Hierarchical Design of Piecewise Linear Hybrid Dynamical Systems Using a Control Regulator Approach*

Xenofon D. Koutsoukos
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
Tel. +1-650-812-4385
Fax +1-650-812-4334
koutsouk@parc.xerox.com

Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
Tel. +1-219-631-5792
Fax +1-219-631-4393
antsaklis.1@nd.edu

Abstract

In this paper, a novel methodology for hierarchical control design of piecewise linear hybrid dynamical systems is presented. The main characteristic of this class of hybrid systems is that the continuous dynamics are described by linear difference equations, the discrete dynamics by finite automata, and the interaction between the continuous and the discrete part is defined by piecewise linear maps. The regulator problem is formulated and algorithms for the synthesis of dynamical controllers are developed. Control specifications are modeled as finite automata. Both static specifications that do not change as time progresses and dynamic specifications that include sequencing of events and eventual execution of actions are considered. Control design is implemented using finite automata and linear programming techniques. Simulation results of a temperature control system are used to illustrate the approach.

1 Introduction

In this paper, new methodologies to solve important control problems in hybrid systems are presented. Our work is motivated by the need to address challenging problems in the control and coordination of modern complex engineering applications such as autonomous vehicles, chemical and manufacturing plants, and multiple robotic systems. A mathematical model that can capture both discrete and continuous phenomena is formulated. The continuous dynamics are described by linear time invariant difference equations and the discrete dynamics by finite automata. The interaction between the continuous and discrete parts is defined by piecewise linear maps characterized by sets of linear equalities and inequalities. We refer to this class of systems as *piecewise linear hybrid dynamical systems* in order to emphasize the hybrid nature of the systems and problems of interest. Piecewise linear hybrid dynamical systems is an important class of systems with many practical applications. This model is general enough to describe important engineering applications, but simple enough to facilitate the development of analysis, and more importantly, synthesis tools. Piecewise linear hybrid dynamical systems have an efficient representation for modeling and simulation.

*The partial financial support of the National Science Foundation (ECS99-12458) and the Army Research Office (DAAG55-98-1-0199) is gratefully acknowledged.

Furthermore, current modeling tools such as MATLAB,SIMULINK, and STATEFLOW offer the necessary flexibility for modeling and simulation of this class of systems.

In the model proposed in the present work, we restrict ourselves to linear continuous dynamics evolving in discrete-time. However, we also consider a plant that may also contain discrete dynamics. More importantly, we consider a larger class of inputs which may contain both discrete and continuous control inputs as well as discrete and continuous disturbances. The main advantage of the approach is that it provides a convenient general framework for both analysis and synthesis. In particular, the regulator problem for hybrid systems is formulated as a systematic methodology for hyrbid system controller design based on discrete abstractions of the continuous dynamics.

In order to analyze hybrid systems and design control algorithms, it is desirable to induce dynamical systems in finite quotient spaces that preserve the properties of interest and then study the simplified models. A systematic methodology for analysis of piecewise linear hybrid systems based on discrete abstractions of the continuous dynamics has been presented in [21]. In this paper, we briefly present the technical results from [21] (see Section 6) that are necessary for the formulation of the regulator problem. An important characteristic of the approach is that the available control inputs are taken into consideration in order to simplify the system. The main mathematical tool to be used is the *predecessor operator* applied recursively to subsets of the hybrid state space. The application of the predecessor operator corresponds to partition refinement into finer partitions that allow the formulation of conditions that guarantee the existence of appropriate controls for the objectives of interest.

Typical control specifications investigated in this paper are formulated in terms of partitions of the state space of the system. Examples include safety problems, where the controller guarantees that the plant will not enter an unsafe region, for example, guaranteeing that two interacting robots will not collide. Also reachability problems where the controller drives the plant from an initial operating region or state to a desired one; this is the case for example in the startup procedure of a chemical plant. Safety and reachability conditions are tested using efficient linear programming techniques. Safety and reachability specifications can be characterized as static specifications since they do not change as time progresses. In this paper, we also present a formal control design framework for dynamic specifications that involve sequencing of events and eventual execution of actions. In a manufacturing system, for example, the assembly of a component may require that a set of tasks is executed in a specific order while each task is satisfying safety specifications.

