t) + \tag{3.7}$$

$$B_2 \delta(k-1-j|t) + B_3 z(k-1-j|t) + B_5) \tag{3.8}$$

Similarly to what was shown in [48], it is easy to prove that the vector q that satisfies equations (3.6) and simultaneously minimizes

$$\begin{aligned}
J(q) = \; & \varepsilon_0^u + \ldots + \varepsilon_{T-1}^u + \varepsilon_0^\delta + \ldots + \varepsilon_{T-1}^\delta + \\
& \varepsilon_0^z + \ldots + \varepsilon_{T-1}^z + \varepsilon_1^x + \ldots + \varepsilon_{T-1}^x + \\
& \varepsilon_0^y + \ldots + \varepsilon_{T-1}^y
\end{aligned} \tag{3.9}$$

also solves the original problem, i.e. the same optimum $J^*(v_0^{T-1}, x(t))$ is achieved. Therefore, problem (3.1) can be reformulated as the following MILP problem

$$
\begin{array}{llcl}
\min\limits_{q} & J(q) & = & [1\ 1\ \ldots\ 1]q \\
 & & & \\
\text{s.t.} & -\mathbf{1_m}\varepsilon_k^u & \leq & \pm Q_1(u(k|t)-u_e), \quad k=0,1,\ldots,T-1 \\
 & -\mathbf{1_m}\varepsilon_k^\delta & \leq & \pm Q_2(\delta(k|t)-\delta_e), \quad k=0,1,\ldots,T-1 \\
 & -\mathbf{1_m}\varepsilon_k^z & \leq & \pm Q_3(z(k|t)-z_e), \quad k=0,1,\ldots,T-1 \\
 & -\mathbf{1_n}\varepsilon_k^x & \leq & \pm Q_4(A^k x(0|t)+\sum_{j=0}^{k-1} A^j(B_1 v(u-1-j|t)+ \\
 & & & B_2\delta(k-1-j|t)+B_3 u(k-1-j|t))-x_e), \quad k=1,\ldots,T-1 \\
 & -\mathbf{1_n}\varepsilon_k^y & \leq & \pm Q_5(CA^k x(0|t)+C\sum_{j=0}^{k-1} A^j(B_1 u(k-1-j|t)+ \\
 & & & B_2\delta(k-1-j|t)+B_3 z(k-1-j|t)))+ \\
 & & & D_1 u(k)+D_2\delta(k|t)+D_3 z(k|t)-y_e), \quad k=0,\ldots,T-1 \\
 & x_{\min} & \leq & A^k x(0|t)+\sum_{j=0}^{k-1} A^j(B_1 u(k-1-j|t)+ \\
 & & & B_2\delta(k-1-j|t)+B_3 z(k-1-j|t)) \leq x_{\max}, \quad k=1,\ldots,T \\
 & u_{\min} & \leq & u(k|t) \leq u_{\max}, \quad k=0,1,\ldots,T-1 \\
 & x(T|t) & = & x_e \\
 & x(k+1|t) & = & Ax(k|t)+B_1 u(k)+B_2\delta(k|t)+B_3 z(k|t), \quad k\geq 0 \\
 & y(k|t) & = & Cx(k|t)+D_1 u(k)+D_2\delta(k|t)+D_3 z(k|t) \\
 & E_2\delta(k|t) & + & E_3 z(k|t) \leq E_1 u(k)+E_4 x(k|t)+E_5 \quad k\geq 0 \\
 & \overline{E}_2\delta(k|t) & + & \overline{E}_3 z(k|t) = \overline{E}_1 u(k)+\overline{E}_4 x(k|t)+\overline{E}_5, \quad k\geq 0
\end{array}
\tag{3.10}
$$

where the variable $x(0|t)$ appears only in the constraints in (3.10) as a vector parameter. Problem (3.10) can be rewritten in the more compact MILP form

$$
\begin{array}{rcl}
q_t^* \triangleq & \arg\min\limits_{q} & f_c^T q_c + f_d^T q_d \\
 & & \\
 & \text{s.t.} & G_c q_c + G_c q_d \leq S + Fx(t)
\end{array}
\tag{3.11}
$$

where the matrices G, S, F can be straightforwardly defined from (3.10), and q_c, q_d represent the continuous and binary components, respectively, of the optimization vector q. The case of quadratic cost functions leads to an MIQP, whose derivation is very similar and therefore omitted here.

3.2.4 Mixed Integer Program Solvers

With the exception of particular structures, mixed-integer programming problems involving 0-1 variables are classified as $\mathcal{NP}$-complete, which means that in the worst case, the solution time grows exponentially with the problem size [119]. Despite this combinatorial nature, several packages exist for solving MILP and MIQP problems, including the commercial software Cplex [92], and other software packages [61, 75, 104, 125]. A basic MILP/MIQP solver implemented in Matlab is also freely available for download [29].

Many algorithmic approaches have been proposed and applied successfully to medium and large size application problems [76], the four major ones being

- *Cutting plane methods*, where new constraints (or "cuts") are generated and added to reduce the feasible domain until a 0-1 optimal solution is found.

- *Decomposition methods*, where the mathematical structure of the models is exploited via variable partitioning, duality, and relaxation methods.

- *Logic-based methods*, where disjunctive constraints or symbolic inference techniques are utilized which can be expressed in terms of binary variables.

- *Branch and Bound / Branch and Cut methods*, where the 0-1 combinations are explored through a binary tree, the feasible region is partitioned into sub-domains systematically, and valid upper and lower bounds are generated at different levels of the binary tree.

See [124] for a review of these methods. For MIQP, in [75] a numerical study compares the different approaches, and Branch and Bound is shown to be superior by an order of magnitude. While OA and GBD techniques can be attractive for general Mixed-Integer Nonlinear Problems (MINLP), for MIQP at each node the relaxed QP problem can be solved without approximations and reasonably quickly (for instance, the Hessian matrix of each relaxed QP is constant).

As described by [75], the Branch and Bound algorithm for MIQP consists of solving and generating new QP problems in accordance with a tree search, where the nodes of the tree correspond to QP subproblems. Branching is obtained by generating child-nodes from parent-nodes according to branching rules, which can be based for instance on a-priori specified priorities on integer variables, or on the amount by which the integer constraints are violated. Nodes are labeled as either pending, if the corresponding QP problem has not yet been solved, or fathomed, if the node has already been fully explored. The algorithm stops when all nodes have been fathomed. The success of the branch and bound algorithm relies on the fact that whole subtrees can be excluded from further exploration by fathoming the corresponding root nodes. This happens if the corresponding QP subproblem is either infeasible or an integer solution is obtained. In the second case, the corresponding value of the cost function serves as an upper bound on the optimal solution of the MIQP problem, and is used to further fathoming other nodes having greater optimal value or lower bound.

The simulation results reported here have been obtained by interfacing Matlab with the mixed-integer solvers [29, 61, 92, 104, 125]. All these have different advantages, in terms of reliability, speed, platform portability, etc.

3.3 Explicit Hybrid MPC Control Laws

There is an alternative to on-line mixed integer optimization for implementing MPC for hybrid systems.

By generalizing the result of [33] for linear systems to hybrid systems, we proposed the idea in [15, 16] of handling the state vector $x(t)$, which appears in the the linear part of the objective function and of the rhs of the constraints, as a vector of parameters. Then, for performance indices based on the ∞-norm, the optimization problem can be treated as a *multi-parametric MILP* (mp-MILP). Solving an mp-MILP amounts to express the solution of the MILP as a function of the parameters, as we will detail in Section 3.3.1.

Once the multi-parametric problem (3.1) has been solved off line, i.e., the solution $U_t^* = f(x(t))$ of (3.1) has been found, the model predictive controller is available explicitly. In [1] the authors also show that the solution $U^* = f(x)$ of the mp-MILP problem is piecewise affine. Clearly, the same properties are inherited by the controller, i.e.,

$$\begin{aligned} u(t) = \quad & F_i x(t) + g_i, \text{ for} \\ & x(t) \in \mathcal{X}_i \triangleq \{x: \ H_i x \leq S_i\}, \ i = 1, \dots, s \end{aligned} \tag{3.12}$$

where $\cup_{i=1}^{s} \mathcal{X}_i$ is the set of states for which a feasible solution to (3.1) exists.

We remark that the piecewise affine controller (3.12) and the MPC controller (3.1) *are equal*, in the sense that they produce the same control action, and therefore share the same stabilizing and optimality properties. The difference is only in the implementation: for the form (3.12), on-line computation reduces to a function evaluation, instead of an expensive mixed-integer linear program.

The technique was later extended in [17, 39] for quadratic performance indices using a combination of dynamic programming and multiparametric quadratic programming.

The explicit representation of the MPC controller discussed above is significant for several reasons. First of all, it gives some insight into the mathematical structure of the controller which is otherwise hidden behind the optimization formalism. Furthermore, it offers an alternative route to an efficient controller implementation, opening up the route to use MPC in “fast” and “cheap” systems where the on-line solution of a QP or, even worse, of an MIQP is prohibitive. Finally, the fact that we can represent the closed loop system in a PWA form allows us to apply new tools for performance analysis based on reachability analysis, as discussed in Section 4.5.

3.3.1 Multiparametric-MILP Solvers

Two main approaches have been proposed for solving mp-MILP problems. In [1], the authors developed an algorithm based on branch and bound (B&B) methods. At each node of the B&B tree an mp-LP is solved where a certain number of integer variables is relaxed to continuous values in $[0, 1]$. The solution at the root node, where all the integer variables are relaxed, represents a valid lower bound, while the solution at a leaf node where all the integer variables have been fixed to 0 or 1 represents a valid upper bound. As in standard B&B methods, the complete enumeration of combinations of 0-1 integer variables is avoided by comparing the multiparametric solutions, and by eliminating the nodes where there is no improvement of the value function. In [68] an alternative algorithm was proposed, which only solves mp-LPs where the integer variables are fixed to the optimal value determined by an MILP, instead of solving mp-LP problems with relaxed integer variables. More in detail, problem (3.11) is alternatively decomposed into an mp-LP and an MILP subproblem. First, by treating also the parameters as optimization variables, an MILP problem is solved. Then an mp-LP is solved where the binary variables are fixed to the optimal values determined by the previous MILP. The solution of the mp-LP provides a parametric upper bound. A new integer vector is determined by solving an MILP that includes an additional constraint imposing a decrease of the value function with respect to the previous mp-LP (see [68] for more details). The algorithmic implementation of the mp-MILP [68] algorithm adopted here relies on [41] for solving mp-LP problems, and on [92] for solving MILP’s.

Example 3.3.1 Consider again system (2.32). In order to optimally transfer the state from $x_0 = [-1\ 1]'$ to the origin, the performance index (3.1a) is minimized subject to (3.1b) and the MLD system dynamics (2.32), along with the weights $Q_1 = 0.001$, $Q_2 = 0.04$, $Q_3 = 30I_4$, $Q_4 = 10I_2$, $Q_5 = 700$. By solving the mp-MILP associated with this MPC

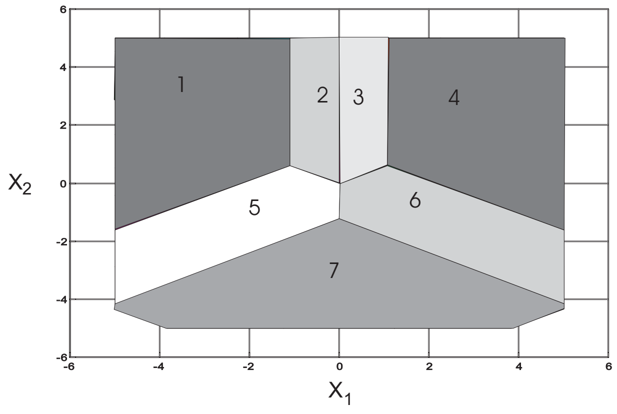


Figure 3.3: Polyhedral partition of the state-space

problem we obtain the following explicit controller:

$$u = \begin{cases} -1.0000 & \text{if} \begin{bmatrix} 1.0000 & -1.7296 \\ -1.0000 & 0.0000 \\ 0.0000 & 1.0000 \\ 94.5067 & -1.0000 \end{bmatrix} x \leq \begin{bmatrix} -2.1680 \\ 5.0000 \\ 5.0000 \\ -105.1385 \end{bmatrix} \\ & \text{(Region \#1)} \\ \begin{bmatrix} 0.9238 & 0.0000 \end{bmatrix} x & \text{if} \begin{bmatrix} -188.1504 & 1.0000 \\ -20.3158 & -34.1997 \\ 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} x \leq \begin{bmatrix} 204.3621 \\ 1.0000 \\ 0.0000 \\ 5.0000 \end{bmatrix} \\ & \text{(Region \#2)} \\ \begin{bmatrix} -0.9238 & -0.0000 \end{bmatrix} x & \text{if} \begin{bmatrix} -217.00 & 1.0000 \\ 38.1919 & -64.29 \\ 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{bmatrix} x \leq \begin{bmatrix} 4.9708 \\ 1.0000 \\ 1.0000 \\ 5.0000 \end{bmatrix} \\ & \text{(Region \#3)} \\ -1.0000 & \text{if} \begin{bmatrix} -188.15 & -1.0000 \\ -1.0000 & -1.6858 \\ 1.0000 & 0.0000 \\ 1.0000 & 133.68 \end{bmatrix} x \leq \begin{bmatrix} -204.39 \\ -2.1673 \\ 5.0000 \\ 665.60 \end{bmatrix} \\ & \text{(Region \#4)} \\ \begin{bmatrix} 0.4619 & -0.8000 \end{bmatrix} x & \text{if} \begin{bmatrix} 1.0000 & -1.7193 \\ -1.0000 & 1.7295 \\ -1.0000 & 0.0000 \\ 13.4029 & 23.6383 \\ 43.4009 & 0.0000 \end{bmatrix} x \leq \begin{bmatrix} 2.1387 \\ 2.1174 \\ 5.0000 \\ -1.0000 \\ -1.0000 \end{bmatrix} \\ & \text{(Region \#5)} \\ \begin{bmatrix} -0.4619 & -0.8000 \end{bmatrix} x & \text{if} \begin{bmatrix} 1.0000 & 1.7175 \\ -20.2527 & 34.1997 \\ -1.0000 & 0.0000 \\ -1.0000 & -1.7369 \\ 106.3247 & 1.0000 \end{bmatrix} x \leq \begin{bmatrix} 2.0413 \\ -1.0000 \\ 0.0000 \\ 2.2345 \\ 525.0259 \end{bmatrix} \\ & \text{(Region \#6)} \\ 1.0000 & \text{if} \begin{bmatrix} 1.0000 & -1.6446 \\ -1.0000 & 1.7449 \\ -1.0000 & 0.0000 \\ -1.0000 & -1.6788 \\ 1.0000 & -265.137 \\ 1.0000 & 1.7369 \\ 1.0000 & 0.0000 \end{bmatrix} x \leq \begin{bmatrix} 12.0940 \\ -2.2448 \\ 5.0000 \\ 12.3141 \\ 1329.55 \\ -2.2345 \\ 5.0000 \end{bmatrix} \\ & \text{(Region \#7)} \end{cases}$$

whose corresponding partition of the state space into polyhedral regions is depicted in Fig. 3.3.

□

Example 3.3.2 Consider the following hybrid control problem for the heat exchange example proposed by Hedlund and Ranzter [85]. The temperature of two furnaces should be controlled to a given set-point by alternate heating. Only three modes of operation are allowed: heat only the first furnace, heat only the second one, do not heat. The amount of power u_0 to be fed to the furnaces at each time instant is fixed. The system is described by the following equations:

$$\begin{cases} \dot{T} &= \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} T + B_i u_0 \\ B_i &= \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \text{if heat furnace 1} \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if heat furnace 2} \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \text{if no heating} \end{cases} \end{cases} \tag{3.13}$$

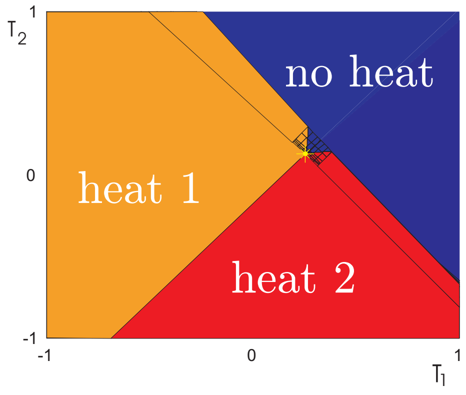
System (3.13) has two binary inputs, heat1 and heat2, which cannot be active simultaneously, is converted to a discrete-time model by sampling (3.13) for each combination of active inputs (sampling time $T_s = 0.08s$), and is described in HYSDEL as follows:

```
/* HEAT EXCHANGE model */

SYSTEM furnaces {

INTERFACE {
    STATE { REAL t1,t2,u0;
        }
    INPUT { BOOL heat1,heat2;
        }
    PARAMETER {
        REAL Ts=0.08; /* sampling time, seconds */

        REAL Bd1=0.07688365361336;
            /* matrix B, heat 1 on */
        REAL Bd2=0.07392810551689;
            /* matrix B, heat 2 on */

        REAL A11=exp(-Ts);
        REAL A22=exp(-2*Ts);
            /* discretization of matrix A */

        REAL u0max=10; /* upperbound on u0 */
        REAL e = 1e-6; /* precision for ‘‘>’’ */
        }
}

IMPLEMENTATION {
        AUX {REAL z1,z2; }

        DA {    z1 = {IF heat1 THEN Bd1*u0 [Bd1*u0max,0,e]};
                z2 = {IF heat2 THEN Bd2*u0 [Bd2*u0max,0,e]}; }

        CONTINUOUS { t1 = A11*t1+z1;
                     t2 = A22*t2+z2;
                     u0 = u0;  }

        MUST   {  ~heat1 | ~heat2;
                    /* heat1 and heat2 not both active */  }
}
}
```

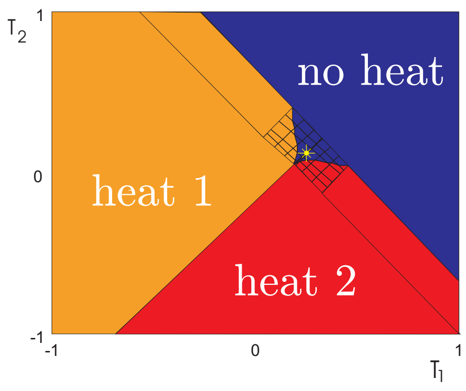
where we modeled u_0 as an additional constant continuous state. The resulting MLD model is

$$x(t+1) = \begin{bmatrix} 0.9231 & 0 & 0 \\ 0 & 0.8521 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix} x(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} z(t)$$

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} z(t) \leq \begin{bmatrix} -0.7688 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.7688 & 0 \\ 0 & -0.7393 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.7393 \\ -1.0000 & -1.0000 \end{bmatrix} u(t) + \begin{bmatrix} -0.0769 \\ 0.0769 \\ 0 \\ 0 \\ -0.0739 \\ 0.0739 \\ 0 \\ 0 \\ 0 \end{bmatrix} x_3(t) + \begin{bmatrix} 0.7688 \\ 0 \\ 0 \\ 0 \\ 0.7393 \\ 0 \\ 0 \\ 0 \\ 1.0000 \end{bmatrix}. \tag{3.14}$$



(a) Explicit optimal controller for $u_0 = 0.4$



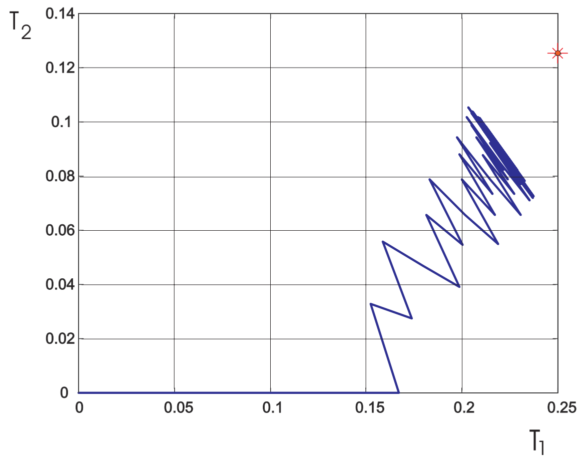
(b) Explicit optimal controller for $u_0 = 0.8$