Our control design methodology is based on a formulation of the regulator problem for piecewise linear hybrid dynamical systems. In general, a regulator requests certain types of outputs from the plant and these are to be attained in the presence of disturbances. The desired outputs can be described as the outputs of another dynamical system, called the *exosystem*. We present a modeling formalism for the exosystem based on finite automata models. We consider both static and dynamic specifications. The main advantages of the formal modeling of the specifications are the following. First, formal modeling allows to compose complex specifications by simpler ones. The controller synthesis relies directly on the finite automaton model of the specifications. Therefore, by putting the emphasis on the specifications, we can develop a systematic methodology for controller synthesis. Furthermore, the formal models of the specifications are necessary for simulation and verification tools.

Our objective is to design a controller so that the closed loop system consisting of the plant and the controller exhibits the same behavior as the exosystem. In order to define the closed loop system, we model the constituent systems as *set-dynamical systems* [38]. The main question is if there exists a controller so that the closed loop system follows the behavior of the exosystem. This question is directly related to the existence of appropriate control resources in order for the plant to achieve the desired behavior. We formalize this notion using the *attainability* of the specified behavior. In this work, *attainable behavior refers to behavior that can be forced to the plant by a control mechanism.*

Based on the proposed notion of attainability for the desired behavior of piecewise linear hybrid systems, we present a systematic procedure for controller design. We present a convenient representation for the controller as a dynamical system which consists of three agents. (a) The *event generator* which receives the discrete-time measurement signal of the hybrid plant, and issues appropriate events when the state enters a new region of the state space. (b)The *control automaton* which is a finite automaton whose purpose is to select an appropriate cost functional based on the control objective. (c) The *actuator* that determines the control input which is applied to the hybrid plant by solving a set of linear programming problems at each time step. The plant and the exosystem are linked by a controller to form a regulator. A feedback controller can be designed to regulate the system. The main characteristic of the controller is that it contains a copy of the exosystem in " an appropriate sense" in accordance to the *internal model principle*. Simulation results are used to illustrate the proposed methodology using a temperature control system.

A great amount of research work has already been done in the hybrid systems area during the past decade; see for example [3] and the references there in. Piecewise linear systems arise very often as mathematical models for practical applications. For example, piecewise linear systems can be used to model systems with discontinuous dynamics that arise because of saturation constraints, hysteresis, friction in mechanical systems and so on. For another example, in order to avoid dealing directly with a set of nonlinear differential equations one may choose to work with linear equations and switch among these simpler models. Furthermore, piecewise linear systems arise in the switching control paradigm [28, 29] where the behavior of the plant is controlled by switching between different controllers for each region of the state space. It should be noted that the class of piecewise linear systems has been studied extensively in the circuit theory community; see for example [24] and the references therein. Here, we are interested in approaches that have been developed for modeling, analysis, and synthesis of hybrid control systems. The first investigations of piecewise linear hybrid systems can be found in [39, 40, 41]. The main problems studied in this framework were stability, controllability, and input-output regulation. Piecewise linear dynamical systems have been considered also in [9, 4, 5]. A methodology for approximating the reachable states is developed and a supervisory control framework is used for controller design. A class of hybrid systems which is similar to piecewise linear hybrid systems is considered in [6, 7, 8]. These systems are described by linear dynamic equations subject to linear inequalities involving real and integer variables. Piecewise linear systems were also studied in [16] to develop computational algorithms for the analysis of nonlinear and uncertain dynamical systems.

The hybrid system model used in this paper can be viewed as a input-output hybrid automaton evolving in discrete-time. Hybrid automata provide a general modeling formalism for the formal specification and algorithmic analysis of hybrid systems [1]. Formalisms for input/output hybrid automata have been also proposed in [27, 45, 25]. A related approach to the work presented in this paper is based on the modeling formalism of hybrid automata and uses bisimulations to study the decidability of verification algorithms [14, 23, 2]. Bisimulations are quotient systems that preserve the reachability properties of the original hybrid system and therefore, problems related to the reachability of the original system can be solved by studying the quotient system. The idea of using finite bisimulations for the analysis and synthesis of hybrid systems is similar to the approximation of the continuous dynamics with discrete event systems. Verification techniques of hybrid systems based on discrete approximations have been presented in [11], where the reachable sets are approximated by sequences of overlapping convex polygons.

The approach presented in this paper is directly related to supervisory control framework for hybrid systems [43, 22]. Similar approaches based on approximations of the continuous dynamics by a discrete event system have also been proposed in [32, 36, 12, 26]. The hybrid system model typically used in the supervisory control framework [22]. consists of a plant described by nonlinear differential or difference equations, a discrete event controller described by a deterministic finite automaton, and an interface which provides the means for the communication between the plant

and the controller. This type of supervisory control problems arise whenever a continuous system is to be controlled by a discrete process such as a switching mechanism or a digital computer program. Using the supervisory control framework, it has been possible to design discrete-event controllers for hybrid systems based on discrete abstractions of the continuous dynamics. Although supervisory control of hybrid systems has been applied successfully for an important class of hybrid systems, it has important limitations. This approach does not intend to address problems that involve continuous controls, as it has been assumed that any continuous control action has already been considered and is included in the plant. However, practical engineering applications usually involve interacting continuous and discrete control resources. Furthermore, the assumption that we can first design the continuous and discrete controllers separately usually leads to conservative design.

Supervisory control of hybrid systems is based on the fact that if undesirable behaviors can be eliminated from the DES plant then these behaviors can likewise be eliminated from the actual system. The uncontrolled DES plant model is assumed to generate "illegal behavior" that should be avoided by appropriate control action. On the other hand, just because a control policy permits a given behavior in the DES plant, there is no guarantee that that behavior will occur in the actual system. Therefore, the supervisory control framework is not suitable when we want to guarantee that the plant will achieve its goals and it will eventually execute the desired actions. Supervisory control can be used efficiently only when the objective is to restrict the behavior of the system by disabling undesirable events. The events are usually divided into two sets, those which are *controllable* and those which are *uncontrollable*. A plant symbol being controllable means that the supervisor can prevent it from being issued. When the supervisor prevents a controllable plant symbol from being issued, the plant symbol is said to be *disabled*. In the hybrid system case, we cannot guarantee that the system satisfies the control specification by only disabling events because the supervisor may be *blocking*. Consider, for example, a reachability specification between the regions R_1 and R_2. Using the supervisory control framework, the goal is to disable all the events that cause the state to exit from R_1 to a different region than R_2. However, this does not guarantee that the state will eventually reach the region R_2, since it is possible for the state to remain forever in R_1.

In order to take into consideration these limitations of the supervisory control framework for hybrid systems, we have formulated the hybrid systems regulator. The main difference is that for the regulator problem the control objective is for the closed loop system to follow the desired output which is assumed to be generated by another dynamical system. This framework leads naturally to an input-output representation of the constituent system which is more similar to classical control design than the supervisory control framework. Additionally, in order to guarantee the eventual execution of actions, a controller that can force events is taken into consideration. Events can be forced by selecting appropriately the control signal by solving an optimization problem at every time step.

The main contributions of the paper are the following. An algebraic system theoretical framework is developed for the analysis, verification, and synthesis of piecewise linear hybrid dynamical systems. This framework enables us to develop a novel methodology for control of piecewise linear hybrid systems based on discrete abstractions of the continuous dynamics. Algorithms for control design based on the regulator problem for discrete-time piecewise linear hybrid systems are presented in detail. It should be noted that these algorithms can be applied in the general case when the discrete dynamics contain controllable and uncontrollable events and the continuous dynamics contain control inputs and disturbances. The research contributions of this work impact the areas of reachability analysis, verification, and synthesis of piecewise linear hybrid systems. Note that the main results of this paper have appeared in [17]; early results have been reported in [19, 18, 20].

This paper is organized as follows. In Section 2, we present the modeling framework for discrete-time hybrid dynamical systems. Section 3 contains the necessary mathematical preliminaries that are used to formally define

piecewise linear hybrid dynamical systems. Our mathematical model is presented in Section 4 and is illustrated using a temperature control system. In Section 5, we describe the proposed hierarchical control architecture using an algebraic system theory framework and we formalize the conditions under which such a hierarchical scheme can be used for control design of hybrid systems. In Section 6, we present algorithms for backward reachability analysis of piecewise linear hybrid systems. The regulator problem for piecewise linear hybrid dynamical systems is formulated in Section 7. First, we model the control specifications using finite automata models, and then, we define the notion of attainability for the behaviors of interest and we show that if the desired behavior is attainable then there exists a controller which guarantees that the hybrid plant satisfies the specifications. The controller design methodology is described in Section 8 where we also demonstrate that the hybrid system regulator satisfies the internal model principle in "an appropriate sense". In Section 9, we present simulation results of a temperature control system to illustrate the validity of the design methodology. Finally, concluding remarks are presented in Section 10.