Figure 3.4: Polyhedral partition of the state-space associated with the optimal control law

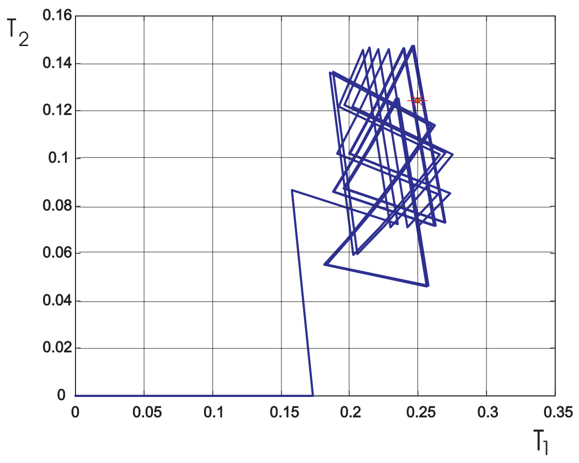
In order to optimally control the two temperatures to the desired values $T_e^1 = 1/4$ and $T_e^2 = 1/8$, the following performance index is minimized:

$$\min_J(\{v, \delta, z\}_0^2, x(t)) \triangleq \sum_{k=0}^{2} \|R(v(k+1) - v(k))\|_\infty + \|Q(T(k|t) - T_e)\|_\infty \tag{3.15}$$

subject to the MLD system dynamics (3.14), along with the weights $Q = 1$, $R = 700$. The cost function weights the tracking error and trades it off with the number of input switches occurring along the prediction horizon. By solving the mp-MILP associated with the MPC problem we obtain the explicit controller for the range $T \in [-1, 1] \times [-1, 1]$, $u_0 \in [0, 1]$ (the mp-MILP consists of 168 linear constraints, 33 continuous variables, 6 binary variables, 3 parameters, and is solved in Matlab in about 5m on a standard PC, leading to 105 polyhedral regions and affine gains). In Fig. 3.4 two slices of the three-dimensional state-space partition for different constant power levels u_0 are depicted. Around the equilibrium, the solution appears more finely partitioned, in order to perform an optimal control action. The resulting optimal trajectories are shown in Fig. 3.5. For a low power $u_0 = 0.4$ the set-point is never reached. $\square$



(a) Closed-loop MPC control of system (3.13) for $u_0 = 0.4$



(b) Closed-loop MPC control of system (3.13) for $u_0 = 0.8$

Figure 3.5: Closed-loop MPC control of system (3.13)

Chapter 4

Reachability Analysis

Models are mathematical objects that allow one to reproduce the dynamic evolution of real plants and to quantitatively analyze their properties. A good model should combine a satisfactory detail of the real system with a tractable mathematical structure in order to formulate and efficiently solve interesting applicative problems like definition and computation of trajectories, stability and safety analysis, control, state estimation, etc. As mentioned in Chapter 1, several classes of hybrid models have been proposed in the literature, each class is usually tailored to solve a particular problem or it is specific to a certain subclass of hybrid processes, and often can approximate other classes with arbitrary precision, or be equivalent under certain transformations.

A simulation of the mathematical model enables to probe the system, but very often it does not permit to assess its structural properties. In fact, any analysis based on simulation is likely to miss the subtle phenomena that a model may generate, especially in the case of hybrid models. Tools like *reachability analysis* and *piecewise quadratic Lyapunov stability* are becoming a standard in analysis of hybrid systems. Reachability analysis (or *safety analysis* or *formal verification*) aims at detecting if a hybrid model will eventually reach an unsafe state configuration or satisfy a temporal logic formula [58]. Reachability analysis relies on a reach set computation algorithm, which strongly depends on to the mathematical model of the system [127]. Piecewise quadratic Lyapunov stability is a deductive way to prove the stability of an equilibrium point of a subclass of hybrid systems (piecewise linear systems), the computational burden is usually low, at the price of a convex relaxation of the problem which may lead to conservative results.

Timed automata and hybrid automata have proved to be a successful modeling framework for formal verification, and have been widely used in the literature. The starting point for both models is a finite state machine equipped with continuous dynamics. In the theory of *timed automata* [7], the dynamic part is the continuous-time flow $\dot{x} = c \in \mathbb{R}^d$. Efficient computational tools complete the theory of timed automata and allow one to perform verification [38] and optimal control [11] of such models. Timed automata were extended to *linear hybrid automata* [3], where the dynamics is modeled by the differential inclusion $a \leq \dot{x} \leq b$. Specific tools consent the formal verification of such models against safety and liveness requirements. Linear hybrid automata were further extended to *hybrid automata* where the continuous dynamics is governed by differential equations. Tools exists to model and analyze those systems, either directly [6, 57] or by approximating the model with timed automata or linear hybrid automata.

In this chapter we present an algorithm that computes the set of states that a given DHA system can reach within $K_{\max}$ sampling times from any initial state $x(0) \in \mathcal{X}(0)$ and under any possible input excitation of $u(t) \in \mathcal{U}$, $t = 0, \ldots, K_{\max}$.

By exploiting the linearity of the dynamics in each operating mode of the DHA system, the proposed reachability algorithm computes the reach set via simple polyhedral computations, and using the properties of the Minkowski's sum it is able to detect the set of the state trajectories that have switched mode. A significantly fast analysis is obtained by over-approximating the reach set. This introduces some conservativeness that is eventually removed in a simple post-processing operation.

The importance of reach set computations is twofold. First, as shown in Section 4.4, it allows one to check for safety/liveness properties, for instance that the trajectories of the hybrid system will never enter some unsafe regions of the state space, or that all the trajectories will reach a target region within a given maximum time. Interesting control-theoretical questions like stability [35] and observability [18] can be reformulated as reachability questions. Second, the proposed algorithmic ingredients for reachability analysis can be suitably used for optimal control purposes: as shown in Section 4.6, if an optimization stage is performed in parallel with reach-set computation, the latter can be selectively carried out according to a convenient strategy that discards evolutions which are recognized not to be optimal, and finally determines the desired optimal input sequence. Compared with approaches based on mixed-integer programming [31], where the complexity depends exponentially on the prediction horizon $K_{\max}$, the method proposed here appears particularly attractive for solving hybrid optimal/receding horizon control problems where the prediction horizon is large and switching is not frequent. In particular, the larger the number of sampling steps between switches (e.g., because of a small sampling time), the more efficient the algorithm is with respect to the use of mixed-integer solvers. An additional feature of this approach is that it intrinsically embeds a practical reachability test for the system to be controlled, which is generally a prerequisite.

This chapter summarizes the results in [26, 32, 35, 37] and is mainly based on the report [132].

4.1 Preliminaries on Polyhedral Computation

The purpose of this section is to recall some basic notions and theorems on convex polyhedra that will be used later.

For two subsets $\mathcal{P}_1$ and $\mathcal{P}_2$ of $\mathbb{R}^d$, their *Minkowski's sum*, denoted by $\mathcal{P}_1 + \mathcal{P}_2$, is the set $\{x : x = x_1 + x_2, x_1 \in \mathcal{P}_1, x_2 \in \mathcal{P}_2\}$. For a finite subset $V = \{v_1, \ldots, v_n\}$ of $\mathbb{R}^d$, its convex hull $\operatorname{conv}(V)$ and the conic hull $\operatorname{cone}(V)$ are defined by

$$\operatorname{conv}(V) := \{x : x = \sum_{i=1}^{n} \lambda_i v_i, \lambda \geq 0, \sum_{i=1}^{n_1} \lambda_i = 1\}$$
$$\operatorname{cone}(V) := \{x : x = \sum_{i=1}^{n} \lambda_i v_i, \lambda \geq 0\}.$$

Here is a fundamental theorem of convex polyhedra.

Theorem 3 (Motzkin's Theorem [141, p. 30]) *For a subset $\mathcal{P} \subseteq \mathbb{R}^d$, the following two statements are equivalent:*

(a) $\mathcal{P} = \operatorname{conv}(V) + \operatorname{cone}(R)$ *for some finite subsets V and R of $\mathbb{R}^d$;*

(b) $\mathcal{P} = \{x : \ Ax \leq B\}$ *for some matrix $A \in \mathbb{R}^{m \times d}$ and some vector $B \in \mathbb{R}^m$.*

A subset $\mathcal{P}$ of $\mathbb{R}^d$ represented by either (a) or (b) is called a *convex polyhedron* or simply *polyhedron*. A *convex polytope* or simply *polytope* is a bounded convex polyhedron. A representation (V, R) of a convex polyhedron $\mathcal{P}$ is called a *V-representation*, and a representation (A, B) is an *H-representation*. A convex polyhedron given by V-representation (H-representation) is called *V-polyhedron* (*H-polyhedron*). It is possible to go from one representation to another [77], but the numerical sensitivity and complexity of those operations increases with the dimension d of the polytopes [8].

Given an H-polyhedron $\mathcal{P} = \{x : \ Ax \leq B\}$, the i-th inequality $A_i x \leq B_i$ is *redundant* for $\mathcal{P}$ if its removal preserves the polyhedron, i.e. $\mathcal{P} = \{x : \ Ax \leq B\} = \{x : \ A_j x \leq B_i \text{ for all } j \neq i\}$. Similarly, given a collection of n points V in $\mathbb{R}^d$, we say that $v \in V$ is *redundant* for V if $\text{conv}(V) = \text{conv}(V \setminus \{v\})$.

For two V-polytopes $\mathcal{P}_1 = \text{conv}(V), V = \{v_1, \ldots, v_n\}$ and $\mathcal{P}_2 = \text{conv}(W), W = \{w_1, \ldots, w_m\}$, the Minkowski's sum $\mathcal{P}_1 + \mathcal{P}_2 = \text{conv}(Z), Z = \{v_i + w_j, \forall i = 1, \ldots, n, \forall j = 1, \ldots, m\}$, where however many of the vertices in Z might be redundant. It is possible to compute a V-representation of $\mathcal{P}_1 + \mathcal{P}_2$ minimal in the number of vertices [78, 84, 113].

Given two H-polytopes $\mathcal{P}_1 = \{x : A_1 x \leq B_1\}$ and $\mathcal{P}_2 = \{x : A_2 \leq B_2\}$, the Minkowski's sum $\mathcal{P}_1 + \mathcal{P}_2 = \{x : x = x_1 + x_2, A_1 x_1 \leq B_1, A_2 x_2 \leq B_2\}$. It is possible to eliminate the variables x_1 and x_2 from the definition of $\mathcal{P}_1 + \mathcal{P}_2$ by using Fourier-Motzkin elimination [141], however, in general, the Fourier-Motzkin elimination provides an highly redundant H-representation.

Remark 4.1.1 It follows from the definition of Minkowski's sum that the maximum of a linear function fx on $\mathcal{P}_1 + \mathcal{P}_2$ is the sum of the maxima on the single polytopes, i.e.: $\max_{x \in (\mathcal{P}_1 + \mathcal{P}_2)} fx = \max_{x \in \mathcal{P}_1} fx + \max_{x \in \mathcal{P}_2} fx$, and $\arg\max_{x \in (\mathcal{P}_1 + \mathcal{P}_2)} fx = \arg\max_{x \in \mathcal{P}_1} fx + \arg\max_{x \in \mathcal{P}_2} fx$.

□

The problem of determining if a polytope $\mathcal{P}$ is contained in a polytope $\mathcal{Q}$ is known as a *containment problem* in the field of computational convexity. If $\mathcal{P}$ is a V-polytope and $\mathcal{Q}$ is given in H-representation, then the solution is trivial. If $\mathcal{P}$ and $\mathcal{Q}$ are both V- or H-polytopes then the problem has a polynomial-time solution. Finally if $\mathcal{P}$ is given in H-representation and $\mathcal{Q}$ is a V-polytope then problem is coNP-complete [83, Theorem 4.1].

The following Lemma allows one to check if the Minkowski's sum of polytopes is contained in a given H-polytope.

Lemma 3 *Given n polytopes $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$ and a H-polytope $\mathcal{Q} = \{x : Ax \leq B\}$, let $A_j x \leq B_j, j \in J = \{1, \ldots, m\}$ be the j-th inequality of $\mathcal{Q}$. $(\mathcal{P}_1 + \mathcal{P}_2 + \ldots + \mathcal{P}_n) \subseteq \mathcal{Q}$, if and only if $\sum_{i=1}^{n} (\max_{x \in \mathcal{P}_i} A_j x) \leq B_j, \forall j \in J$.*

Proof. Denote by $\mathcal{P}_s$ the Minkowski's sum $(\mathcal{P}_1 + \mathcal{P}_2 + \ldots + \mathcal{P}_n)$. We first prove the “if” part of the lemma. By Remark 4.1.1, the point $\bar{x} = \sum_{i=1}^{n} \arg\max_{x \in \mathcal{P}_i} A_j x = \arg\max_{x \in \mathcal{P}_s} A_j x$ is the point that maximizes the normal to the j-th constraint of $\mathcal{Q}$ on the set $\mathcal{P}_s$. If $A_j \bar{x} \leq B_j$ then $A_j x \leq B_j$ for all the $x \in \mathcal{P}_s$, i.e. all the points of $\mathcal{P}_s$ satisfy the j-th constraint of $\mathcal{Q}$. As this holds for all $j \in J$ then $\mathcal{P}_s \subseteq \mathcal{Q}$. For proving the “only if” part, assume by contradiction that there exists some $\bar{j}$ such that $\sum_{i=1}^{n} (\max_{x \in \mathcal{P}_i} A_{\bar{j}} x) > B_{\bar{j}}$. Then, the point $\bar{x} = \sum_{i=1}^{n} \arg\max_{x \in \mathcal{P}_i} A_{\bar{j}} x$ does not satisfy the $\bar{j}$-th constraint of $\mathcal{Q}$ but $\bar{x} \in \mathcal{P}_s$ and contradicts the hypothesis $\mathcal{P}_s \subseteq \mathcal{Q}$. □

The weighted Minkowski's sum of n polytopes $\mathcal{P}_1 \in \mathbb{R}^{d_1}$, $\mathcal{P}_2 \in \mathbb{R}^{d_2}$, ..., $\mathcal{P}_n \in \mathbb{R}^{d_n}$ with weights $R_1 \in \mathbb{R}^{d \times d_1}$, $R_2 \in \mathbb{R}^{d \times d_2}$, ..., $R_n \in \mathbb{R}^{d \times d_n}$ denoted by $R_1 \mathcal{P}_1 + R_2 \mathcal{P}_2 + \ldots + R_n \mathcal{P}_n = \sum_{i=1}^{n} R_i \mathcal{P}_i$, is defined inductively as:

1. $M_1 = \{x : R_1 x_1, x_1 \in \mathcal{P}_1\}$,
2. $M_i = M_{i-1} + R_i \mathcal{P}_i = \{x : x = x_{i-1} + R_i x_i, x_i \in M_{i-1}, x_n \in \mathcal{P}_n\}$, for $i = 2, \ldots, n$.

The weighted Minkowski's sum $R_1 \mathcal{P}_1 + R_2 \mathcal{P}_2 + \ldots + R_n \mathcal{P}_n$ is a polytope as it is the projection of the polytope $\mathcal{P}_1 \times \mathcal{P}_2 \times \ldots \times \mathcal{P}_n$ on the coordinates $x = R_1 x_1 + R_2 x_2 + \ldots R_n x_n$ [141].

Remark 4.1.2 It follows directly from the definition that a weighted Minkowski's sum $\sum_{i=1}^{n} R_i \mathcal{P}_i$ is feasible if and only if all the sets $\mathcal{P}_i$ are feasible.

□

The proof of the following is along the same lines of Lemma 3 and is therefore omitted.

Corollary 1 *Given n polytopes $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_n$, a H-polytope $\mathcal{Q}$ as in Lemma 3, let R_1, R_2, ..., R_n be matrices of suitable dimensions. Then, $\sum_{i=1}^{n} R_i\mathcal{P}_i \subseteq \mathcal{Q}$, if and only if $\sum_{i=1}^{n}(\max_{x\in\mathcal{P}_i} A_j R_i x) \leq B_j, \forall j \in J$.*

Remark 4.1.3 Note that Corollary 1 provides an algorithmic way of checking if $\sum_{i=1}^{n} R_i\mathcal{P}_i \subseteq \mathcal{Q}$ by solving $m \times n$ linear programs (LP). Moreover it provides as byproduct the indices of the constraints of $\mathcal{Q}$ violated by some point of the Minkowski's sum $\sum_{i=1}^{n} R_i\mathcal{P}_i$.

□

Remark 4.1.4 If $\mathcal{P} \subseteq \mathcal{Q}$ then $\mathcal{P} \cap \mathcal{Q} = \mathcal{P}$.

□

4.2 Reach-Set Computation

In this section we focus on the problem of computing the reach set of a piecewise affine system subject to constraints. In order to answer the control and analysis questions described in the introduction, we are interested in computing the set of states that are reachable in *exactly* k steps from $\mathcal{X}(0)$ when the input $u(t) \in \mathcal{U}, 0 \leq t < k$, and denote such a set as *(forward) reach set* $\text{Reach}_k(\mathcal{X}(0),\mathcal{U})$. The set of all the points that are reachable *within* $K_{\max}$ time steps starting from the set of initial conditions $\mathcal{X}(0)$ is therefore the union of the sets Reach_k defined above, $\text{Reachable}_{K_{\max}}(\mathcal{X}(0),\mathcal{U}) = \cup_{k\leq K_{\max}} \text{Reach}_k(\mathcal{X}(0),\mathcal{U})$. Note that already for linear systems the set $\text{Reachable}_{K_{\max}}$ could be nonconvex and disconnected.

4.2.1 Switching Sequences

The evolution of a PWA system (2.1) is given by

$$x(k) = \Pi_{j=0}^{k-1} A_{i(j)} x(0) + \sum_{j=0}^{k-1} \left(\Pi_{h=j+1}^{k-1} A_{i(k)} B_{i(j)} u(j) + f_{i(j)} \right) \triangleq \Phi_{I(k)} \begin{bmatrix} x(0) \\ u(0) \\ \vdots \\ u(k-1) \end{bmatrix} + \psi_{I(k)}, \tag{4.1}$$

where $i(t) \in \mathcal{I}_s \triangleq \{1, \ldots, s\}$ is the active mode at time t, such that the pair $x(t)$, $u(t)$ satisfies (2.1c)–(2.1d), the ordered collection of modes $I(k) \triangleq \{i(0), i(1), \ldots, i(k-1)\} \in \mathcal{I}_s^k$ is called *switching sequence*, and $\Phi : \mathcal{I}_s^k \mapsto \mathcal{X} \times \mathcal{U}^k$, $\Psi : \mathcal{I}_s^k \mapsto \mathcal{X}$ are implicitly defined by (4.1)[1].

The set of points that are reachable from any $x(0) \in \mathcal{X}(0)$ with $u(t) \in \mathcal{U}, 0 \leq t < k$ by following a given switching sequence $I(k)$ is:

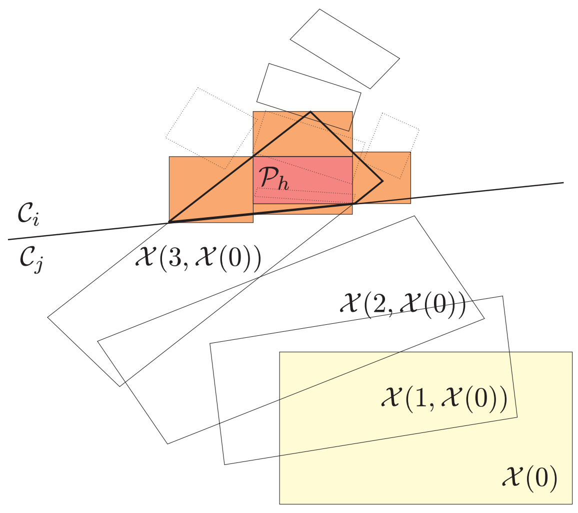
$$\text{Reach}_k(\mathcal{X}(0),\mathcal{U}) \mid_{I(k)} = \left\{ x : \begin{array}{l} x = \Phi_{I(k)} \begin{bmatrix} x(0) \\ u(0) \\ \vdots \\ u(k-1) \end{bmatrix} + \psi_{I(k)}, \\ x(0) \in \mathcal{X}(0), u(t) \in \mathcal{U}, \\ H_{i(t)} x(t) + J_{i(t)} u(t) \leq K_{i(t)}, \\ \tilde{H}_{i(t)} x(t) + \tilde{J}_{i(t)} u(t) < \tilde{K}_{i(t)}, \\ 0 \leq t < k \end{array} \right\}, \tag{4.2}$$

where $x(t)$ is given by Equation (4.1) using the first t components of $I(k)$.