2 Discrete-time Hybrid Dynamical Systems

Hybrid systems are modeled by the discrete-time dynamical system

$$x(t+1) = f(q(t), x(t), u(t)) \tag{1}$$

$$q(t+1) = \delta(q(t), x(t), \sigma(t)) \tag{2}$$

$$y(t) = g(q(t), x(t)) \tag{3}$$

where

- $t \in \{0, 1, 2, \ldots\} \subset \Re$ is the time index,
- $x \in X \subseteq \Re^n$ is the continuous state,
- $q \in Q$ is the discrete state or *mode* of the system, where the set Q is assumed to be finite,
- $u \in U \subset \Re^m$ is the continuous input,
- $\sigma \in \Sigma$ are the input events,
- $y \in Y$ is the output of the hybrid system,
- $f : Q \times X \times U \to X$ is the continuous state transition function,
- $\delta : Q \times X \times \Sigma \to Q$ is the discrete state transition function, and
- $g : Q \times X \to Y$ is the output function.

Note that the key characteristic of the hybrid system model is the two-sided interaction between the continuous and discrete dynamics. Often, it is desirable to distinguish between controlled and uncontrolled inputs, and we may include in the continuous state transition function both continuous controls $u \in U$ and continuous disturbances $d \in D$. Furthermore, the set of input events can be written as $\Sigma = \Sigma_c \cup \Sigma_u$. The set Σ_c represents the *controllable* events which are associated with discrete state transitions which can be issued by a control mechanism. The set Σ_u contains the *uncontrollable* events generated by the environment. In our modeling framework, these events are viewed as discrete disturbances. Finally, in the case when the measurements are different from the outputs, a measurement set and a measurement function can be included in the system's description.

The dynamic evolution of the system is defined as follows. While the system is at mode (discrete state) q, the continuous state evolves according to the difference equation (1) driven by the control input $u(t)$. A change in the discrete state of the system can be caused by two type of events. First, an input event $\sigma(t)$ generated by either the controller or the environment. Second, an event $e(t)$ generated by the continuous dynamics when the continuous state enters a prescribed region of the continuous state space X. The events generated by the continuous dynamics are defined with respect to a partition of the continuous state space. The state space $X = \Re^n$ is partitioned into a finite number of regions. When the continuous state enters a new region, an event $e(t)$ is triggered and may cause a discrete (state) transition. For every region, there exists a set of feasible discrete transitions. Conversely, each discrete transition can take place only in a specific region of the state space, which is usually called the *guard* for this transition. These notions will be explicitly defined for piecewise-linear systems later in Section 4. The state transitions are synchronized by a clock. At every clock tick an event $\sigma(t)$ may be triggered and an event $e(t)$ caused by the continuous dynamics may occur. Therefore, every change in the state occurs synchronously to a clock. In many physical systems, however, events occur asynchronously at time instants that do not necessarily coincide with the clock ticks. Discrete-time systems can be used as approximations of physical processes. The approximation is based on the fact that events that occur asynchronously are detected in the next clock tick (using digital computers). In many situations, the discrepancy in the time instants of the event occurrences can be studied by considering continuous disturbances in the model.

Presently, we have focused on *piecewise-linear systems* [39, 41] to facilitate the development of analysis and synthesis tools. These systems arise when the state set and/or the input set are partitioned into regions described by linear equalities and inequalities and the dynamics at each region are described by linear (or affine) state transitions. Output and measurement maps can be defined also in a similar way. The class of piecewise-linear systems is quite general as it includes linear systems, finite state machines, and their interconnections. They can be used also in many instances as approximations of more general systems. In the following, we present some necessary background material in order to formally define the mathematical model of piecewise linear hybrid dynamical systems.

3 Preliminaries

In this section, we present some basic notions and the necessary notation that are used in the modeling formalism of piecewise linear hybrid dynamical systems.

A *piecewise-linear (PL) subset* [40] of a finite dimensional vector space V is the union of a finite number of sets defined by (finitely many) linear equations $f(x) = a$ and linear inequalities $f(x) > a$. A *PL relation* $R : X \to Y$ between PL sets is one whose graph is a PL set (as a subset of $X \times Y$). Similarly for a PL map. Equivalently, the map $f : X \to Y$ is PL if there exists a covering of X by PL subsets X_i such that the restrictions $f|X_i$ are all affine (linear + translation).

Consider the state space X and define the mapping $\pi : X \to 2^X$ from X into the power set of X. The mapping π defines an equivalence relation E_π on the set X in the natural way

$$x_1 \, E_\pi \, x_2 \;\text{ iff }\; \pi(x_1) = \pi(x_2).$$

The image of the mapping π is called the *quotient space* of X by E_π and is denoted by X/E_π. Adopting this notation we can write $\pi : X \to X/E_\pi$ where π is understood as the *projection* of X onto X/E_π. The mapping π generates a partition of the state set X into the equivalence classes of E_π and will be called *generator*.

More specifically, we are interested in the case when $X = \Re^n$ and the generator is defined by a set of hyperplanes. Note that such piecewise-linear regions arise in many practical applications. Consider the collection

$\{h_i\}_{i=1,2,\ldots,\ell}$, $h_i : \Re^n \to \Re$ of real-valued functions of the form $h_i(x) = g_i^T x - w_i$, where $g_i \in \Re^n$ and $w_i \in \Re$. Let

$$H_i = \ker(h_i) = \{x \in \Re^n : h_i(x) = g_i^T x - w_i = 0\} \tag{4}$$

and assume that H_i is an $(n-1)$-dimensional hyperplane ($\nabla h_i(x) = g_i^T \neq 0$). We define the function $\hat{h}_i : \Re^n \to \{-1, 0, 1\}$ by

$$\hat{h}_i(x) = \begin{cases} -1 & \text{if} \quad h_i(x) < 0 \\ 0 & \text{if} \quad h_i(x) = 0 \\ 1 & \text{if} \quad h_i(x) > 0 \end{cases} \tag{5}$$

Then, the generator is defined by $\pi(x) = [\hat{h}_1(x), \ldots, \hat{h}_\ell(x)]^T$. Although the generator has been defined as $\pi : \Re^n \to \{-1, 0, 1\}^\ell$ there is a bijection between $\{-1, 0, 1\}^\ell$ and the quotient set X/E_π (they are the same set). The quotient set can be represented as $X/E_\pi = \{P_i\},\ i = 1, \ldots, |\pi|$ where each P_i corresponds to a polyhedral region of $\Re^n$.

It is assumed that the partition defined by the mapping π is appropriate for extraction of important information for the system and it will be called the *primary partition*. The primary partition is determined by considering the regions which are used to describe the control specifications and the interaction between the continuous and discrete part of the open loop hybrid system.

Let $E(X)$ be the set of all equivalence relations on X. The set $E_P(X)$ of all equivalence relations on X induced by mappings $\pi : X \to X/E_\pi$ which are defined using finite collections of $(n-1)$-dimensional hyperplanes and thus, they separate the state space X into polyhedral equivalence classes, is a sublattice of the equivalence lattice $E(X)$ (a proof can be found in [21]), and will be called polyhedral equivalence lattice. Partition refinement is defined with respect to the order relation of the polyhedral equivalence lattice. A partition defined by the mapping π' is finer than the partition defined by π, if the induced equivalence relations considered as elements of the equivalence lattice satisfy the condition $E_{\pi'} \leq E_\pi$.

We are interested in characterizing the events that occur when the continuous state enters a new region of the state space. The set of events generated by the continuous dynamics is called the set of *plant events* and is denoted by E. Since our hybrid model evolves in discrete-time, the generator will not be able to identify the exact moment that a hypersurface is crossed. It identifies the first sample after a crossing has occurred. The sequence of time instants when plant events occur is given by the following equations:

$$\tau_e[0] = 0 \tag{6}$$

$$\tau_e[n] = \min\{t > \tau_e[n-1] : \exists i, h_i(x(t))h_i(x(\tau_e[n-1])) < 0\} \tag{7}$$

Each plant event is generated according to

$$e[n] = \ell\left(x(\tau_e[n]), x(\tau_e[n-1])\right) \tag{8}$$

$$e(t) = \begin{cases} e[n] & \text{if } t = \tau_e[n] \\ \epsilon \text{ (null)} & \text{otherwise} \end{cases} \tag{9}$$

where $\ell : X \times X \to E$ is a function labeling the plant events.

4 Piecewise Linear Hybrid Dynamical Systems

In the following, we define the class of *piecewise linear hybrid dynamical systems*. The main characteristic of this class is that the continuous dynamics are described by linear difference equations, the discrete dynamics by finite automata,

and the interaction between the continuous and the discrete part is defined by piecewise linear maps. The proposed modeling formalism separates the physical plant to be controlled from the control specifications and the controller. It provides the necessary mathematical tools to describe explicitly what control actions are available in order to influence the behavior of the plant. A very important consequence of these characteristics is that it is possible to define open loop and closed loop connections between the plant and the controller and try to exploit the advantages of feedback.