Let $\mathcal{I}(k)$ be the collection of all the switching sequences of length k, then

$$\text{Reach}_k(\mathcal{X}(0),\mathcal{U}) = \cup_{I(k)\in\mathcal{I}(k)} \text{Reach}_k(\mathcal{X}(0),\mathcal{U}) \mid_{I(k)} . \tag{4.3}$$

[1]In case $\mathcal{X}, \mathcal{U}$ are not vector spaces, a well posedness assumption as in Definition 2 implies that $\Phi_{I(k)} [\cdot] + \psi_{I(k)} \in \mathcal{X}$.



$\mathcal{P}_h$
$\mathcal{C}_i$
$\mathcal{C}_j$
$\mathcal{X}(3,\mathcal{X}(0))$
$\mathcal{X}(2,\mathcal{X}(0))$
$\mathcal{X}(1,\mathcal{X}(0))$
$\mathcal{X}(0)$

Figure 4.1: Reach-set evolution, guardline crossing, outer approximation of a new intersection

In general the set $\mathrm{Reach}_k(\mathcal{X}(0),\mathcal{U})$ is not convex and disconnected. We remark also that the number of all switching sequences of length k is combinatorial with respect to k and makes an explicit enumeration approach impractical. A switching sequence $I(k)$ is *feasible* for the initial set $\mathcal{X}(0)$ and the input set $\mathcal{U}$ if the corresponding set $\mathrm{Reach}_k(\mathcal{X}(0),\mathcal{U})\mid_{I(k)}$ is nonempty. Clearly only the feasible switching sequences contribute to determining the set $\mathrm{Reach}_k(\mathcal{X}(0),\mathcal{U})$.

In the next section we will present a forward reachability analysis algorithm capable of enumerating all the feasible switching sequences and therefore to determine $\mathrm{Reach}_k(\mathcal{X}(0),\mathcal{U})$ using Equations (4.2) and (4.3).

4.2.2 Determination of the Reachable Set

In order to determine admissible switching sequences $I(T)$, we can exploit the structure of the PWA system (2.1) to easily compute the forward reach set as long as the evolution remains within a single region $\mathcal{X}_i$ of the polyhedral partition. Whenever the reach set crosses a guardline and enters a new region $\mathcal{X}_j$, a new reach-set computation based on the j-th linear dynamics is computed, as shown in Figure 4.1. The procedure is summarized by the following Algorithm 4.2.1.
In Algorithm 4.2.1 we assume that $\mathcal{X}(0)$ and $\mathcal{U}$ are convex polytopes and STACK is a stack for which appropriate pop(), push() and isEmpty() operations are defined. If $\mathcal{X}(0)$ is a non convex polytope given as the union of convex polytopes $\mathcal{X}(0) = \cup_i \mathcal{X}_i(0)$, then it is enough to push each polytope $\mathcal{X}_i(0) \times \mathcal{U}$ in the STACK in Step 1.. The algorithm determines the set $\mathrm{Reachable}_{K_{\max}}$ as union of sets $\mathrm{Reach}_k(\mathcal{X}(0),\mathcal{U})\mid_{I(k)}, I(k) \in \mathcal{I}(k), k \leq K_{\max}$. Each while loop in Step 2.3. is referred to as *exploration*. The function terminate() in Step 2.3. contains the following termination conditions for the exploration:

F1 The evolution left the current region ($\mathcal{X}\mathcal{U}(t) \cap \mathcal{P}_i = \emptyset$),

F2 The exploration horizon is over $k > K_{\max}$.

Reach-Set Representation

The high level description of Algorithm 4.2.1 is conceptually similar to other existing forward reachability algorithms. What makes each algorithm specific is the way the reach set is stored and updated.

```
function $\mathrm{Reachable}_{K_{\max}}$ = reachset($\Sigma_{\mathrm{PWA}}, \mathcal{X}(0), \mathcal{U}, K_{\max}$)
1. STACK.push($(\mathcal{X}(0) \times \mathcal{U})(0)$);
2. while not STACK.isEmpty(),
  2.1. $\mathcal{XU}(k)$ = STACK.pop();
  2.2. let $i$ such that $\mathcal{XU}(k) \subseteq \mathcal{P}_i$;
  2.3. while not terminate($\mathcal{XU}(k)$,$K_{\max}$),
      2.3.1. $\mathrm{Reachable}_{K_{\max}} = \mathrm{Reachable}_{K_{\max}} \bigcup \mathcal{XU}(k)$;
      2.3.2. if $(\mathcal{XU}(k)) \subseteq \mathcal{P}_i$,
          2.3.2.1. $k \leftarrow k+1$;
          2.3.2.2. $\mathcal{X}(k) \leftarrow [A_i\ B_i]\mathcal{XU}(k-1) + f_i$;
          2.3.2.3. $\mathcal{XU}(k) \leftarrow \mathcal{X}(k) \times \mathcal{U}$;
      2.3.3. else
          2.3.3.1. for all $h \neq i$ such that $\mathcal{XU}_h \triangleq \mathcal{P}_h \bigcap \mathcal{XU}(k) \neq \emptyset$,
              2.3.3.1.1. STACK.push($\mathcal{XU}_h$);
          2.3.3.2. $\mathcal{XU}(k) \leftarrow (\mathcal{X}(k) \times \mathcal{U}) \bigcap \mathcal{P}_i$;
3. return $\mathrm{Reachable}_{K_{\max}}$.
```

Algorithm 4.2.1: Computation of the reach set

The formal simplicity of timed automata implies that the reach set can be stored as zones. Zones are simple polytopes for which efficient manipulation tools exist [7, 66]. The main problem when dealing with timed automata is to capture complex dynamics into the limited model capability $\dot{x} = c$. Nonetheless verification of timed automata encountered a remarkable success especially because of the efficient tools capable of checking a timed automata against a specification [38, 140], moreover the same tools allow to perform control [11]. On the other hand, more complex hybrid models usually lead to more complex reach set representations (either exact or approximated by some simpler shapes). HyperTech [89] represents the reach set as the union of hyper-rectangles. It provides guarantees on the overapproximation as it uses interval arithmetics. ChechMate [56, 57] uses convex polytopes to track the evolution of a set subject to a nonlinear dynamics. However the overapproximation property may not be guaranteed if the dynamics are nonlinear. The tool d/dt [6, 60] represents an over-approximation of the reachable set as *orthogonal polyhedra* while similarly to CheckMate uses convex polyhedra to compute the one step reach set. VeriSHIFT [43] uses a different approach and profits from the compact representation of ellipsoids [99].

In this chapter the reachable set ($\mathrm{Reachable}_{K_{\max}}$) and the reach set ($\mathrm{Reach}_k$) are represented as the union of polytopes, and within a single exploration the reach set is kept as a single polytope. In order to implement the algorithm we need four key operations: (i) the computation of the Minkowski's sum in Step 2.3.2.1., (ii) the intersection of polyhedra in steps 2.3.3.2., 2.3.3.1. and for the termination condition **F1** of Step 2.3., (iii) a set containment test in Step 2.3.2. and (iv) an emptiness test. In addition, in order to contain the time and space requirements, it is desirable to minimize the complexity of the operations mentioned above and to have a compact reach set representation.

The explicit computation of the Minkowski's sum is not attractive as both H- and V-representation have computational drawbacks. The explicit computation of an H-representation of the Minkowski's sum requires the projection of the polytope, while the computation of a V-representation requires repeated computations to remove the redundant vertices, moreover the number of non-redundant vertices of a V-representation has an upper bound exponential in the dimension of the polytope, that is achieved when the summand polytopes are zonotopes [84, Theorem 2.1.10]. A zonotope [74] is the projection of an n-dimensional cube on a d-dimensional linear subspace, examples of Zonotopes are hyper-rectangles. Therefore, if $\mathcal{X}(0)$ and $\mathcal{U}$ are hyper-rectangles, then the reach set for a given switching sequence ($\text{Reach}_k(\mathcal{X}(0),\mathcal{U}) \mid_{I(k)}$) is a zonotope by Equation (4.1). Apart form that, computing the intersection of polytopes in V-representation is the dual problem of the extended convex hull problem, which is computationally involving [77].

Each polyhedron is stored as a weighted Minkowski's sum (we remark once more that the weighted Minkowski's sum is an implicit representation where we store the summand and the weights). It is attractive because it allows to save the operations needed for computing the explicit representation and it is compact. Such a representation addresses the first computational request described above.

Example 4.2.1 Consider the reach set of a 4-dimensional linear system with 2 inputs on $K_{\max} = 10$ steps. Assume that $\mathcal{X}(0)$ and $\mathcal{U}$ are hyper-rectangles (i.e.: $m_i \le x_i \le M_i$, $i = \{1,\ldots,4\}$ and $n_j \le u_j \le N_j$, $j = \{1,2\}$) then the number of constraints of $\mathcal{X}(0)$ and $\mathcal{U}$ is respectively $n_x = 8$ and $n_u = 4$. The reach set after 10 steps has 4096 vertices, while the implicit representation would need $n_x + K_{\max}n_u = 48$ inequalities and Equation (4.1).

□

The following Proposition 1 addresses the second computational requirement.

Proposition 1 *The intersection of a weighted Minowski's sum $R_1\mathcal{P}_1 + R_2\mathcal{P}_2 + \ldots + R_n\mathcal{P}_n$, $\mathcal{P}_i = \{x : C^i x \le D^i\}$ with a polytope $\mathcal{Q} = \{x : Ax \le B\}$ is a weighted Minkowski's sum $S\mathcal{T}$ where*

$$S = \begin{bmatrix} R_1 & R_2 & \ldots & R_n \end{bmatrix}, \tag{4.4}$$

and

$$\mathcal{T} = \left\{ y : \begin{bmatrix} C^1 & 0 & \cdots & 0 \\ 0 & C^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C^n \\ AR_1 & AR_2 & \cdots & AR_n \end{bmatrix} y \le \begin{bmatrix} D^1 \\ D^2 \\ \vdots \\ D^n \\ B \end{bmatrix} \right\}. \tag{4.5}$$

Proof. The set $S\mathcal{T}$ is a weighted Minkowski's sum of one polytope. The first n block of rows of constraints of $\mathcal{T}$ and S are an other way of writing $\sum_{i=1}^n R_i\mathcal{P}_i$ with $y = [x_1' \ \ldots \ x_n']'$, while the last block of constraints of $\mathcal{T}$ together with S define the set $\mathcal{Q}$, to see this is enough to substitute $x = Sy$ in the inequalities $Ax \le B$ to obtain the last block of constraints of $\mathcal{T}$. □

Note that, in general, the H-representation of the set $\mathcal{T}$ may contain redundant constraints. However Corollary 1 and Remark 4.1.3 provide a computational tool to solve the containment problem and as a byproduct provide the constraints of $\mathcal{Q}$ that are violated by points of the Minkowski's sum, therefore the intersection of Step 2.3.3.2. contains only non-redundant constraints of $\mathcal{Q}$.

Finally feasibility of a weighted Minkowski's sum is simply addressed by Remark 4.1.2. We recall here that the feasibility of a H-polytope can be checked by solving a linear program (LP).

Remark 4.2.1 The containment test in Step 2.3.2. provides information on the constraints of $\mathcal{P}_i$ that are violated. This information can be exploited in the following Step 2.3.3.1. to save computations and check the intersection only for the regions $\mathcal{P}_h$ that share at least one violated constraint with $\mathcal{P}_i$.

□

Tree of Evolution

The result of the exploration algorithm detailed in the previous sections is conveniently stored in a tree G (see Figure 4.2). The nodes of the tree represent sets from which a

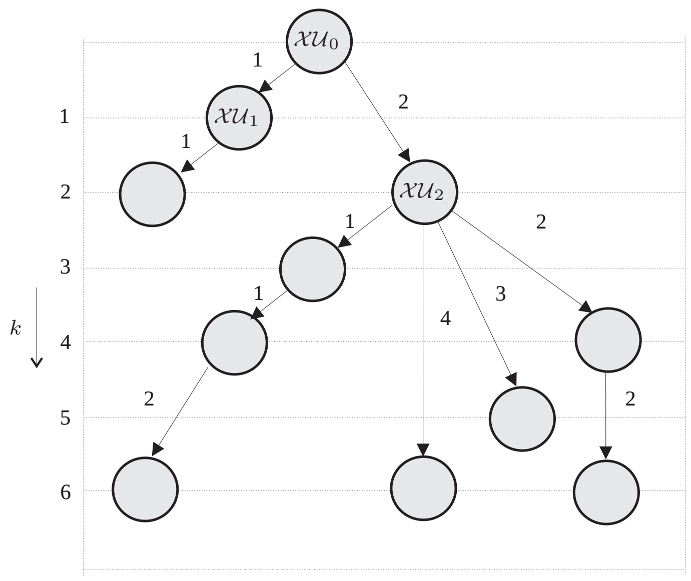


Figure 4.2: Tree of evolution

reach set evolution is computed, and a branch connects two nodes if a transition exists between the two corresponding sets, starting at time k_1 and ending at time k_2, with $i(k) \equiv h$, $\forall k = k_1, \ldots, k_2$, where $h \in \mathcal{I}_s$ is a given mode, associated with the starting node. Each branch b has an associated weight $k_b = k - 2 - k_1$ which represents the number of steps needed for the transition. The root node of the tree is the initial set[2] $\mathcal{XU}_0 = \mathcal{X}(0) \times \mathcal{U}$, from which the reach set evolution is computed. When a new set $\mathcal{XU}_h$ crossing a guard-line is detected, it becomes a new node. The new node is connected by a weighted branch from $\mathcal{XU}_0$, and inserted in a list of sets to be subsequently explored. Then, computation of the reach set proceeds in each region $\mathcal{P}_h$ from each new intersection $\mathcal{XU}_h$. The oriented paths on G from the initial node $\mathcal{XU}_0$ determine all the feasible switching sequences $\mathcal{I}(K_{\max}) \in \mathcal{I}_s^{K_{\max}}$.

4.3 Approximate Reachability Analysis

In this section we present tools for determining over-approximations of the reachable set that have a reduced complexity, in particular that have an associated tree of evolution with less nodes. Over-approximations have the consequence of making the analysis conservative when used for instance in the context of verification of safety properties, although, as we will show, this conservativeness can be eventually removed in a simple post-processing operation.

4.3.1 Approximation of Intersections

The computation of the reach set proceeds in each region $\mathcal{P}_h$ from each new intersection $\mathcal{XU}_h$. Before starting a new reach set computation from a set $\mathcal{XU}_j$ extracted from STACK, we check for inclusion of $\mathcal{XU}_j$ in other nodes of G. If this happens, say $\mathcal{XU}_j \subseteq \mathcal{XU}_1$ as in Figure 4.3, the node associated with $\mathcal{XU}_j$ is connected to $\mathcal{XU}_1$ with an arc of weight 0.[3]

[2]With a slight abuse of notation, $\mathcal{XU}_k$ will be referred to as both a node of the tree and the associated set in the state space.

[3]If we are interested in an inner approximation of the reach set then it is enough to direct the new arc in the other direction.

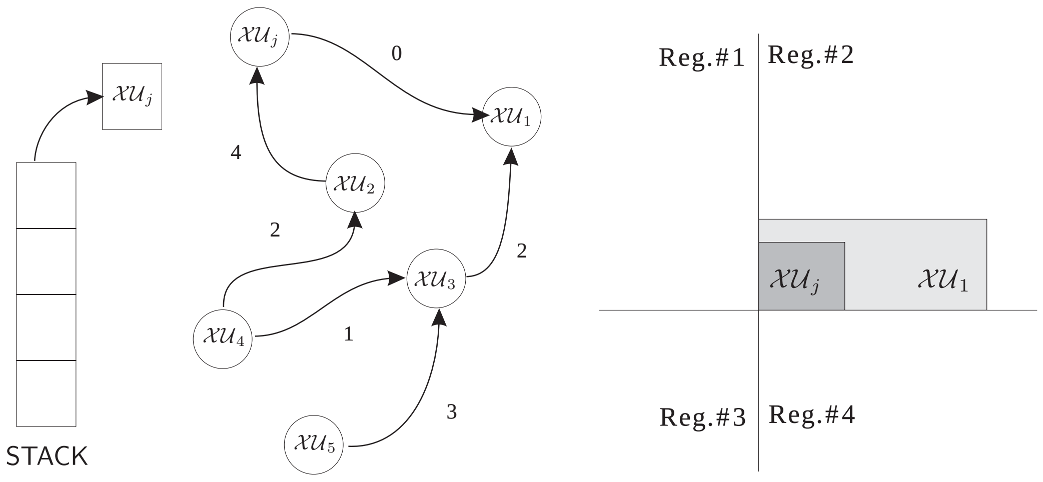


Figure 4.3: Adding and removing nodes to the graph G

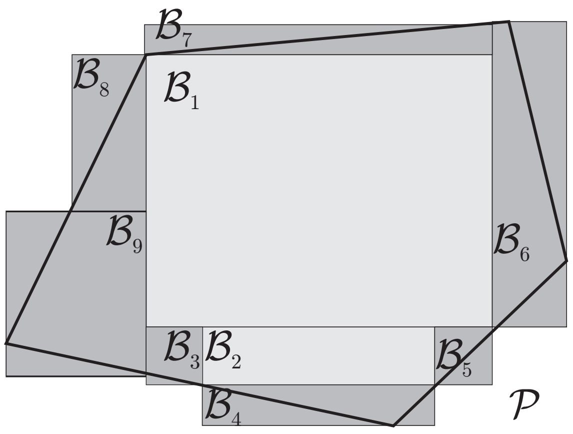


Figure 4.4: Inner ($\mathcal{B}_1 \cup \mathcal{B}_2$) and outer ($\cup_{i=1}^{9} \mathcal{B}_i$) approximation of a polytope $\mathcal{P}$

Clearly, if we perform this check we save explorations but the tree becomes an oriented graph.

In principle the number of constraints defining $\mathcal{XU}_h$ grows linearly with time, we therefore need to approximate $\mathcal{XU}_h$ so that its complexity is bounded (and therefore reach set computation from $\mathcal{XU}_h$ has a limited complexity with respect to the initial region), and checking for set inclusion is a simple task. Hyper-rectangular approximations are the best candidates, as set inclusion between hyper-rectangles reduces to a simple comparison of the coordinates of the vertices. On the other hand, a crude rectangular outer approximation of $\mathcal{XU}_h$ can lead to explore large regions which are not reachable from the initial set $\mathcal{XU}(0)$, as they are just introduced by the approximation itself. In [19] the authors propose an iterative method for inner and outer approximation which is based on linear programming, and approximates with arbitrary precision polytopes by a collection of hyper-rectangles, as depicted in Figure 4.4[4].

Again all the switching sequences are contained in the oriented path coming out from node $\mathcal{XU}_0$ in graph G. However, due to the hyper-rectangular outer approximation of new intersections $\mathcal{XU}_h$ and to the inclusion of nodes with small initial set $\mathcal{XU}_j$ in larger nodes, not all the paths in G correspond to feasible switching sequences. Nevertheless,

[4] The inner approximation is useful when we are interested in an inner approximation of the $\text{Reachable}_{K_{\max}}$ set, see also previous footnote 3.

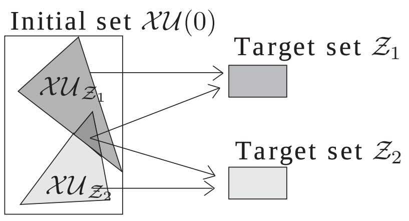

Initial set $\mathcal{XU}(0)$
Target set $\mathcal{Z}_1$
$\mathcal{XU}_{\mathcal{Z}_1}$
Target set $\mathcal{Z}_2$
$\mathcal{XU}_{\mathcal{Z}_2}$

Figure 4.5: Verification problem

feasibility of each switching sequence can be simply tested via linear programming[5].

4.4 Verification of Safety Properties

The main purpose of reachability analysis is to provide the certification that a given hybrid dynamical system enjoys certain safety properties, or provide a counterexample. A safety property can be stated as follows: for a given set of initial conditions and exogenous signals, the set of unsafe states is never entered. We formalize the problem of *verification* of safety properties as follows:

Problem 1 (Verification Problem) *Given a hybrid system Σ in PWA form (2.1)[6], a set of initial conditions $\mathcal{X}(0)$, a collection of disjoint target sets $\mathcal{Z}_1$, $\mathcal{Z}_2$, ..., $\mathcal{Z}_L$, a bounded set of inputs $\mathcal{U}$, and a time interval $\{0,\ldots,K_{\max}\}$, determine* (i) *if $\mathcal{Z}_j$ is reachable from $\mathcal{X}(0)$ within $k \leq K_{\max}$ steps for some sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$ of exogenous inputs;* (ii) *if yes, the subset of initial conditions $\mathcal{X}_{\mathcal{Z}_j}(0)$ of $\mathcal{X}(0)$ from which $\mathcal{Z}_j$ can be reached within $K_{\max}$ steps;* (iii) *for any $x_1 \in \mathcal{X}_{\mathcal{Z}_j}(0)$ and $x_2 \in \mathcal{Z}_j$, (one of) the input sequence(s) $\{u(0), \ldots, u(k-1)\} \subseteq \mathcal{U}$, $k \leq K_{\max}$, that drives x_1 to x_2.*

From now on, we will assume that $\mathcal{Z}_j$, are polyhedral sets, and, without loss of generality, that they are also convex. The reason for focusing on finite-time reachability is that states which are not reachable in less than $K_{\max}$ steps are in practice, if $K_{\max}$ is large enough, unreachable. Although finite time reachability analysis cannot guarantee certain liveness properties (for instance, if $\mathcal{Z}_i$ will be ever reached), the reachability problem stated above is clearly decidable. Nevertheless, the problem is $\mathcal{N}P$-hard [37].

The problem formulated above is very similar to the one stated in [67], where the authors use big-M techniques to transform the problem into a mixed integer linear problem (MILP). While here we exploit the reach set computation techniques of Section 4.2, in [67] the solution approach is based on the solution to a (large) MILP that summarizes the system dynamics over the time $K_{\max}$ and the target regions configuration. The cost function aims at maximizing the permanence in the unsafe regions and therefore finds the less safe trajectory. Although good MILP solvers have a satisfactory performance for large problems, the problem might become intractable already for small $K_{\max}$.

The previous questions of verification can be answered once all the switching sequences $\mathcal{I}(k)$, $\forall k \leq K_{\max}$ leading to $\mathcal{Z}_1$, or $\mathcal{Z}_2$, ..., or $\mathcal{Z}_L$ from $\mathcal{XU}(0) = \mathcal{X}(0) \times \mathcal{U}$ are determined using Algorithm 4.2.1. In fact, it is enough to check that the reach set at time k satisfies $\mathrm{Reach}_k(\mathcal{X},\mathcal{U}) \cap \mathcal{Z}_j \neq \emptyset$. Nonetheless Algorithm 4.2.1 can be improved by exploiting the results of Section 4.3, as the following Algorithm 4.4.1 shows.

Steps 1.1., 1.2., 1.2.1., 1.2.4.1., 1.2.4.1.2.–1.2.4.1.3.1., and 1.2.4.1.3.2. are directly inherited by the generic Algorithm 4.2.1. At Step 1., the set of initial states is considered as an union of convex sets, $\mathcal{X}(0) = \cup_{i=1}^{n_o} \mathcal{X}_i$. GRAPH is the graph G summarizing

[5]In the case of inner approximation of the $\mathrm{Reachable}_{K_{\max}}$ set all the switching sequences in the graph G are feasible, therefore there is no need for testing the feasibility of the switching sequences.

[6]If the system is not directly given in PWA form, but in MLD or DHA form, an equivalent PWA form can be always obtained by exploiting the equivalence properties recalled in Chapter 2.

```
function GRAPH=Verification ($\Sigma_{\mathrm{PWA}}$, $\mathcal{X}(0)$, $\mathcal{U}$, $K_{\max}$)
1. GRAPH.init($\{(\mathcal{X}_i(0) \times \mathcal{U})(0)\}_{i=1}^{n_0}$);
  1.1. STACK.push($(\mathcal{X}_i(0) \times \mathcal{U})(0)$), $i = 1, \dots, n_0$;
  1.2. while not STACK.isEmpty(),
        1.2.1. $\mathcal{XU}_f = \mathcal{XU}(k)$ = STACK.pop(), let $i$ such that $\mathcal{XU}(k) \subseteq \mathcal{P}_i$;
        1.2.2. $k \leftarrow k^* \triangleq$ minimum arrival time from initial nodes to $\mathcal{XU}(k)$;
        1.2.3. if $\mathcal{XU}(k) \subseteq \mathcal{XU}_j$, $\mathcal{XU}_j \in$ GRAPH,
                 GRAPH.connect($\mathcal{XU}_j$,$\mathcal{XU}_f$,0), next();
        1.2.4. if relaxedSafe($\mathcal{XU}(k)$,$\{\mathcal{Z}_j\}_{j=1}^{n_f}$), next();
               1.2.4.1. while not terminate($\mathcal{XU}(k)$,$K_{\max}$),
                     1.2.4.1.1. for $j = 1, \dots, n_f$,
                        1.2.4.1.1.1. if $\mathcal{XU}(k) \bigcap (\mathcal{Z}_j \times \mathcal{U}) \neq \emptyset$,
                           GRAPH.connect($\mathcal{Z}_j$,$\mathcal{XU}_f$,$k - k^*$);
                     1.2.4.1.2. if $(\mathcal{XU}(k)) \subseteq \mathcal{P}_i$,
                        1.2.4.1.2.1. $k \leftarrow k + 1$;$\mathcal{X}(k) \leftarrow [A_i\ B_i]\mathcal{XU}(k-1) + f_i$;
                        1.2.4.1.2.2. $\mathcal{XU}(k) = \mathcal{X}(k) \times \mathcal{U}$;
                     1.2.4.1.3. else
                        1.2.4.1.3.1. for all $h \neq i$ such that $\mathcal{XU}_h \triangleq \mathcal{P}_h \bigcap \mathcal{XU}(k) \neq \emptyset$,
                           STACK.push($\lceil \mathcal{XU}_h \rceil$);
                           GRAPH.connect($\lceil \mathcal{XU}_h \rceil$,$\mathcal{XU}_f$,$k = k^*$);
                        1.2.4.1.3.2. $\mathcal{XU}(k) \leftarrow (\mathcal{X}(k) \times \mathcal{U}) \bigcap \mathcal{P}_i$;
2. return GRAPH.
```

Algorithm 4.4.1: Verification Algorithm

the evolution according to Section 4.3.1, the function init() initializes the graph with the initial sets, the function connect(nodeTo,nodeFrom,arcWeight) connects two nodes and creates the node nodeTo if it does not exist. The function relaxedSafe() in Step 1.2.4. checks if the convex over approximation $\mathcal{C}_k([I\ 0]\mathcal{XU}(k))$ of $\mathrm{Reach}_k([I\ 0]\mathcal{XU}(k))$ is safe, if so the exact evolution is also safe and there is no need to compute the exact reach set. As $\mathcal{C}_k([I\ 0]\mathcal{XU}(k))$ is a weighted Minkowski's sum, the safety check is formally equivalent to the case of exact reach set and is therefore omitted. The function next() exits from the current exploration and jumps to Step 1.2.1.. In Steps 1.2.4.1.3.1., $\lceil \mathcal{XU}_h \rceil$ stands for the over approximation of $\mathcal{XU}_h$ obtained using the approach of [19]. Finally, Step 1.2.4.1. contains the termination conditions which include the conditions **F1** and **F2** defined in Section 4.2.2 and the following conditions:

F3 The reach set entered target set $\mathcal{XU}(k) \subseteq \mathcal{Z}_i$,

F4 The reach set will never exit region $\mathcal{P}_i$.

We already discussed how containment conditions can be checked, therefore condition **F3** is easy to check. Condition **F4** deserves some more explanation, as in the given formulation may seem hard to test. In [98], the authors present the computation of the maximal robust positively invariant set O_∞ for a discrete-time linear system with state constraints $x \in \mathcal{P}_i$, subject to additive polyhedral bounded disturbances $u \in \mathcal{U}$. In other words O_∞ is the largest set such that $[x(k) \in O_\infty] \Rightarrow [x(k+t) \in O_\infty, \forall u \in \mathcal{U}, \forall t > 0]$. Clearly if we compute the set O_∞^i for each region $\mathcal{P}_i$ of the PWA system, then termination condition **F4** reduces to the already discussed set containment condition.

With minor modifications Algorithm 4.4.1 is capable of reporting the first sequence that enters any target set as soon as it is identified, to refine an already built graph by increasing the maximum number of steps $K_{\max}$, and to remove the conservatism due to the over approximations introduced by adopting the techniques discussed in Section 4.3.1.

4.4.1 Applications

A typical application of Algorithm 4.4.1 is the verification of safety properties of an embedded system, namely a digital controllers that operates on a continuous system. In [32] a preliminary version of this algorithm was used to asses the maximum excursion of the height of an electronic car suspension system, and the computational times were comparable to the those achieved by using symbolic approaches. In [131] the authors verified if a cruise control is able to reach the speed set point within a maximum time without overshooting more than the tolerance enforced by local authorities. The algorithm was also applied in [37] to the batch evaporator process benchmark, where the aim is to asses if the controller is capable of shutting down a chemical plant in a safe way in case of equipment failures.

Verification can be used also to assess system theoretic properties, such as stability. In [35], we formulated the problem of characterizing the stability of a piecewise affine (PWA) system as a reachability analysis problem. The basic idea is to take the whole $\mathbb{R}^n$ as the set of initial conditions, and check that all the trajectories go to the origin. More precisely, it is possible to recast the stability problem as a reachability analysis problem by restricting the set of initial conditions to an (arbitrarily large) bounded set $\mathcal{X}(0)$, and label as "asymptotically stable in T steps" the trajectories that enter an invariant set $\mathcal{Z}_1$ around the origin within a finite time T, or as "unstable in T steps" the trajectories which enter a set $\mathcal{Z}_2$ of (very large) states. Subsets of $\mathcal{X}(0)$ leading to none of the two previous cases are labeled as "non-classifiable in T steps". The domain of asymptotical stability in T steps is a subset of the domain of attraction of an equilibrium point, and has the practical meaning of collecting the initial conditions from which the settling time to a specified set around the origin is smaller than T. In addition, it can be computed algorithmically in finite time using Algorithm 4.4.1. The same idea was used to assess the stability and the performance of PWA (i.e.: model predictive) controllers in [36], and is reviewed in the next section.

4.5 Performance Assessment of Hybrid MPC

The closed-loop constituted by a hybrid MLD/PWA system and the explicit MPC controller (3.12) is of the form (2.1) [28]. Note that the form of the closed-loop MPC system remains PWA also when the plant model and the prediction model are different hybrid models. This is for instance the case when the MPC law is designed based on a linear model obtained by linearizing the nonlinear model of the plant around some operating condition. When the nonlinear model can be approximated by a PWA system (e.g., through multiple linearizations at different operating points or by approximating nonlinear static mappings by piecewise linear functions), the closed-loop consisting of the nonlinear plant model and the MPC controller can be approximated by a PWA system as well.

In this section we discuss the application of the reachability analysis algorithm developed in Section 4.4 for performance analysis of MPC in some detail.

For MPC applications, it is interesting to estimate the domain of attraction of the equilibrium state, and the set of initial conditions from which the state trajectory remains feasible for the constraints. As mentioned in the previous section, the nominal MPC closed-loop system is an autonomous PWA system. The origin typically belongs to the interior of one of the sets of the partition, for instance a region where a linear controller is asymptotically stabilizing while fulfilling the constraints, which by convention will be referred to as $\mathcal{X}_0$. Denote by $\mathcal{D}_\infty(0) \subseteq \mathbb{R}^n$ the (unknown) domain of attraction of the origin. Given a bounded set $\mathcal{X}(0)$ of initial conditions, we want to characterize $\mathcal{D}_\infty(0) \bigcap \mathcal{X}(0)$.

By construction, the matrix A_0, associated with the region $\mathcal{X}_0$, is strictly Hurwitz and $f_0 = 0$ (in fact, in $\mathcal{X}_0$ the feedback gain is the unconstrained LQR gain $F_0 = K$, $g_0 = 0$ [33]).

Then we can compute an invariant set in $\mathcal{X}_0$. In particular, we compute the *maximum output admissible set* (MOAS) $\mathcal{X}_\infty \subseteq \mathcal{X}_0$. $\mathcal{X}_\infty$ is the largest invariant set contained in $\mathcal{X}_0$, which by construction of $\mathcal{X}_0$ is compatible with the constraints $u_{\min} \leq Kx(t) \leq u_{\max}$, $x_{\min} \leq x(t) \leq x_{\max}$. By [80, Th.4.1], MOAS is a polyhedron with a finite number of facets, and is computed through a finite number of linear programs (LP's) [80].

In order to circumvent the undecidability of stability, we give the following

Definition 5 *Consider the PWA system (2.1), and let the origin* $0 \in \mathring{\mathcal{X}}_0 \triangleq \{x : \; H_0 x < S_0\}$, *and* A_0 *be strictly Hurwitz. Let* $\mathcal{X}_\infty$ *be the maximum output admissible set (MOAS) in* $\mathcal{X}_0$, *which is an invariant for the linear system* $x(t+1) = A_0 x(t)$. *Let* T *be a finite time horizon. Then, the set* $\mathcal{X}(0) \subseteq \mathbb{R}^n$ *of initial conditions is said to belong to the* domain of attraction in T steps $\mathcal{D}_T(0)$ *of the origin if* $\forall x(0) \in \mathcal{X}(0)$ *the corresponding final state* $x(T) \in \mathcal{X}_\infty$.

Note that $\mathcal{D}_T(0) \subseteq \mathcal{D}_{T+1}(0) \subseteq \mathcal{D}_\infty(0)$, and $\mathcal{D}_T(0) \to \mathcal{D}_\infty(0)$ as $T \to \infty$. The horizon T is a practical information about the speed of convergence of the PWA system to the origin and thus about its dynamic performance.

Definition 6 *Consider the PWA system (2.1), and let* $\mathcal{X}_{\text{infeas}} \triangleq \mathbb{R}^n \setminus \cup_{i=1}^s \mathcal{X}_i$. *The set* $\mathcal{X}(0) \subseteq \mathbb{R}^n$ *of initial conditions is said to belong to the* domain of infeasibility in T steps $\mathcal{I}_T(0)$ *if* $\forall x(0) \in \mathcal{X}(0)$ *there exists* t, $0 \leq t \leq T$ *such that* $x(t) \in \mathcal{X}_{\text{infeas}}$.

In Definition (6), the set $\mathcal{X}_{\text{inf}}$ must be interpreted as a set of "very large" states. Although instability in T steps does not guarantee instability (for any finite T, a trajectory might reach $\mathcal{X}_{\text{inf}}$ and converge back to the origin), it has the practical meaning of labeling as "unstable" the trajectories whose magnitude is unacceptable, for instance because the PWA system is no longer valid as a model of the real system.

Given a set of initial conditions $\mathcal{X}(0)$, we aim at finding subsets of $\mathcal{X}(0)$ which are safely asymptotically stable ($\mathcal{X}(0) \bigcap \mathcal{D}_T(0)$), and subsets which lead to infeasibility in T steps ($\mathcal{X}(0) \bigcap \mathcal{I}_T(0)$). Subsets of $\mathcal{X}(0)$ leading to none of the two previous cases are labeled as *non-classifiable in* T *steps* As we will use linear optimization tools, we assume that $\mathcal{X}(0)$ is a convex polyhedral set (or the union of convex polyhedral sets). Typically, non-classifiable subsets shrink and eventually disappear for increasing T.

Example 4.5.1 Consider the system $y(t) = \frac{s+1}{s^2+s+2} u(t)$, sample the dynamics with $T = 0.2$ s, and obtain the state-space representation

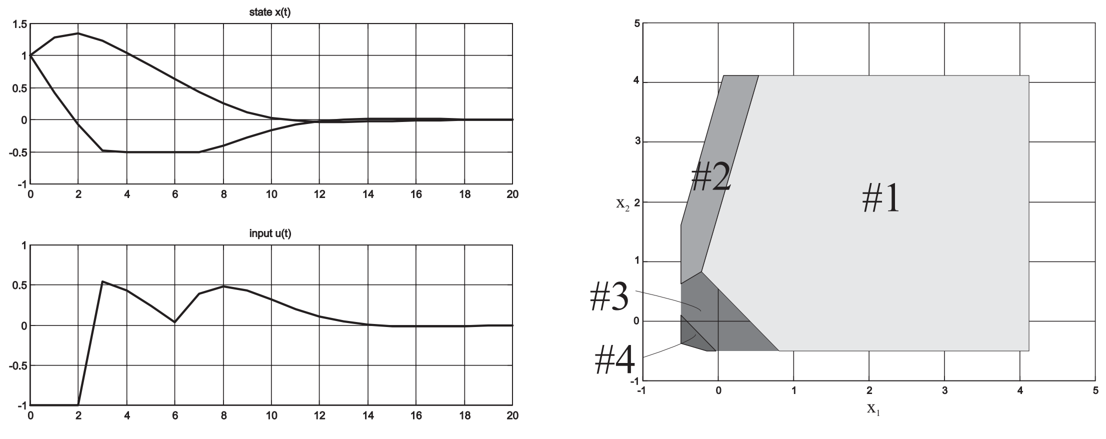
$$\begin{cases} x(t+1) = \begin{bmatrix} 0.7839 & -0.1788 \\ 0.3577 & 0.9627 \end{bmatrix} x(t) + \begin{bmatrix} 0.1788 \\ 0.0372 \end{bmatrix} u(t) \\ y(t) = \begin{bmatrix} 1 & 0.5 \end{bmatrix} x(t) \end{cases} \tag{4.6}$$

The task is to regulate the system to the origin while fulfilling the constraints $-1 \leq u(t) \leq 1$ and $x(t) \geq \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}$. To this aim, we design a linear MPC controller based on the optimization problem

$$\begin{array}{ll} \min\limits_{u_t, u_{t+1}} & ||x_{t+2|t}||_P^2 + \sum\limits_{k=0}^{1} ||x_{t+k|t}||^2 + .1||u_{t+k}||^2 \\ \text{s.t.} & -2 \leq u_{t+k} \leq 2, \; k = 0, 1 \\ & x_{t+k|t} \geq x_{\min}, \; x_{\min} \triangleq \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix}, \; k = 0, 1 \end{array} \tag{4.7}$$

where P is the solution to the Riccati equation (in this example $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 0.1$, $N_u = N_y = N_c = 2$). Note that this choice of P corresponds to setting $u_{t+k} = Kx_{t+k|t}$ for $k \geq 2$, where K is the LQR gain, and minimizes $\sum_{k=0}^{\infty} x'_{t+k|t} x_{t+k|t} + .01 u_{t+k}^2$ with respect to u_t, u_{t+1}. The closed loop response from the initial condition $x(0) = [1 \; 1]'$ is shown in Fig. 4.6(a).