First, we formally describe the interaction between the discrete and continuous components of a piecewise linear hybrid system. For each discrete mode, we assign a region of the state space using the mapping

$$\text{inv} : Q \to 2^{X/E_\pi}. \tag{10}$$

The continuous state may evolve according to the difference equation determined by the discrete state q only if $x(t) \in \text{inv}(q)$. The regions $\text{inv}(q)$ are called *invariants*. It is assumed that the invariants are regions of the primary partition. These regions arise from the control specifications that do not allow certain modes in a region of the state space. They can also arise from discontinuities in the continuous dynamics when, for example, saturation or sign functions are used to model the physical processes. Note that in our modeling framework, the invariants do not necessarily correspond to disjoint regions of the state space. This is a realistic assumption, since many times in modeling of practical applications, it is not straightforward to assign a unique difference equation to each region of the state space. This is a task to be accomplished by the controller depending on the control specifications. An alternative way to describe the notion of invariants that will be useful in our analysis is by defining the set of feasible modes for each region of the primary partition. The *active mode set* is defined by the mapping

$$\text{act} : X/E_\pi \to 2^Q. \tag{11}$$

¿From the definition of the invariants and the active mode sets, it follows that for each discrete state $q \in Q$ and for each region of the primary partition $P \in X/E_\pi$ we have

$$P \in \text{inv}(q) \Leftrightarrow q \in \text{act}(P). \tag{12}$$

Definition 1 A *piecewise linear hybrid dynamical system* (*PLHDS*) is defined by

$$\begin{aligned} x(t+1) &= A_{q(t)}x(t) + B_{q(t)}u(t) + E_{q(t)}d(t) && (13)\\ q(t+1) &= \delta(q(t), \pi(x(t)), \sigma_c(t), \sigma_u(t)),\ q(t+1) \in \text{act}(\pi(x(t))) && (14)\\ y(t) &= g(q(t), x(t)) && (15) \end{aligned}$$

where $x(0) = x_0 \in \Re^n, q(0) = q_0 \in Q$ and

- $\pi : X \to X/E_\pi$ partitions the continuous state space $\Re^n$ into polyhedral equivalence classes,
- $\text{act} : X/E_\pi \to 2^Q$ defines the active mode set,
- $A_q \in \Re^{n\times n}, B_q \in \Re^{n\times m}$, and $E_q \in \Re^{n\times p}$ are the system matrices for the discrete state q,
- $\delta : Q \times E \times \Sigma_c \times \Sigma_u \to Q$ is the discrete state transition function, and
- $g : Q \times X \to Y$ is the output function which is assumed to be piecewise linear.

Assume that the current discrete state is q and that $q' \in \text{act}(\pi(x(t)))$ for some state $x(t) \in \Re^n$, then q' is a possible new state, and the transition $q \to q'$ (or (q, q')) may occur. Each feasible discrete state transition is associated either with a controllable event $\sigma_c \in \Sigma_c$ or an uncontrollable event $\sigma_u \in \Sigma_u$. A controllable event is issued by a control mechanism and forces the transition to occur. An uncontrollable event is generated by the environment and may also force a discrete state transition. As it is described in the previous definition, the discrete state transition function is assumed to be deterministic which means that for a given controllable or uncontrollable event the next discrete state can be uniquely determined. The *guard* $G(q, q')$ of the transition (q, q') is defined as the set of all states (q, x) such that $q' \in \text{act}(\pi(x(t)))$ and there exist controllable event $\sigma_c \in \Sigma_c$ such that $q' = \delta(q, \pi(x), \sigma_c, \sigma_u)$ for every uncontrollable event $\sigma_u \in \Sigma_u$. The guard of the transition describes the region of the hybrid state space where the transition can be forced to take place independently of the disturbances generated by the environment.

Remark It should be noted that the above model does not include jumps in the continuous state that may occur when certain state variables are discontinuously reset, for example, upon crossing a hyperplane. Jumps can be added in the modeling formalism described above and in the subsequent analysis if they can be represented by piecewise linear maps. However, the notation becomes tedious, and the ideas and methodologies presented harder to follow.

Example - Temperature Control System We present a temperature control system to illustrate the piecewise linear hybrid system model. An electrical analog of a temperature control system is used by considering the temperature being analogous to electric voltage, heat quantity to current, heat capacity to capacitance, and thermal resistance to electrical resistance. The system consists of a furnace that can be switched on and off. When the furnace is on, a continuous input controls the produced heat. The control objective is to control the temperature at a point of the system by applying the heat input at a different point.