The solution to the mp-QP problem was computed by using the solver in [33] in 0.66 s on a PC Pentium III 650 MHz running Matlab 5.3, and the corresponding polyhedral



(a) Closed-loop MPC (b) Explicit solution

Figure 4.6: Example (4.7)

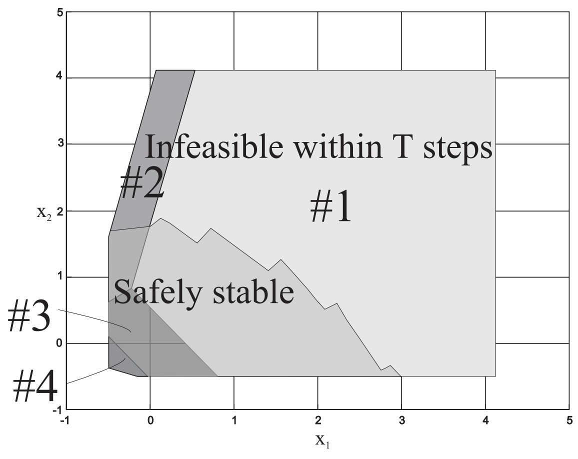


Figure 4.7: Partition of initial states into safely stable, and infeasible in $T = 20$ steps

partition of the state-space is depicted in Fig. 4.6(b). The MPC law is linear in each one of the four depicted regions.

Region #3 corresponds to the unconstrained LQR controller, #1 and #4 to saturation at -1 and $+1$, respectively, and #2 is a transition region between LQR control and the saturation.

Note that the union of the regions depicted in Fig. 4.6(b) should not be confused with the region of attraction of the MPC closed-loop. For instance, by starting at $x(0) = [3.5\ 0]'$ (for which a feasible solution exists), the MPC controller runs into infeasibility after $t = 5$ time steps.

The reachability analysis algorithm described above was applied to determine the set of safely stable initial states and states which are infeasible in $T = 20$ steps (Fig. 4.7). The algorithm computes the graph of evolutions in 115 s on a Pentium II 400 running Matlab 5.3.

□

4.6 Optimal Control Solutions Based on Reachability Analysis

Consider the following optimal control problem

$$J_{\text{opt}} = \min_{U_0^{K-1}} \left\{ \|x(K) - x_f\|_P^2 + \sum_{k=0}^{K-1} \|u(k)\|_R^2 + \|x(k) - x_f\|_Q^2 \right\} \tag{4.8a}$$

$$\text{s.t.} \begin{cases} x'(k) = A_{i(k)}x(k) + B_{i(k)}u(k) + f_{i(k)} \text{ for } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{P}_{i(k)} \\ u_{\min} \le u(k) \le u_{\max},\ k = 0, 1, \dots, K-1 \\ x_{\min} \le x(k) \le x_{\max},\ k = 1, \dots, K \\ x(0) = x_0 \\ i(k) \in \{1, \dots, s\} \\ x(K) \in \mathcal{Z}_{\text{fin}} \end{cases} \tag{4.8b}$$

where K is the prediction horizon, P, Q and R are positive definite weighting matrices, $x(k)$ is the state evolved at time k by applying the input sequence $U_0^{k-1} \triangleq \{u(0), \dots, u(k-1)\}$ to (2.1) from the initial state $x(0) = x_0$, $i(k) \in \{1, \dots, s\}$ is the index such that Equations (2.1c)-(2.1c) are satisfied, x_f and $\mathcal{Z}_{\text{fin}}$ are a reference state and a polyhedral final target set, respectively [7], $u_{\min}$, $u_{\max}$ and $x_{\min}$, $x_{\max}$ are hard bounds on inputs and states, respectively. The sets $\mathcal{U} \triangleq \{u : u_{\min} \le u \le u_{\max}\}$ and $\mathcal{Z}_{\text{safe}} \triangleq \{x : x_{\min} \le x \le x_{\max}\}$ will be used in the sequel for compactness of notation[8]. In this section we exploit the ideas of the reachability analysis algorithm proposed in Section 4.2 in order to solve the optimal control problem (4.8).

Remark 4.6.1 For ease of notation, we have supposed that the weighting matrices are constant with respect to i (=space) and k (=time). Nonetheless, the proposed framework can easily handle the case of region-dependent and/or time-varying weights $R_{i(k)}(k)$, $Q_{i(k)}(k)$. This feature is helpful for instance when the index i reflects different plant operation modes. The same extension can be done for the input and state limits $u_{\min}$, $u_{\max}$, $x_{\min}$, $x_{\max}$.

□

Remark 4.6.2 Existing heuristic information about the expected optimal schedule can be easily embedded into (4.8). In fact, when heuristics are given as space/time “landmarks” to be hit, such requirements can be expressed in (4.8) as additional constraints of the form $x(k) \in \mathcal{Z}_{\text{heur}}[k_{\min}, k_{\max}]$, where $\mathcal{Z}_{\text{heur}}[k_{\min}, k_{\max}]$ represents the landmark area in the state-space to be reached at time $k \in [k_{\min}, k_{\max}]$.

□

By taking into account the equivalence between PWA and MLD systems (2.20) mentioned in Chapter 2, the optimization problem (4.8) can be reformulated as the mixed-integer quadratic programming (MIQP) problem

$$J_{\text{opt}} = \min_{U_0^{K-1}} \left\{ \|x(K) - x_f\|_P^2 + \sum_{k=0}^{K-1} \|u(k)\|_R^2 + \|x(k) - x_f\|_Q^2 \right\} \tag{4.9a}$$

$$\text{s.t.} \begin{cases} x'(k) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \\ E_2\delta(k) + E_3z(k) \le E_1u(k) + E_4x(k) + E_5 \\ u_{\min} \le u(k) \le u_{\max},\ k = 0, 1, \dots, K-1 \\ x_{\min} \le x(k) \le x_{\max},\ k = 1, \dots, K \\ x(0) = x_0 \\ \delta(k) \in \{0, 1\}^{r_b} \\ x(K) \in \mathcal{Z}_{\text{fin}} \end{cases} \tag{4.9b}$$

[7] For instance, $\mathcal{Z}_{\text{fin}}$ can be a satisfactory range around the equilibrium x_f

[8] More in general, the sets $\mathcal{U}$ and $\mathcal{Z}_{safe}$ could also be polyhedral

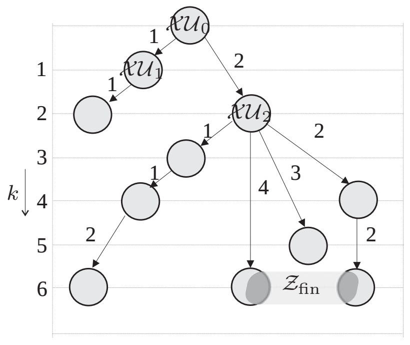


Figure 4.8: Optimal Control Algorithm

This is indeed the approach pursued in [31]. We stress that in the MLD+MIQP formulation (4.9) the binary variables $\delta(k)$ play the role of the index variables $i(k)$ in (4.8).

Practical applications of the above optimal control problems can be developed according to two different philosophies. In some cases, an open-loop solution to (4.8) is sought. This situation resembles a typical scenario for discrete event systems [49], in which optimal input sequences have to be planned. Sometimes a closed-loop implementation is required instead. In this case the proposed optimization strategy can be pursued on-line, combined with a receding horizon philosophy. The result is an optimal control law in feedback form enjoying nice stability properties, provided that suitable terminal state conditions are chosen. In particular, when (4.8)/(4.9) are solved on-line within an MPC scheme, under feasibility assumptions the choice of the terminal state constraint $\mathcal{X}_{\text{fin}} = \{x_f\}$ guarantees stability of the controlled system, which is proven by using standard Lyapunov analysis techniques [31].

Here, we use reachability analysis to determine admissible switching profiles $I_0^{K-1} = \{i(0), \ldots, i(K-1)$. During the computation of reachable sets, set evolutions can be selectively propagated in accordance with the value function J. More precisely, set evolutions having an intermediate cost which is greater than a current upper-bound on J_{opt} are not propagated. Here below, we detail the basic ingredients of the algorithm that together with the computation of the reach sets in Section 4.2 solve the problem. The idea leads to a solver to the optimal control problem which is of branch-and-bound nature, see Figure 4.8. A *branch* occurs whenever a switching is detected, and a *bound* on the optimal cost allows instantaneous termination of entire non-optimal subtrees.

Partial Cost Computation

We note that each node $\mathcal{XU}_h$ of the tree considered in Section 4.2.2 corresponds to a unique switching path from $\mathcal{XU}_0$ to $\mathcal{XU}_h$ itself. The switching path is associated with a switching sequence $I_0^{K_h-1} = \{i_h(0), \ldots, i_h(K_h-1)\}$ of length $K_h = \sum_b K_b$, where K_b are the time intervals associated with the arcs along the switching path. If we were to build up the whole tree as described before, the leaves at distance K from the initial set (those which were terminated by condition **F1** in Section 4.2) and possessing a nonempty intersection with $\mathcal{Z}_{\text{fin}}$ would certainly be feasible candidates for solving the optimal control problem. We could then enumerate all possible winning K-step switching sequences[9] and then compute for each of them the solution J^* to problem (4.8) with I_0^{K-1} fixed, through standard quadratic programming.

On the other hand, we are not interested in the full reachability analysis, so we do not need to build up the whole tree. The idea is to associate a cost J_h to each node $\mathcal{XU}_h$

[9] Selecting only K-step sequences is consistent with the fact that the considered optimal control problem is not a minimum-time one. Note in fact that, for some sequences of inputs, a state-set evolution could possibly reach the final set in a time $k < K$ and leave it afterwards.

by computing the intermediate minimum cost from $\mathcal{XU}_0$ to the corresponding region, given by the solution of the following quadratic program

$$J_h = \min_{U_0^{K_h-1}} J(U_0^{K_h-1}, 0, K_h) \tag{4.10a}$$

$$\text{s.t.} \begin{cases} x'(k) & = & A_{i_h(k)}x(k)+B_{i_h(k)}u(k)+f_{i_h(k)} \text{ for } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{P}_{i_h(k)} \\ u_{\min} & \leq & u(k) \leq u_{\max},\ k = 0, 1, \dots, K_h-1 \\ x_{\min} & \leq & x(k) \leq x_{\max},\ k = 1, \dots, K_h \\ x(K_h) & \in & \mathcal{XU}_h \\ x(0) & = & x_0 \end{cases} \tag{4.10b}$$

where $I_0^{K_h-1} = \{i_h(0), \dots, i_h(K_h-1)\}$ is the corresponding switching sequence, and

$$J(U_{K_i}^{K_f-1}, K_i, K_f) \triangleq \left(\sum_{k=K_i}^{K_f-1} \|u(k)\|_R^2 + \|x(k) - x_f\|_Q^2 \right).$$

When no feasible solution to (4.10) exists, we conventionally set $J_h = +\infty$. The advantage provided by these additional calculations is that — once the target region is eventually reached in K steps and we can compute an upper bound J^* on the overall cost — we have an additional termination condition. More precisely, as soon as an evolution intersects $\mathcal{Z}_{\text{fin}}$ at $k = K$, we compute J^* as described before and enforce the following new termination condition

F5 $J_h \geq J^*$, the intermediate cost exceeds or is equal to the current upper bound[10].

Whenever a new exploration reaches $\mathcal{Z}_{\text{fin}}$ in K steps with a lower cost, the upper bound J^* is updated, along with the the correspoding minimizer $U^* = \arg\min J^*$. Note that we can still use the hyper-rectangular approximation and that the partial cost computation is also able to rule out possible infeasible switches as soon as they are detected. In fact, in this case at least one of the intermediate optimal control problems associated to the child nodes will be infeasible.

Node Selection Criterion

The last point to be addressed is the choice of an effective exploration strategy, that is, the ordering criterion according to which new nodes are taken from the list and explored. In order to reduce fruitless explorations, an adequate strategy should recognize the more promising paths to be searched. To this purpose, we suggest the following node selection criterion

NS1 Select the node having the smallest associated *normalized cost* $\hat{J}_h \triangleq J_h/K_h$

where, for convenience, we set $J_0 = \hat{J}_0 = 0$. The normalization factor $1/K_h$ is instrumental to the proposed performance-driving mechanism. In fact, it penalizes, among sequences characterized by identical intermediate cost J_h, those sequences with a smaller minimum cumulated time K_h, which therefore have a larger time-to-go, and are more likely to give rise to a higher overall cost J^*. As a result, the exploration is guided by performance in the sense that the procedure aims at reaching the target set $\mathcal{Z}_{\text{fin}}$ through the most promising evolutions. This strategy leads to tighter upper bounds J^*, and thus to a more effective termination condition **F5**.

```
function $U^*$ = control($x_0$,$\mathcal{U}$,$K$)
1. $\mathcal{XU}(0) = \{x_0\} \times \mathcal{U}$; $J^* \leftarrow +\infty$; $U^* \leftarrow \emptyset$;
2. TREE.init($\mathcal{XU}(0)$); LIST.init($\mathcal{XU}(0), J_0 = 0, T_0 = 0\}$);
3. while not LIST.isEmpty(),
  3.1. $\mathcal{XU}_f = \mathcal{XU}(k)$ = LIST.selectNode(), let $i$ such that $\mathcal{XU}(k) \subseteq \mathcal{P}_i$;
  3.2. $k_0 = k$
  3.3. while not terminate($\mathcal{XU}(k)$,$K_{\max}$,$J^*$),
        3.3.1. if $(\mathcal{Z}_{\text{fin}} \times \mathcal{U}) \bigcap \mathcal{XU}(k) \neq \emptyset$ and $k = K$,
              3.3.1.1. let $J_{\text{fin}} \leftarrow$ minimum cost along the path $\mathcal{XU}_0 - \mathcal{XU}(k) - \mathcal{Z}_{\text{fin}}$;
              3.3.1.2. if $J_{\text{fin}} < J^*$, $J^* = J_{\text{fin}}$; $U^* = \text{argmin} J_{\text{fin}}$;
        3.3.2. if $(\mathcal{XU}(k)) \subseteq \mathcal{P}_i$,
              3.3.2.1. $k \leftarrow k+1$; $\mathcal{X}(k) \leftarrow [A_i\ B_i]\mathcal{XU}(k-1) + f_i$;
              3.3.2.2. $\mathcal{XU}(k) = \mathcal{X}(k) \times \mathcal{U}$;
        3.3.3. else
              3.3.3.1. for all $h \neq i$ such that $\mathcal{XU}_h \triangleq \mathcal{P}_h \bigcap \mathcal{XU}(k) \neq \emptyset$,
                    3.3.3.1.1. let $J_h \leftarrow$ minimum cost along the path $\mathcal{XU}_0 - \mathcal{XU}_h$
                    3.3.3.1.2. if $J_h < J^*$,
                        3.3.3.1.2.1. TREE.connect($\lceil \mathcal{XU}_h \rceil$,$\mathcal{XU}_f$,$k - k_0$);
                        3.3.3.1.2.2. LIST.addNode($\{\lceil \mathcal{XU}_h \rceil, J_h, k\}$);
              3.3.3.2. $\mathcal{XU}(k) \leftarrow (\mathcal{X}(k) \times \mathcal{U}) \bigcap \mathcal{P}_i \bigcap (\mathcal{Z}_{\text{safe}} \times \mathcal{U})$;
4. if $J^* = +\infty$, return infeasible else return $U^*$.
```

Algorithm 4.6.1: Algorithm for optimal control of hybrid systems based on reachability analysis

The Optimal Control Algorithm

The following is a complete algorithmic representation of the optimization procedure described in the previous section.
Steps 3.3.2., 3.3.2.1., 3.3.2.2., 3.3.3., 3.3.3.1. are directly inherited by Algorithm 4.4.1. When LIST is a prioritized stack, the function selectNode() implements **NS1**. If a feasible input sequence U^* is already known, in step 1. J^* can be initialized accordingly, in order to improve the termination condition **F5** already in the early stages of the algorithms. As discussed before, in steps 3.3.3.1.2.1.-3.3.3.1.2.2. the hyper-rectangular outer approximation $\lceil \mathcal{XU}_h \rceil$ is used, rather than $\mathcal{XU}_h$. In step 3.3.1.1., it is conventionally understood that $J_{\text{fin}} \triangleq +\infty$ when no feasible solution exists. The termination conditions **F1–F5** are invoked in Step 3.3. by the function terminate.

Remark 4.6.3 Algorithm 4.6.1 is a branch and bound algorithm, where branching is associated with the switching of the system, and bounding is given by the termination conditions. In particular, conditions **F1 -F4** provide a bound for infeasibility, while **F5** a bound related to the cost function. Compared to a branch and bound MIQP solver, Algorithm 4.6.1 is neither a depth-first nor a breadth-first algorithm, but rather a *best-first* algorithm which exploits the structure of the control problem. The adjective "best-first" stems from the node selection criterion, that aims at exploring first the most promising nodes. Note that the structure of the control problem also determines the way Algorithm 4.6.1 computes the lower bounds. In fact, while a standard MIQP solver would obtain lower bounds by relaxing the integrality constraints, Algorithm 4.6.1 compute

[10]The non-strict inequality in **F5** ensures that, in case of multiple global minima, we obtain only one of the solutions. If the strict inequality is used, the algorithm determines all the optimal solutions.

lower bounds by optimizing over reachable sub-paths. Even though the worst-case performance of this algorithm is lower with respect to plain enumeration, as in general are branch and bound algorithms, the proposed method is considerably faster in the average.

□

Clearly the node selection criterion **NS1** could be augmented to take into account landmarks and select first the evolutions that match a spatial and temporal landmark $\mathcal{Z}_{\text{heur}}[k_{\min}, k_{\max}]$.

Remark 4.6.4 When binary inputs $u_j(k) \in \{0,1\}$ are present ($m_b \neq 0$), they can be handled as shown in [117]. In the particular case where the dynamics of the system is simply linear ($s = 1$), the algorithm executes just one single QP (as only one reach-set computation is performed), in accordance with non-hybrid, conventional finite-horizon linear quadratic solvers.

□

4.6.1 Variations to the Optimization Algorithm

One of the key steps of the optimization algorithm described in the previous section is the computation of the intermediate costs J_h. The approach proposed in Algorithm 4.6.1 solves problem (4.10) at each switching detection. This is computationally expensive, since the number of free inputs (and therefore the computation time for solving the QP) grows as the exploration proceeds.

In this section we propose some alternatives to the termination condition **F5** and the node selection criterion, which can be implemented by computing a lower bound to the intermediate cost J_h, and a lower bound to the cost to go.

Lower Bound to the Intermediate Cost

At each switching detection the minimum cost to get from the father node $\mathcal{XU}_f$ to the new node $\mathcal{XU}_h$ can be computed by solving a QP. Let K_f be the length of the switching sequence associated with the father node $\mathcal{XU}_f$, and let i_f be the region index associated with the father node, the *node-to-node* cost J_b from G_f to G_k is defined as

$$J_b = \min_{U_{K_f}^{K_h-1}} J(U_{K_f}^{K_h-1}, K_f, K_h) \tag{4.11a}$$

$$\text{s.t.} \begin{cases} x'(k) &= A_{i_f}x(k) + B_{i_f}u(k) + f_{i_f}, \quad \text{for } \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \in \mathcal{P}_{i_f} \\ u_{\min} \leq & u(k) \leq u_{\max}, \ k = K_f, \dots, K_h - 1 \\ x_{\min} \leq & x(k) \leq x_{\max}, \ t = K_f + 1, \dots, K_h \\ x(K_f) &\in \mathcal{XU}_f \\ x(K_h) &\in \mathcal{XU}_h \ . \end{cases} \tag{4.11b}$$

Node-to-node costs are clearly properties of the arcs of the tree, exactly as transition times, and the time required for their computation depends on the inter-switching time, $K_h - K_f$, but, contrarily to the partial costs J_h computed in (4.10), does not grow with the absolute time K_h, K_f. The sum of all node-to-node costs along the path to region $\mathcal{XU}_h$ is a lower bound to J_h. The reason for this is simply explained in Fig. 4.9). Compared to the partial cost J_h associated with the path $\mathcal{XU}_0 - \mathcal{XU}_f - \mathcal{XU}_h$, the sum of the node-to-node costs $J_f + J_b$ is computed with one extra artificial degree of freedom, namely that x_1 and x_2 can be different.

Note that using a lower bound to J_h, although cheaper to compute, is less effective when used in place of the exact value in the termination condition **F5**, and renders the node selection criterion less accurate.