When the furnace is on, the system is described by the electrical circuit shown in Figure 1. Let x_1 and x_2 denote the voltages across the capacitors C_1 and C_2 respectively. Suppose that the (voltages) temperatures x_1 and x_2 are to be controlled by changing the (current) heat input u, which takes values in the set $U \subset \Re$. The temperature x_2 is also affected by the temperature d of the environment which is modeled as a continuous disturbance.

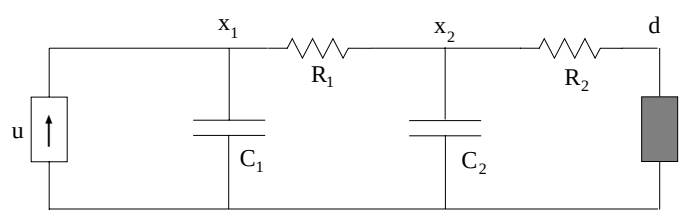

Figure 1: Electric circuit describing the case when the furnace is on.

When the furnace is turned off, the temperature is decreasing and the behavior of the system is described by the electrical circuit shown in Figure 2. The values of the resistors and the capacitors model the time constants of the system. The time constants are, in general, different depending on whether the temperature is increasing or decreasing.

The voltages (temperatures) x_1 and x_2 can be affected by either the continuous control input $u \in U$ or by switching on (mode q_1) or off (mode q_0) the furnace as illustrated in Figure 3. For example, we can consider a safety specification where the goal is to maintain the temperature x_2 between appropriate levels described by the tolerance interval $[lt, ht]$. A safety guard may be included in the system representation, so that the furnace is switched off automatically whenever the temperature x_1 exceeds a prescribed level M.

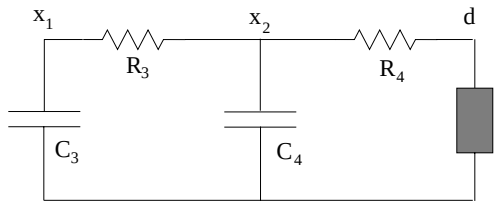

Figure 2: Electric circuit describing the case when the furnace is off.

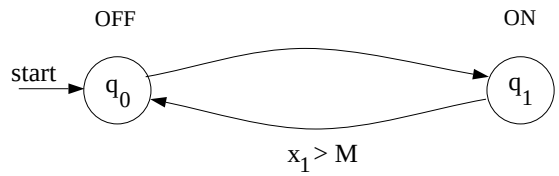

Figure 3: Temperature control system.

In the case when the system is on, the system is described by the state-space equation (using Kirchhoff's laws)

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{R_1C_1} & \frac{1}{R_1C_1} \\ \frac{1}{R_1C_2} & -\frac{1}{R_{12}C_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{1}{C_1} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ \frac{1}{R_2C_2} \end{bmatrix} d \tag{16}$$

where $R_{12} = \frac{R_1R_2}{R_1+R_2}$. When the system is off, then the state-space representation of the system takes the form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{R_3C_3} & \frac{1}{R_3C_3} \\ \frac{1}{R_3C_4} & -\frac{1}{R_{34}C_4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{R_4C_4} \end{bmatrix} d. \tag{17}$$

A partition of the continuous state space is obtained by considering the hyperplanes $h_1 = x_1 - M,\ h_2 = x_2 - lt,\ h_3 = x_2 - ht$, and $h_4 = x_1$ that describe the safety guard and the control specifications of the system. The partition of the continuous state space is shown in Figure 4. Discrete-time representations of the continuous dynamics for each mode are obtained using (zero-order hold) sampling. A piecewise linear hybrid dynamical system which models the temperature control system is described by the following equations:

$$x(t+1) = A_{q(t)}x(t) + B_{q(t)}u(t) + E_{q(t)}d(t),\ x_0 = x(0) \tag{18}$$

$$q(t+1) = \delta(q(t), \pi(x(t)), \sigma(t)),\ q_0 = q(0) \tag{19}$$

$$y(t) = x_2(t) \tag{20}$$

The discrete state transition function is described by the automaton of Figure 3. The transitions can be forced by controllable events issued by a control mechanism. The transition (q_1, q_0) may also occur because of the safety guard. The safety guard is described using the mapping act : $X/E_\pi \to 2^Q$ that defines the enabled transitions for each region of the primary partition. For the temperature control system, we have that the system cannot be at discrete state q_1 (on) if $x_1 > M$. The temperature control system example is used to illustrate the control design methodology in Section 7. □