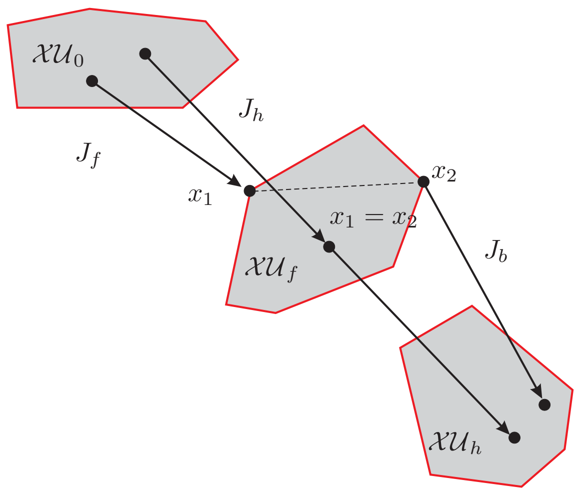


Figure 4.9: Partial cost computation and node-to-node costs: $J_f + J_b \leq J_h$

Lower Bound to the Cost-to-Go

In Section 4.6 we have proposed the lowest J_h/K_h as a criterion for selecting the next node to explore. A possible alternative is to compute the estimate J_{res} of the cost-to-go needed to reach $\mathcal{Z}_{\text{fin}}$ from $\mathcal{XU}_h$, and adopt the following node selection criterion

NS2 Select the node having the lowest $J_h + J_{\text{res}}$

Contrarily to the node selection criterion **NS1**, the alternative criterion **NS2** does not explicitly favors the nodes which are likely to reach the final time T soon (which is in the spirit of a depth-first strategy).

Computing the exact cost-to-go amounts to solve Problem (4.8) from the current arrival time K_h to the final time K. Similarly to (4.9), such a problem can be equivalently formulated as the MIQP, however the solution of the MIQP might be time consuming. Consider

$$\min_{U_{K_h}^{K-1}} \left(\sum_{k=K_h}^{K-1} \|u(k)\|_R^2 + \|x(k) - x_f\|_Q^2 \right) + \|x(K) - x_f\|_P^2 \tag{4.12a}$$

$$\text{s.t.} \begin{cases} \begin{bmatrix} x(k) \\ u(k) \\ x'(k) \end{bmatrix} & \in & \mathcal{CXU}(\mathcal{X},\mathcal{U}), \ k = K_h, \dots, K-1, \\ u_{\min} & \leq & u(k) \ \leq \ u_{\max}, \ k = K_h, \dots, K-1, \\ x_{\min} & \leq & x(k) \ \leq \ x_{\max}, \ k = K_h + 1, \dots, K, \\ x(K_h) & \in & \mathcal{XU}_h, \\ x(K) & \in & \mathcal{Z}_{\text{fin}}, \end{cases} \tag{4.12b}$$

where $\mathcal{CXU}(\mathcal{X},\mathcal{U})$ is the convex hull of the single step flow of the system Σ. Note that problem 4.12 is a QP.

We remark here again that the relaxation obtained by removing the integrality constraints from the MLD model 2.20 is an alternative relaxation and could be used as well to compute J_{res}. However, as the relaxation is worse, the estimated cost-to-go could be over optimistic.

The lower bound J_{res} can be also used to strengthen the termination condition **F5** in the following

F6 $J_h + J_{\text{res}} \geq J^*$, the lower bound to the total cost exceeds or is equal to the current upper bound

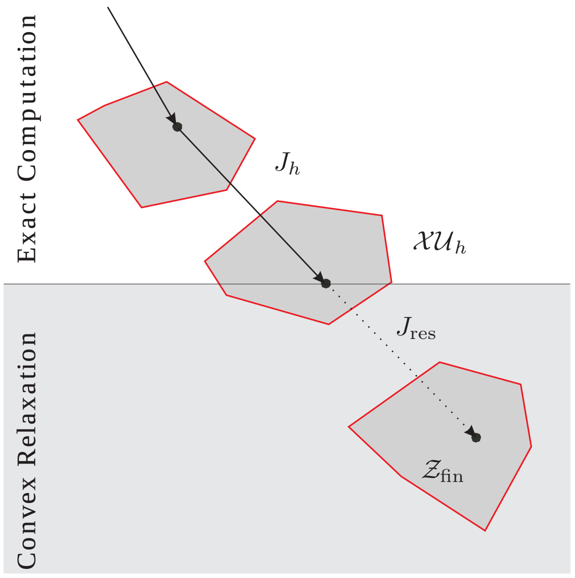


Figure 4.10: Estimate of the total cost $J_h + J_{\text{res}}$ in the termination condition **F6**: J_h is given by the reachability analysis, J_{res} by the relaxed MLD form

where J_h can be either the partial cost or its lower bound presented in Section 4.6.1. As shown in Figure 4.10, the idea is that the total cost from time 0 to time K can be lower-estimated by replacing the PWA dynamics from time k to time K by its convex relaxation. Note that **F5** amounts to use the estimate $J_{\text{res}} = 0$ in **F6**. Clearly, the additional computation needed to compute a better estimate J_{res} and the use of **F6** is only meaningful when $J^* < \infty$.

4.7 Conclusions

For a large class of hybrid systems with switching affine discrete-time dynamics, we have presented a theoretical and algorithmic machinery for computing reach and reachable sets, either exact or over/under-approximated, and for solving complex optimal control problems.

We believe that for a relatively wide class of hybrid systems the approach is computationally viable for (1) verifying the satisfaction of safety, robust stability, and liveness properties for a given set of initial conditions and against prescribed bounded disturbances, and (2) for computing optimal command input sequences according to the philosophy “optimize what is reachable and reach what is optimizing”. It is expected that the method is especially advantageous for seldom switching systems (e.g., over-sampled discrete-time systems).

The potentials of the proposed method are illustrated on a nontrivial hybrid problem in Section 5.1. Another example dealing with the verification of the batch evaporator process benchmark is reported in [37].

Chapter 5

Application Examples

5.1 Analysis and Control of a Cruise Control System

In this section we use HYSDEL to obtain a hybrid model of a car with robotized manual gear shift, and show how such a model can be directly used (i) to formally verify certain safety and liveness properties of a simple cruise controller based on PI control and a set of gear-switching rules and (ii) to synthesize a cruise control system that is piecewise affine and optimal with respect to a certain performance index.

5.1.1 Car model

We focus on a car equipped with manual transmission, and we assume that the gear command is robotized, namely that a slave control system takes care of releasing the clutch, shifting the gear, and engaging the clutch. We only consider the longitudinal dynamics of the car: the continuous variables are the scalar position x (m) and the speed $v = \dot{x}$ (m·s^{-1}). The continuous inputs are the engine torque u_t (Nm), the braking force u_b (N), and the sinus of the road slope u_s, plus six binary inputs g_1, g_2, g_3, g_4, g_5 and $g_R \in \{0, 1\}$ corresponding to the selected gear. Denoting by ω the engine speed (rad·s^{-1}), we have the kinematic relation

$$v = \frac{k_{\text{loss}} r_{\text{wheel}}}{R_g(i) R_{\text{fin}}} \omega, \tag{5.1}$$

where $R_g(i)$ is the gear ratio corresponding to the i-th gear, R_{fin} is the final drive ratio, r_{wheel} is the wheel radius, and k_{loss} is the drive train efficiency level [42]. Note that by using a kinematic relation for the speed engine, we are neglecting the clutch, the motor dynamic behavior, and we are assuming that the time spent for gear shift is negligible.

The dynamic equation of motion of the car is $m\ddot{x} = F_e - F_b - F_r - F_s$ where m (kg) is the vehicle mass, F_e (N) is the traction force, $F_b = u_b$ is the braking force (N), F_r (N) is the friction force, and F_s (N) takes into account the slope of the road. $F_s = mgu_s$ (N), where $u_s = \sin\alpha$ and α is the slope of the road and as a first approximation, we assume that F_r is linear in v, $F_r = \beta v$, where β (kg·m·s^{-1}) is a constant that takes into account all the frictions (i.e. aerodynamic, tires deformation, drive train). From the conservation of mechanical power, we have $F_e v = \omega u_1$, which gives $F_e = \frac{k_{\text{loss}}}{R_g(i)} u_1$. The commanded torque u_t is upper-bounded by the maximum torque deliverable at a certain engine speed ω, $u_t \leq C_e^+(\omega)$ where $C_e^+(\omega)$ is a nonlinear function typically reported in the data sheets of the car and $C_e^-(\omega)$ is the maximum braking force that the engine can provide when the throttle is fully released. In order to derive a hybrid model of the car as described in Section 2.2, we piecewise-linearize $C_e^+(\omega)$ into four regions using the PWL toolbox [96], which requires the introduction of three event variables δ_{PWL1}, δ_{PWL2}, δ_{PWL3}, and as a first approximation, we assume $C_e^-(\omega) = -\alpha_1 - \beta_1\omega$, cf. Fig. 5.1.

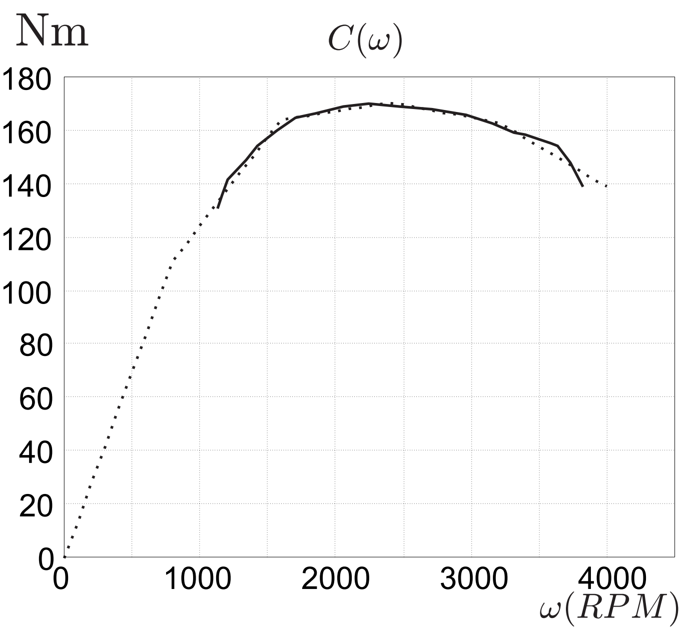


Figure 5.1: Torque of the engine of the Clio 1.9 DTI RXE (solid line) and PWL approximation (dotted line)

The engine speed ω can be written as a SAS, and by (2.3), as the sum of auxiliary continuous variables, $\omega = \omega_1 + \omega_2 + \omega_3 + \omega_4 + \omega_5 + \omega_R$, where

$$\omega_i = \begin{cases} \frac{k_s}{R_g(i)}\dot{x}, & \text{if } g_i = 1, \\ 0, & \text{otherwise.} \end{cases}$$

To validate the model, we took the parameters of the Renault Clio 1.9 DTI RXE from `http://www.renault.com/`. The simulated acceleration and max speed tests gave the same results as the experimental counterpart, reported in the technical documentation. For the reader's convenience we report the main parameters of the car under consideration: $R_g(1) = 3.7271$, $R_g(2) = 2.048$, $R_g(3) = 1.321$, $R_g(4) = 0.971$, $R_g(5) = 0.756$, $R_g(R) = -3.545$, $R_{\text{fin}} = 3.2940$, $k_{\text{loss}} = 0.925$, $r_{\text{wheel}} = 0.2916$ m, $\beta = 25$ kg·m·s^{-1}, $m = 1020$ kg, $\alpha_1 = 10$ Nm, $\beta_1 = 0.3$ kg·m^2·s^{-1}. Figure 5.1 reports the measured torque and the piecewise affine approximation, the maximum error is 5.7 Nm. Finally the dynamics is discretized with sampling time $T_s = 0.5$ s using forward finite differences to obtain the DHA model. The corresponding HYSDEL model of the car is reported in Appendix. The resulting MLD model contains 2 continuous states (vehicle position x and speed v), 3 continuous inputs (engine torque F_e, breaking force F_b, and slope u_s), 6 binary inputs (gears $g_R, g_1, \ldots, g_5$), one continuous output (speed v), 16 auxiliary continuous variables (6 for the traction force, 6 for the engine speed, 4 for the piecewise linearization of the maximum engine torque), 4 auxiliary binary variables (breakpoints for the piecewise linearization of the maximum engine torque), and 96 mixed-integer inequalities. The DHA model can be then transformed into a PWA model using the approach presented in [79][1] or equivalently the MLD model can be converted in a PWA model by running the algorithm proposed in [13][2]. The total number of binary variables is $0+6+4 = 10$, which gives a worst-case number of possible regions in the PWA system equals to $2^{10} = 1024$, while the PWA equivalent to the hybrid MLD model has 30 regions, and is computed

[1]The corresponding software is available for download from `http://control.ee.ethz.ch/~hybrid/hysdel` as a plug-in for HYSDEL.

[2]The corresponding software is available for download from `http://www.dii.unisi.it/~bemporad/tools`

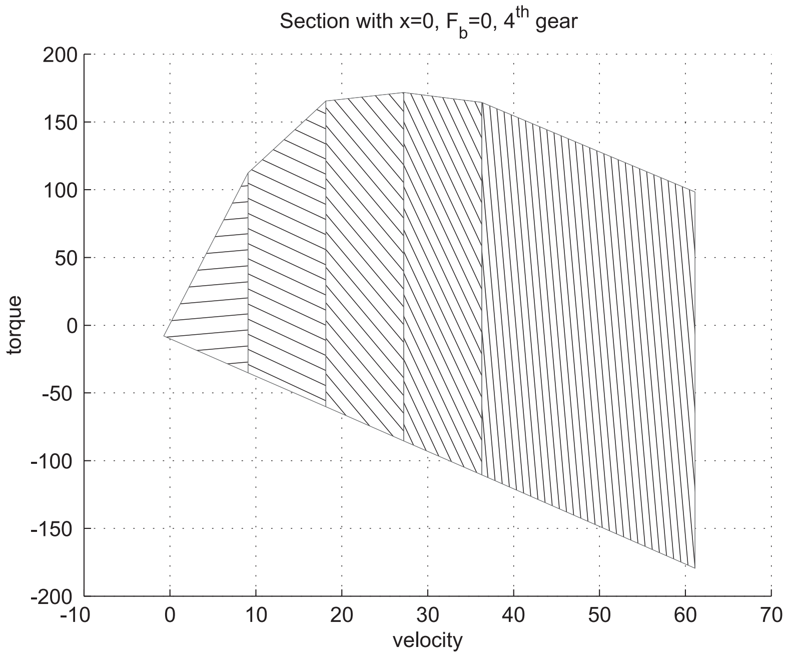


Figure 5.2: PWA system equivalent to the MLD model obtained through HYSDEL from the list reported in Appendix – Section in the (velocity,torque) space for position $x = 0$, braking force $F_b = 0$, gear input vector = $[0\ 0\ 0\ 0\ 1\ 0]^T$ (4^{th} gear)

.

in 7.5 s starting from the DHA and in 72.66 s starting from the MLD using Matlab on a Pentium III 650 MHz machine.

5.1.2 Reachability Analysis

We want to show how the HYSDEL model can be successfully employed to verify safety properties. To this end, we assume we have a simple cruise controller from a previous design. We want to verify that the controlled system will never exceed the target speed by some tolerance, for instance the speed limits imposed by local authorities. The complete hybrid system under examination is now composed of two subsystems: the car dynamic model described in Section 5.1, and the cruise controller. For a detailed description of compositional DHA models, refer to [79].

Model of the Cruise Controller

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

The automaton reported in Fig. 5.3 selects the gear. If the engine speed is faster than $\omega_u = 5000$ RPM then the gear is shifted up. Similarly, if the speed is lower than $\omega_l = 3000$ RPM, the gear is shifted down. The two thresholds are chosen by looking at the max torque plot in Fig. 5.1. To track the desired speed reference $v_r(k)$, the throttle and the brakes are operated by a PI controller. Let $e(k)$ be the integral error, $e'(k) = e(k) + T_s \Delta v(k)$,

$$\begin{tikzcd}
1 \arrow[r, shift left, "w \geq w_u"] & 2 \arrow[l, shift left, "w < w_l"] \arrow[r, shift left, "w \geq w_u"] & 3 \arrow[l, shift left, "w < w_l"] \arrow[r, shift left, "w \geq w_u"] & 4 \arrow[l, shift left, "w < w_l"] \arrow[r, shift left, "w \geq w_u"] & 5 \arrow[l, shift left, "w < w_l"]
\end{tikzcd}$$

Figure 5.3: Gear shift logic controller

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

$$u_t(k) = \begin{cases} k_t \Delta v(k) + i_t e(k), & \text{if } v(k) \leq v_r(k) + 1. \\ 0, & \text{otherwise}, \end{cases} \tag{5.2a}$$

$$u_b(k) = \begin{cases} k_b \Delta v(k), & \text{if } v(k) > v_r(k) + 1, \\ 0, & \text{otherwise}. \end{cases} \tag{5.2b}$$

The control variables u_t and u_b are saturated against the maximum torque and braking force, respectively. The integrator in the PI controller uses an anti-windup scheme: $e(k)$ is integrated only when the control inputs u_t and u_b are not saturated. Note that, the threshold in (5.2) is 1 m/s over the target speed, therefore the fine tracking of the speed reference is performed using only the command coming for the throttle. By fixing the gear ratio in fifth gear we calibrate the parameters k_t, k_b, and i_t on the resulting linear system ($k_t = 70, k_b = 20$, and $i_t = 10$). The HYSDEL model of the car is reported in the Appendix, and it is available together with the cruise control system in the HYSDEL distribution [130]. The corresponding MLD model (2.20) has 173 MLD constraints, $x \in \mathbb{R}^2 \times \{0,1\}^5$, $d \in \{0,1\}^{15}$, $z \in \mathbb{R}^{19}$.

Verification

The HYSDEL compiler is used to generate a PWA model of the cruise control system. The verification is performed using the algorithm presented in [37]. We check the above mentioned safety requirement, namely that the cruise control will never accelerate the car over the speed limits. As the safety specification is independent of the car position we omit this from the model and we use the following initial set $\mathcal{X}(0) = \{x = \left[\begin{smallmatrix} v \\ e \end{smallmatrix}\right] : v \in [0,1], e \in [0,1]\}$ and target set: $\mathcal{Z}_1 = \{v : v > v_r + r_{\text{toll}}\}$ where r_{toll} is a tolerance, in this example we set $r_{\text{toll}} = 1.3889$ m·s^{-1} (5 km/h) that is for instance the tolerance of the speed limit enforcement devices adopted in Switzerland. Moreover, we check the liveness of the controller by adding the set $\mathcal{Z}_2 = \{v, t : v \leq v_r - 2r_{\text{toll}}, t > 10/T_s\}$, where we require that the controller reaches the target speed minus the tolerance $2r_{\text{toll}}$ in 10 s (a controller that stops the car would be safe against fines, but not at all desirable!). We perform parametric verification [37] for a class of constant references $v_r \in [8.333, 19.444]$ m/s $(= [30, 70]$ km/h). The exploration horizon is fixed to $T_{max} = 15.5$s ($K_{\max} = T_{\max}/T_s = 31$ steps).

Verification Results

The result of the verification algorithm is that the controlled system satisfies both the specifications: It does not enter the unsafe region $\mathcal{Z}_1$ (over the speed limit) and guarantees the liveness of the control action (within $10s$ the speed v is in a bounded set around the target speed v_r. The verification required $9109s$ on a PC Pentium 650 MHz running Matlab 5.3.

The algorithm was run also for the same initial and target sets and for an extended range of the parameter $v_r \in [30, 120]$ km/h. The algorithm reported the first counterexample in $415s$: for $v_r = 105.0012$ km/h) the liveness condition is not satisfied. In fact,

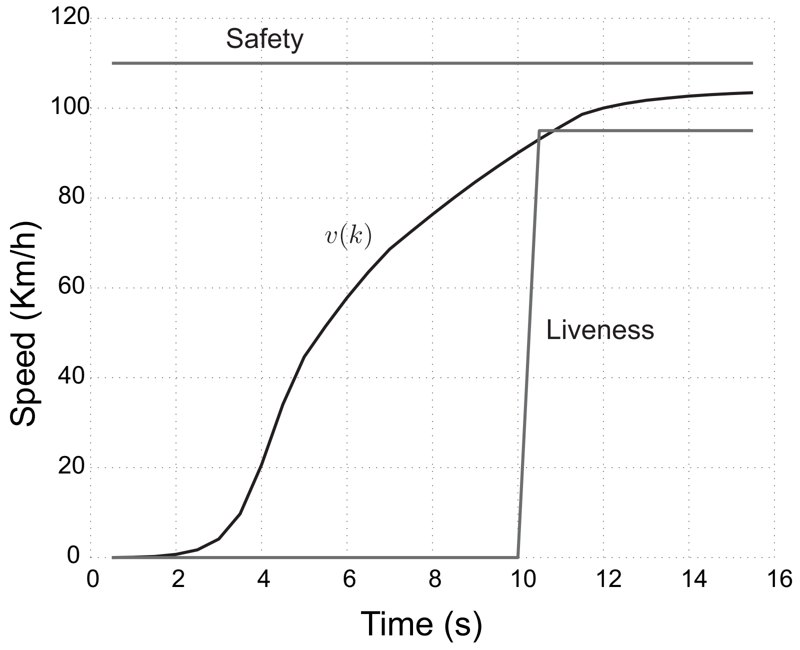


Figure 5.4: Counterexample to the liveness property

by examining the plot of the counterexample reported in Fig. 5.4, one can see that the controller fails to reach the requested vehicle speed within the specified time frame.

5.1.3 Cruise Control Design

Now we replace the heuristic controller with an optimal one. We use the hybrid model of the car to synthesize a cruise control system that commands the gear ratio (discrete input) and gas pedal and brakes (continuous inputs) in order to track a desired speed and minimize fuel consumption. To this end, we design a receding horizon controller and derive its equivalent explicit piecewise affine form [16], so that the cruise controller becomes a look-up table of affine functions of the measured velocity and reference signals, that can be easily implemented in real time. Since the controller does not depend on the position of the car, we will remove x from the model.

The main idea of the approach is to setup a finite-horizon optimal control problem for the hybrid MLD system (2.20) by optimizing a performance index under operating constraints. In particular, we minimize

$$\min_{\xi} J(v(k), v_d(k)) \triangleq |v'(k) - v_d(k)| + \rho|\omega(k)|, \tag{5.3a}$$

$$\text{s.t.} \begin{cases} v'(k) & = & Av(k) + B_1 u(k) + \\ & & B_2\delta(k) + B_3 z(k), \\ E_2\delta(k) & + & E_3 z(k) \leq E_1 u(k) + \\ & & E_4 x(k) + E_5, \end{cases} \tag{5.3b}$$

where $v(k)$ is the measured velocity of the car at time $t = kT_s$, and $\xi \triangleq [u^T(k), \delta^T(k), z^T(k)]^T$ is the optimization vector.

As remarked above, the design of the controller is performed in two steps. First, the RHC controller based on the optimal control problem (5.3) is tuned in simulation using MILP solvers [92], until the desired performance is achieved. The RHC controller is not directly implementable, as it would require a MILP to be solved on-line, which is clearly prohibitive on standard automotive control hardware. Therefore, for implementation, in

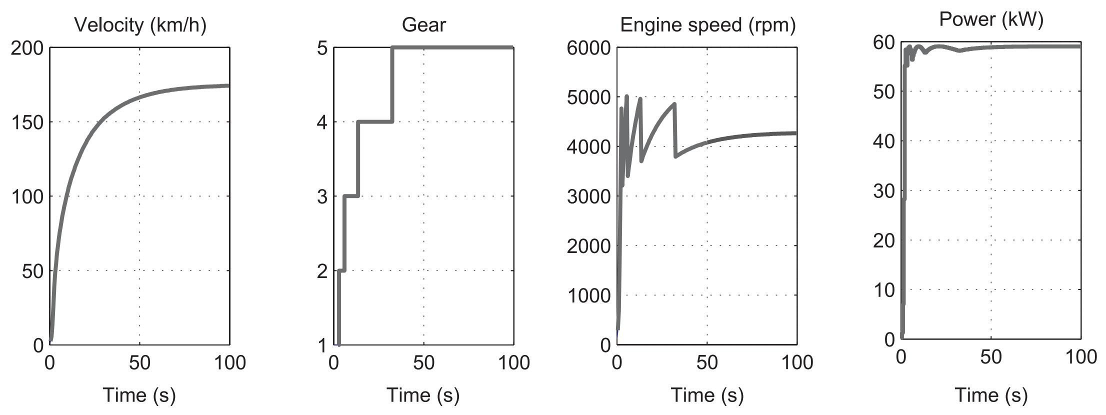


Figure 5.5: Maximum acceleration profiles

the second phase the explicit piecewise affine form of the RHC law is computed off-line by using a multiparametric mixed integer linear programming (mp-MILP) solver, according to the approach of [16], which provides the optimal control action as a piecewise affine function of the measured (or estimated) state vector of the hybrid system and reference signals. As a result, the state+reference space is partitioned into polyhedral sets, where an affine control law is defined in each polyhedron.

As a first design step, we choose $\rho = 0.001$ m/rad in (5.3a). The corresponding multiparametric mixed-integer linear programming has 98 linear inequalities, 19 continuous variables, 10 binary variables, 2 parameters ($v(k)$, $v_d(k)$), and is solved in 27 m on a Sun Ultra 10 running Matlab 5.3 and Cplex. The corresponding piecewise affine control law consists of 49 regions.

For a commanded speed $v_d = 250$ km/h, which cannot be reached by the car, the cruise controller leads to the maximum acceleration curves depicted in Fig. 5.5, that are very close to those reported in the data sheets.

Figure 5.6 shows the closed-loop trajectories for a few changes of the velocity set-point. During the shift from 0 to 120 km/h, the cruise controller commands the gears similarly to what shown in Fig. 5.5, with full-throttle and no action on the brakes. When the set-point changes from 120 km/h to 50 km/h, the cruise controller does its best to slow down the car: switch to second gear, use full brakes, release the gas pedal. As soon as the set-point is recovered, the weight ρ on the fuel consumption leads back to fifth gear. The simulation also includes a nonzero road slope, which acts as an unmeasured and unmodeled disturbance to be rejected by the cruise controller.

Clearly, the controller is too aggressive during the set-point transition. This can be easily fixed by adding in (5.3) the constraint

$$|v'(k) - v(k)| < T_s a_{\max},$$

where $a_{\max}$ is the maximum acceleration tolerated. The resulting MILP problem has 100 linear inequalities, and is solved multiparametrically in 28 m, leading to a partition of the (v, v_d) space into 54 regions. The corresponding closed-loop trajectories are depicted in Fig. 5.7, where a better drive comfort is clearly apparent.

We remark that the cruise control system described in this section has to be considered as a pure exercise of modeling and control synthesis for hybrid systems, and there is no claim that it is sensible, as it is, in a real application. For instance, it is apparent the rotation speed of the engine $\omega \approx 800$ rpm depicted in Figure 5.6 would not be realistic for most commercial vehicles. A more comprehensive study for the synthesis of a supervisor for automatic gear shifting is currently under investigation in collaboration

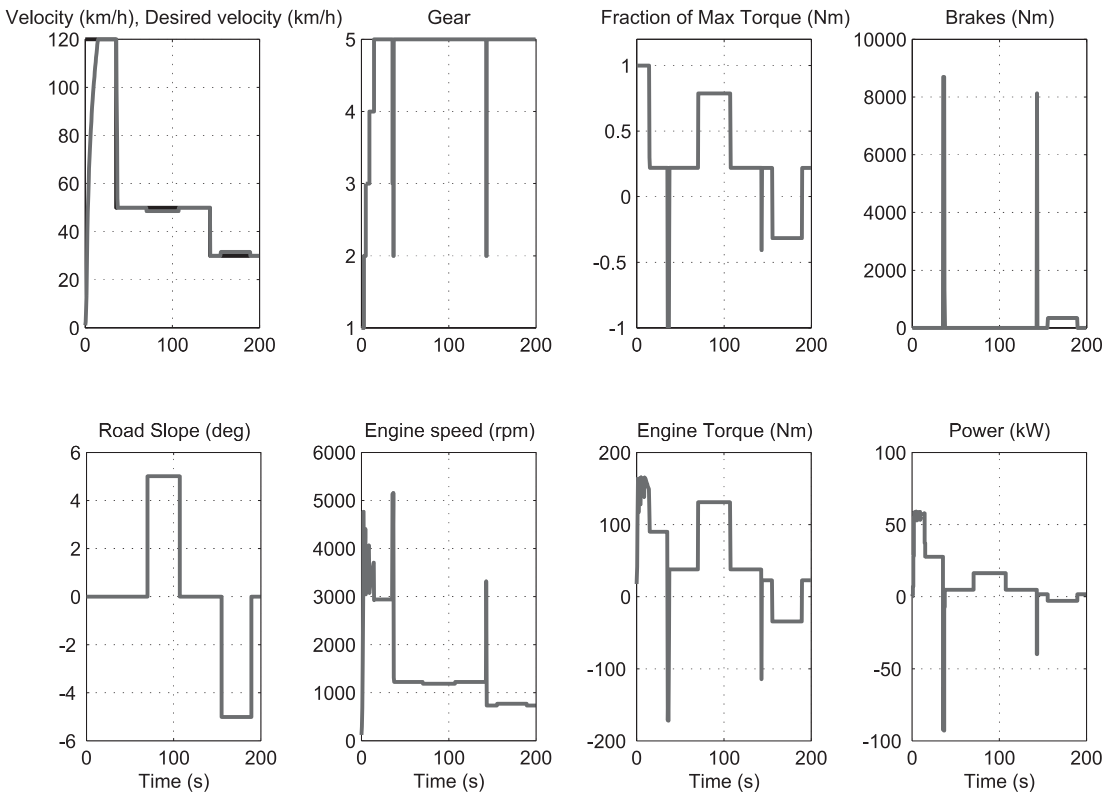


Figure 5.6: Closed-loop profiles: aggressive control action

with the Fiat Research Center, Italy [14].
HYSDEL Code — Car Model

```
SYSTEM car {
INTERFACE {
  STATE {
    REAL position [-1000, 1000];
    REAL speed    [-50*1000/3600, 220*1000/3600];
    }
  INPUT {
    REAL torque  [-300,300]; /* Nm */
    REAL F_brake [0,9000];   /* N  */
    REAL slope   [0, 1];
    BOOL gear1, gear2, gear3, gear4, gear5, gearR;
        }
  OUTPUT {
    REAL position_y, speed_y, w_y;
    }
  PARAMETER {
    REAL mass = 1020; /* kg */
    REAL beta_friction = 25; /* W/m*s */
    REAL Rgear1 = 3.7271;
```

Parameters omitted for brevity, full list available in [130].

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

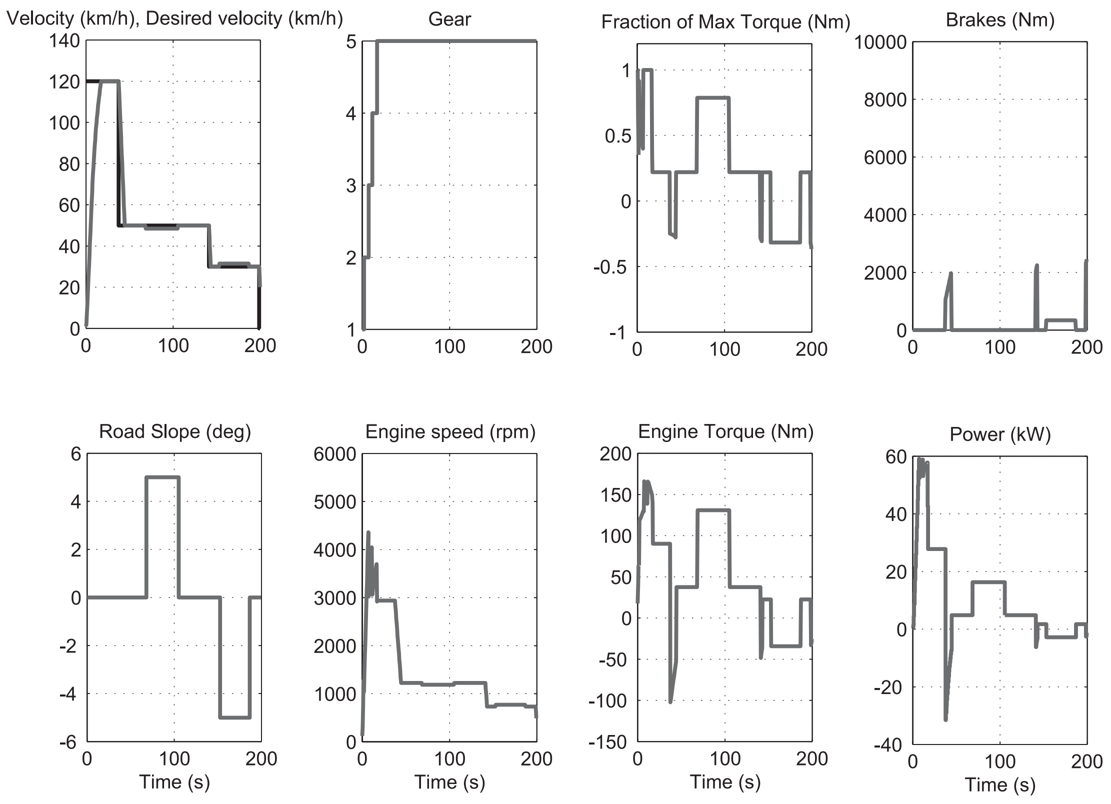


Figure 5.7: Closed-loop profiles: smoother control action

```
  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 / sf * Rgear1};
  Fe2 = {IF gear2 THEN torque / sf * Rgear2};
  Fe3 = {IF gear3 THEN torque / sf * Rgear3};
  Fe4 = {IF gear4 THEN torque / sf * Rgear4};
  Fe5 = {IF gear5 THEN torque / sf * Rgear5};
  FeR = {IF gearR THEN torque / sf * RgearR};
  w1 = {IF gear1 THEN speed / sf * Rgear1};
  w2 = {IF gear2 THEN speed / sf * Rgear2};
  w3 = {IF gear3 THEN speed / sf * Rgear3};
  w4 = {IF gear4 THEN speed / sf * Rgear4};
  w5 = {IF gear5 THEN speed / sf * Rgear5};
  wR = {IF gearR THEN speed / sf * 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) - g * slope;
}
OUTPUT {
  position_y = position;
  speed_y = speed;
```

```
    w_y = (w1 + w2 + w3 + w4 + w5 + wR);
  }
  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; F_brake <= max_brake_force;
    -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;
  }
}
}
```

HYSDEL Code — Cruise Control System

```
SYSTEM carcruise { INTERFACE {
  STATE {
    REAL speed [-50*1000/3600, 220*1000/3600];
    REAL err [-50*100/3600, 220*100/3600]; /* integral */;
    REAL vr [-50*1000/3600, 220*1000/3600];
    BOOL gear1, gear2, gear3, gear4, gear5;
  }
  OUTPUT {
    REAL y, w;
  }
  PARAMETER {
    REAL mass = 1020; /* kg */
    REAL g = 9.8; /* m/s^2 , not used*/
    REAL beta_friction = 25; /* W/m*s */
```

Parameters omitted for brevity.

```
    }
}
IMPLEMENTATION {
  AUX {
    REAL  Fe1,   Fe2,   Fe3,   Fe4,   Fe5,
           w1,    w2,    w3,    w4,    w5,
         DCe1,  DCe2,  DCe3,  DCe4;
    BOOL dPWL1, dPWL2, dPWL3, dPWL4;
    BOOL sd, su, verr;
    REAL zut, zub;
    BOOL sat_torque, sat_F_brake;
    REAL torque, F_brake;
    BOOL no_sat;
    REAL ierr;
  }
  LOGIC {
    no_sat = ~(sat_torque | sat_F_brake) & verr;
  }
  AD {
    dPWL1 = wPWL1 <= (w1 + w2 + w3 + w4 + w5);
    dPWL2 = wPWL2 <= (w1 + w2 + w3 + w4 + w5);
    dPWL3 = wPWL3 <= (w1 + w2 + w3 + w4 + w5);
    dPWL4 = wPWL4 <= (w1 + w2 + w3 + w4 + w5);
    sd = (w1 + w2 + w3 + w4 + w5) <= wl;
    su = wu <= (w1 + w2 + w3 + w4 + w5);
    verr = speed - vr - 2 <= 0;
    sat_torque = (DCe1 + DCe2 + DCe3 + DCe4) + 1 <= zut;
    sat_F_brake = - zub + max_brake_force <= 0;
  }
  DA {
    Fe1 = {IF gear1 THEN torque / sf * Rgear1};
    Fe2 = {IF gear2 THEN torque / sf * Rgear2};
    Fe3 = {IF gear3 THEN torque / sf * Rgear3};
    Fe4 = {IF gear4 THEN torque / sf * Rgear4};
    Fe5 = {IF gear5 THEN torque / sf * Rgear5};
    w1 = {IF gear1 THEN speed / sf * Rgear1};
    w2 = {IF gear2 THEN speed / sf * Rgear2};
    w3 = {IF gear3 THEN speed / sf * Rgear3};
    w4 = {IF gear4 THEN speed / sf * Rgear4};
    w5 = {IF gear5 THEN speed / sf * Rgear5};
    DCe1 = {IF dPWL1 THEN (aPWL2) + (bPWL2) *
      (w1 + w2 + w3 + w4 + w5) ELSE (aPWL1) +
      (bPWL1) * (w1 + w2 + w3 + w4 + w5)};
    DCe2 = {IF dPWL2 THEN (aPWL3 - aPWL2) +
      (bPWL3 - bPWL2) * (w1 + w2 + w3 + w4 + w5)};
    DCe3 = {IF dPWL3 THEN (aPWL4 - aPWL3) +
      (bPWL4 - bPWL3) * (w1 + w2 + w3 + w4 + w5)};
    DCe4 = {IF dPWL4 THEN (aPWL5 - aPWL4) +
      (bPWL5 - bPWL4) * (w1 + w2 + w3 + w4 + w5)};
    zut = {IF verr THEN kt * (vr - speed) + it * err};
    zub = {IF ~verr THEN - kb * (vr - speed) - ib * err};
    torque ={IF sat_torque THEN (DCe1 + DCe2 + DCe3 + DCe4)
      + 1 ELSE zut};
    F_brake = {IF sat_F_brake THEN max_brake_force
      ELSE zub};
    ierr = {IF no_sat THEN err + Ts * (vr - speed)};

  }
  CONTINUOUS {
    speed = speed + Ts / mass * (Fe1 + Fe2 + Fe3 + Fe4 + Fe5
       - F_brake - beta_friction * speed);
    err = ierr;
    vr = vr;
  }
  AUTOMATA {
    gear1 =              (gear2 & sd)|(gear1 &       ~su);
    gear2 = (gear1 & su)|(gear3 & sd)|(gear2 & ~sd & ~su);
    gear3 = (gear2 & su)|(gear4 & sd)|(gear3 & ~sd & ~su);
    gear4 = (gear3 & su)|(gear5 & sd)|(gear4 & ~sd & ~su);
    gear5 = (gear4 & su)             |(gear5 & ~sd );
  }
  OUTPUT {
    y = speed;
    w = w1 + w2 + w3 + w4 + w5;
  }
  MUST {
    /* max engine speed */
    /* wemin <= w1+w2+w3+w4+w5 <= wemax */
    -w1 <= -wemin; w1 <= wemax; -w2 <= -wemin;
    w2 <= wemax; -w3 <= -wemin; w3 <= wemax;
    -w4 <= -wemin; w4 <= wemax; -w5 <= -wemin;
    w5 <= wemax; F_brake <= max_brake_force;
    -F_brake <= 0; /* brakes cannot accelerate ! */
    torque - (DCe1 + DCe2 + DCe3 + DCe4) - 1 <= 0;
    /* xor(gear1,gear2,gear3,gear4,gear5,gearR)=TRUE */
    -((REAL gear1) + (REAL gear2) + (REAL gear3) +
      (REAL gear4) + (REAL gear5)) <= -0.9999;
    (REAL gear1) + (REAL gear2) + (REAL gear3) +
      (REAL gear4) + (REAL gear5)  <= 1.0001;
    dPWL4 -> dPWL3; dPWL4 -> dPWL2; dPWL4 -> dPWL1;
    dPWL3 -> dPWL2; dPWL3 -> dPWL1; dPWL2 -> dPWL1;
  }
}
}
```

Chapter 6

Conclusions

We introduced Discrete Hybrid Automata as a general modeling framework for obtaining hybrid models oriented to the solution of analysis and synthesis problems. The language HYSDEL describes DHA at a high level and its associated compiler generates the corresponding computational models. This simplifies the use of the whole theory and set of tools available for different classes of hybrid systems for solving control, state estimation and verification problems. The effectiveness of HYSDEL was shown on an automotive case study.

HYSDEL has been successfully used in several industrial applications. In [40] the authors modeled the hybrid behavior of a vehicle/tyre system and designed a traction controller that improves a driver's ability to control a vehicle under adverse external conditions such as wet or icy roads. Another automotive application was presented in [22], where the focus is on the application of hybrid modeling and optimal control to the problem of air-to-fuel ratio and torque control in advanced gasoline direct injection stratified charge (DISC) engines. In both cases, the control design leaded to a control law that can be implemented on automotive hardware as a piecewise affine function of the measured and estimated quantities. In [70] the economic optimization of a combined cycle power plant was accomplished by modeling the system in HYSDEL (turning on/off the gas and steam turbine, operating constraints, different modalities start up of the turbines), and then using the generated MLD model in a mixed integer linear optimization algorithm [92].

The HYSDEL compiler is available on-line at `http://control.ee.ethz.ch/~hybrid/hysdel`.

Acknowledgments

Most of the contents of these notes is based on papers that I published with several coauthors. I am thankful to all of them, and especially to Fabio Torrisi and Francesco Borrelli.

Bibliography

[1] J. Acevedo and E.N. Pistikopoulos. A multiparametric programming approach for linear process engineering problems under uncertainty. *Ind. Eng. Chem. Res.*, 36:717–728, 1997.

[2] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. Hybrid modeling and simulation of biomolecular networks. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, 2001.

[3] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[4] 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.

[5] A. Asarin, O. Maler, and A. Pnueli. On the analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.

[6] 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.

[7] E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data-structures for the verification of timed automata. In O. Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *Lecture notes in Computer Science*, pages 346–360. Springer-Verlag, 1997.

[8] D. Avis, D. Bremmer, and R. Seidel. How good are convex hull algorithms? *Computational Geometry: Theory and Applications*, 7:265–301, 1997.

[9] A. Balluchi, M. Di Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Hybrid control for automotive engine management: the cut-off case. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 13–32. Springer-Verlag, 1998.

[10] 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.

[11] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In M.D. Di Benedetto and A. Sangiovanni Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–162. Springer-Verlag, 2001.

[12] A. Bemporad. Identification of hybrid systems: Global convergence via mixed-integer programming. Technical Report AUT00-28, ETH, Zurich, September 2000.

[13] A. Bemporad. An efficient technique for translating mixed logical dynamical systems into piecewise affine systems. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 1970–1975, 2002.

[14] A. Bemporad, P. Borodani, and M. Mannelli. Hybrid control of an automotive robotized gearbox for reduction of consumptions and emissions. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 81–96. Springer-Verlag, 2003.

[15] A. Bemporad, F. Borrelli, and M. Morari. Optimal controllers for hybrid systems: Stability and piecewise linear explicit form. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 1810–1815, Sydney, Australia, December 2000.

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

[17] A. Bemporad, F. Borrelli, and M. Morari. On the optimal control law for linear discrete time hybrid systems. In M. Greenstreet and C. Tomlin, editors, *Hybrid Systems: Computation and Control*, number 2289 in Lecture Notes in Computer Science, pages 105–119. Springer-Verlag, 2002.

[18] 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.

[19] A. Bemporad, C. Filippi, and F.D. Torrisi. Inner and outer approximation of polytopes using boxes. *Computational Geometry*, 2002. Submitted for publication.

[20] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A greedy approach to identification of piecewise affine models. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 97–112. Springer-Verlag, 2003.

[21] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. Set membership identification of piecewise affine models. In *13th IFAC Symposium on System Identification*, Rotterdam, The Netherlands, August 2003.

[22] A. Bemporad, N. Giorgetti, I.V. Kolmanovsky, and D. Hrovat. Hybrid modeling and control of a direct injection stratified charge engine. In *HSCC'02*.

[23] A. Bemporad, N. Giorgetti, I.V. Kolmanovsky, and D. Hrovat. Hybrid modeling and control of a direct injection stratified charge engine. In *Symposium on Advanced Automotive Technologies, ASME International Mechanical Engineering Congress and Exposition*, New Orleans, Louisiana, November 2002.

[24] A. Bemporad, N. Giorgetti, I.V. Kolmanovsky, and D. Hrovat. A hybrid system approach to modeling and optimal control of DISC engines. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 1582–1587, 2002.

[25] A. Bemporad, L. Giovanardi, and F.D. Torrisi. Performance driven reachability analysis for optimal scheduling and control of hybrid systems. Technical Report AUT00-15, Automatic Control Laboratory, ETH Zurich, Switzerland, September 2000. To be submitted.

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

[27] A. Bemporad, A. Giua, and C. Seatzu. Synthesis of state-feedback optimal controllers for switched linear systems. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 3182–3187, 2002.

[28] A. Bemporad, W.P.M.H. Heemels, and B. De Schutter. On hybrid systems and closed-loop MPC systems. *IEEE Trans. Automatic Control*, 47(5):863–869, May 2002.

[29] A. Bemporad and D. Mignone. *MIQP.M: A Matlab function for solving mixed integer quadratic programs*. ETH Zurich, 2000. See related web page at `http://www.dii.unisi.it/~hybrid/tools/miqp`. Code also available at `http://control.ethz.ch/~hybrid/miqp`.

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

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

[32] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In F.W. Vaandrager and J.H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 1999.

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

[34] A. Bemporad, J. Roll, and L. Ljung. Identification of hybrid systems via mixed-integer programming. Technical Report AUT00-29, ETH, Zurich, October 2000.

[35] 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.

[36] A. Bemporad, F.D. Torrisi, and M. Morari. Performance analysis of piecewise linear systems and model predictive control systems. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 4957–4962, Sydney, Australia, December 2000.

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

[38] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, Yi Wang, and Carsten Weise. New Generation of UPPAAL. In *Int. Workshop on Software Tools for Technology Transfer*, June 1998.

[39] F. Borrelli, M. Baotic, A. Bemporad, and M. Morari. An efficient algorithm for computing the state feedback optimal control law for discrete time hybrid systems. In *American Control Conference*, Denver, Colorado, 2003.

[40] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. A hybrid approach to traction control. In A. Sangiovanni-Vincentelli and M.D. Di Benedetto, editors, *Hybrid Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Science, pages 162–174. Springer-Verlag, 2001.

[41] F. Borrelli, A. Bemporad, and M. Morari. A geometric algorithm for multi-parametric linear programming. *Journal of Optimization Theory and Applications*, 2002. Accepted for publication as a regular paper.

[42] R. Bosch. *Automotive Handbook*. Society of Automotive Engineers, 5 edition, December 2000.

[43] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 73–88. Springer-Verlag, 2000.

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

[45] M.S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. Automatic Control*, 43(4):475–482, April 1998.

[46] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: model and optimal control theory. *IEEE Trans. Automatic Control*, 43(1):31–45, 1998.

[47] M.S. Branicky and S.K. Mitter. Algorithms for optimal hybrid control. In *Proc. 34th IEEE Conf. on Decision and Control*, New Orleans, USA, December 1995.

[48] P.J. Campo and M. Morari. Robust model predictive control. In *American Control Conference*, volume 2, pages 1021–1026, 1987.

[49] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[50] C. G. Cassandras, Q. Liu, K. Gokbayrak, and D.L. Pepyne. Optimal control of a two-stage hybrid manufacturing system model. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 450–455, Phoenix, AZ, December 1999.

[51] C.G. Cassandras, D.L. Pepyne, and Y.Wardi. Optimal control of a class of hybrid systems. *IEEE Trans. Automatic Control*, 46(3):3981–415, 2001.

[52] T.M. Cavalier, P.M. Pardalos, and A.L. Soyster. Modeling and integer programming techniques applied to propositional calculus. *Computers Opns Res.*, 17(6):561–570, 1990.

[53] M.K. Çamlıbel, W.P.M.H. Heemels, and J.M. Schumacher. Consistency of a time-stepping method for a class of piecewise linear networks. *IEEE Trans. Circuits Syst. I*, 49(3):349–357, 2002.

[54] V. Chandru and J.N. Hooker. *Optimization methods for logical inference*. Wiley-Interscience, 1999.

[55] D. Christiansen. *Electronics Engineers' Handbook, 4th edition*. IEEE Press/ McGraw Hill, Inc., 1997.

[56] A. Chutinan and B.H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Trans. Automatic Control*, 46(9):1401–1410, 2001.

[57] A. Chutinan and B.H. Krogh. Computational techniques for hybrid system verification. *IEEE Trans. Automatic Control*, 48(1):64–75, 2003.

[58] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[59] F.A. Cuzzola and M. Morari. A generalized approach for analysis and control of discrete-time piecewise affine and hybrid systems. In A. Sangiovanni-Vincentelli and M.D. Di Benedetto, editors, *Hybrid Systems: Computation and Control*, number 2034 in Lecture Notes in Computer Science, pages 189–203. Springer-Verlag, 2001.

[60] T. Dang and O. Maler. Reachability analysis via face lifting. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109. Springer-Verlag, 1998.

[61] Dash Associates. *XPRESS-MP User Guide*, 2003. `http://www.dashopt.com`.

[62] B. De Schutter. Optimal control of a class of linear hybrid systems with saturation. *SIAM J. Control Optim.*, 39(3):835–851, 2000.

[63] B. De Schutter and B. De Moor. The extended linear complementarity problem and the modeling and analysis of hybrid systems. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 70–85. Springer, 1999.

[64] B. De Schutter and T. van den Boom. Model predictive control for max-plus-linear systems. In *American Control Conference*, pages 4046–4050, 2000.

[65] B. De Schutter and T. van den Boom. On model predictive control for max-min-plus-scaling discrete event systems. Technical Report bds:00-04, Control Laboratory, Fac. of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands, February 2000.

[66] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Proc. of the Workshop Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer-Verlag, 1989.

[67] V.D. Dimitriadis, N. Shah, and C.C. Pantelides. Modeling and safety verification of discrete/continuous processing systems. *AIChE Journal*, 43:1041–1059, 1997.

[68] V. Dua and E.N. Pistikopoulos. An algorithm for the solution of multiparametric mixed integer linear programming problems. *Annals of Operations Research*, pages 123–139, 2000.

[69] G. Ferrari-Trecate, F.A. Cuzzola, D. Mignone, and M. Morari. Analysis of discrete-time piecewise affine and hybrid systems. *Automatica*, 38:2139–2146, 2002.

[70] G. Ferrari-Trecate, E. Gallestey, P. Letizia, M. Spedicato, M. Morari, and M. Antoine. Modeling and control of co-generation power plants: A hybrid system approach. In C. J. Tomlin and M. R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 209–224. Springer-Verlag, 2002.

[71] G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Trans. Automatic Control*, 47(10):1663–1676, 2002.

[72] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. In A. Sangiovanni-Vincentelli and M.D. Di Benedetto, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[73] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, February 2003.

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

[75] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. *SIAM J. Optim.*, 8(2):604–616, May 1998.

[76] C. A. Floudas. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995.

[77] K. Fukuda. *cdd/cdd+ Reference Manual*. Institute for Operations Research, ETH-Zentrum, CH-8092 Zurich, Switzerland, 0.61 (cdd) 0.75 (cdd+) edition, December 1997.

[78] K. Fukuda. From the zonotope construction to the minkowski addition of convex polytopes. Technical report, Preprint, School of Computer Scince, McGill University, Montreal, Canada, August 2003.

[79] T. Geyer, F.D. Torrisi, and M. Morari. Efficient Mode Enumeration of Compositional Hybrid Models. In A. Pnueli and O. Maler, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer-Verlag, 2003.

[80] E.G. Gilbert and K. Tin Tan. Linear systems with state and control constraints: the theory and applications of maximal output admissible sets. *IEEE Trans. Automatic Control*, 36(9):1008–1020, 1991.

[81] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[82] 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.

[83] P. Gritzmann and V. Klee. On the complexity of some basic problems in computational convexity: I. Containment problems. *Discrete Mathematics*, 136:129–174, 1994.

[84] P. Gritzmann and B. Sturmfels. Minkowsky addition of polytopes: Computational complexity and applications to Gröbner bases. *SIAM Journal on Discrete Mathematics*, 6(2):246–269, May 1993.

[85] 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.

[86] W.P.H.M Heemels, B. de Schutter, and A. Bemporad. On the equivalence of classes of hybrid dynamical models. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 364–369, Orlando, Florida, 2001.

[87] W.P.M.H. Heemels. *Linear complementarity systems: a study in hybrid dynamics*. PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands, 1999.

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

[89] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 130–144. Springer-Verlag, 2000.

[90] 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 *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, 2001.

[91] T. Huerlimann. *Reference Manual for the LPL Modeling Language, Version 4.42*. Departement for Informatics, Université de Fribourg, Switzerland, `http://www2-iiuf.unifr.ch/tcs/lpl/TonyHome.htm`, 2001.

[92] ILOG, Inc. *CPLEX 8.1 User Manual*. Gentilly Cedex, France, 2003.

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

[94] T.A. Johansen, J. Kalkkuhl, J. Lüdemann, and I. Petersen. Hybrid control strategies in ABS. In *American Control Conference*, Arlington, Virginia, June 2001.

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

[96] P. Julián. *A High Level Canonical Piecewise Linear Representation: Theory and Applications*. Phd thesis, Universidad National del Sur, Bah/'ıa Blanca, 1999.

[97] P. Julian, M. Jordan, and A. Desages. Canonical piecewise-linear approximation of smooth functions. *IEEE Trans. Circuits and Systems — I: Fundamental Theory and Applications*, 45(5):567–571, May 1998.

[98] I. Kolmanovsky and E.G. Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4:317–367, 1998.

[99] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In B. Krogh and N. Lynch, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 202–214. Springer-Verlag, 2000.

[100] B. Lincoln and A. Rantzer. Optimizing linear system switching. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 2063–2068, Orlando, FL, USA, 2001.

[101] J. Lygeros, D.N. Godbole, and S. Sastry. A game theoretic approach to hybrid system design. In R. Alur and T. Henzinger, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1996.

[102] J. Lygeros, K.H. Johansson, S.N. Simic, J. Zhang, and S.S. Sastry. Dynamical properties of hybrid automata. *IEEE Trans. Automatic Control*, 48:2–17, January 2003.

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

[104] A. Makhorin. *GLPK (GNU Linear Programming Kit) User's Guide*, 2003.

[105] F. Martinelli. A scheduling problem for n competing queues with finite capacity. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 2276–2281, Phoenix, AZ, December 1999.

[106] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[107] D. Mignone. *Control and Estimation of Hybrid Systems with Mathematical Optimization*. PhD thesis, Automatic Control Labotatory - ETH, Zurich, 2002.

[108] D. Mignone, G. Ferrari-Trecate, and M. Morari. Stability and stabilization of piecewise affine and hybrid systems: An LMI approach. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 504–509, December 2000.

[109] G. Mitra, C. Lucas, and Moody S. Tool for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research*, 71:262–276, 1994.

[110] R. Möbus, M. Baotic, and M. Morari. Multi-objective adaptive cruise control. In *HSCC'03, Prague, The Czech Republic*, April 2003. Accepted.

[111] M. Morari, A. Bemporad, and D. Mignone. A framework for control, state estimation, fault detection, and verification of hybrid systems. *Automatisierungstechnik*, 47:374–381, 1999.

[112] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[113] Th. Ottmann, S. Schuierer, and S. Soundaralakshmi. Enumerating extreme points in higher dimensions. In E.W. Mayer and C. Puech, editors, *STACS 95: 12th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 562–570. Springer-Verlag, 1995.

[114] C.C. Pantelides, M.P. Avraam, and N. Shah. Optimisation of hybrid dynamic processes. In F. Keil, W. Mackens, H. Voss, and J. Werther, editors, *Scientific Computing in Chemical Engineering*, pages 62–76. Springer-Verlag, 1999.

[115] T. Park and P.I. Barton. Implicit model checking of logic-based control systems. *AIChE Journal*, 43(9):2246–2260, 1997.

[116] B. Piccoli. Necessary conditions for hybrid optimization. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.

[117] B. Potočnik, A. Bemporad, F.D. Torrisi, G. Mušič, and B. Zupančič. Scheduling of hybrid systems: Multi product batch plant. In *Proc. IFAC World Congress*, Barcelona, 2002.

[118] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In *Chemical Process Control - V*, volume 93, no. 316, pages 232–256. AIChe Symposium Series - American Institute of Chemical Engineers, 1997.

[119] R. Raman and I.E. Grossmann. Relation between MILP modeling and logical inference for chemical process synthesis. *Computers Chem. Engng.*, 15(2):73–84, 1991.

[120] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. In *American Control Conference*, pages 1749–1753, Albuquerque, 1997.

[121] A. Rantzer and M. Johansson. Piecewise linear quadratic optimal control. *IEEE Trans. Automatic Control*, 45(4):629–637, April 2000.

[122] P. Riedinger, F.Kratz, C. Iung, and C.Zanne. Linear quadratic optimization for hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.

[123] J. Roll, A. Bemporad, and L. Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 2002. Conditionally accepted for publication as a regular paper.

[124] V. A. Roschchin, O.V. Volkovich, and I.V. Sergienko. Models and methods of solution of quadratic integer programming problems. *Cybernetics*, 23:289–305, 1987.

[125] N. V. Sahinidis. BARON — Branch And Reduce Optimization Navigator. Technical report, University of Illinois at Urbana-Champaign, Dept. of Chemical Engineering, Urbana, IL, USA, 2000.

[126] A.J. van der Schaft and J.M. Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer, London, 2000.

[127] B.I. Silva, O. Stursberg, B.H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 2867–2874, Orlando, Florida, December 2001.

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

[129] H.J. Sussmann. A maximum principle for hybrid optimal control problems. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.

[130] F. Torrisi, A. Bemporad, G. Bertini, P. Hertach, D. Jost, and Mignone D. Hysdel 2.0.5 - user manual. Technical Report AUT02-28, Automatic Control Laboratory, ETH Zurich, 2002.

[131] F.D. Torrisi and A. Bemporad. HYSDEL — A tool for generating computational hybrid models. *IEEE Transactions on Control Systems Technology*, 2002. Accepted for publication. `http://control.ethz.ch/~hybrid/hysdel`.

[132] F.D. Torrisi, A. Bemporad, and L. Giovanardi. Reach-set computation for analysis and optimal control of discrete hybrid automata. Technical report, Automatic Control Laboratory, ETH Zurich, 2003.

[133] M.L. Tyler and M. Morari. Propositional logic in control and monitoring problems. *Automatica*, 35(4):565–582, 1999.

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

[135] H.P. Williams. Logical problems and integer programming. *Bulletin of the Institute of Mathematics and Its Applications*, 13:18–20, 1977.

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

[137] H. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Trans. Automatic Control*, 11(2):161–167, 1966.

[138] X. Xu and P.J. Antsaklis. An approach to switched systems optimal control based on parameterization of the switching instants. In *Proc. IFAC World Congress*, Barcelona, Spain, 2002.

[139] X. Xu and P.J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, number 2623 in Lecture Notes in Computer Science, pages 540–555. Springer-Verlag, 2003.

[140] S. Yovine. KRONOS: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1–2):123–133, October 1997.

[141] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, New York, 1994.