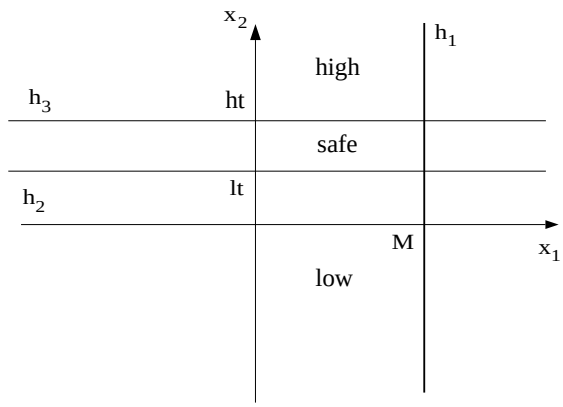

Figure 4: Partition for the temperature control system.

5 Hierarchical Control

In order to analyze hybrid systems and design control algorithms, it is desirable to induce dynamical systems in finite quotient spaces that preserve the properties of interest and then study the simplified models. The main mathematical tool to be used is the *predecessor operator* applied recursively to subsets of the hybrid state space. The application of the predecessor operator corresponds to the refinement of the primary partition into finer partitions that allow the formulation of conditions that guarantee the existence of appropriate controls for the objectives of interest.

In general, the design of the partition depends not only on the plant to be controlled, but also on the control policies available, as well as on the control goals to be attained. Certain control goals may require, for example, detailed feedback information while for others coarser quantization levels of the signals may be sufficient. The former case corresponds to finer partitioning of the feedback signal space, while the latter corresponds to coarser partitioning. The fact that different control goals may require different types of information about the plant is not surprising, as it is rather well known that to stabilize a system, for example, requires less detailed information about the system's dynamic behavior than to do tracking. Note that in general, the fewer the distinct regions in the partitioned signal space, the simpler (fewer states) the resulting induced system will be, and this will result in a simpler controller design. Since the systems to be controlled via hybrid controllers are typically complex, it is important to make every effort to use only the necessary information to attain the control goals. The question of systematically determining the minimum amount of information needed from the plant in order to achieve particular control goals is an important and largely open question; our work only partially resolves this question.

5.1 Induced Dynamical Systems

Let f be the state transition function of a dynamical system and assume that the inputs are fixed. Consider the diagram in Figure 5. Intuitively, the map π is used to coarsen the state set of the system. The question that arises is whether the system f can follow this abstraction. This question is concerned with the existence of a mapping $\tilde{f} : X/E_\pi \to X/E_\pi$ that makes the diagram commute. It is shown in [38] that $\tilde{f}$ exists if and only if

$$x_1 \, E_\pi \, x_2 \quad \Rightarrow \quad (\pi \circ f)(x_1) = (\pi \circ f)(x_2) \tag{21}$$

(where $\circ$ denotes function composition) and moreover, if (21) is satisfied then $\tilde{f}$ is unique. Note that the above result does not require any structure on the set X or the mappings π and f. Using equivalence relations on the state set

X, it is possible to define new dynamical systems in the derived quotient spaces. These systems are called *induced dynamical systems* [38].

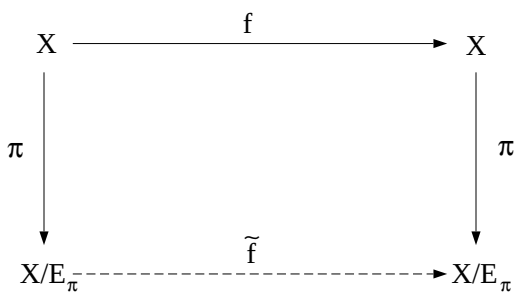

Figure 5: Induced dynamical systems.

In the hybrid systems case, the properties of the original system are not preserved, in general, in the induced system. One of the main difficulties arises because abstractions of continuous systems in finite quotient spaces usually result in nondeterniministic discrete event systems. Consider, for example, two continuous states $x_1, x_2 \in \Re^n,\ x_1 \neq x_2$ such that $\pi(x_1) = \pi(x_2) = P \in X/E_\pi$. The states x_1 and x_2 may be driven even using the same control input to different equivalence classes of the quotient space X/E_π, see Figure 6. Therefore, in general we have that $(\pi \circ f)(x_1) \neq (\pi \circ f)(x_2)$ and a mapping $\tilde{f}$ that makes the diagram commute does not exist. The induced system defined by the mapping $\tilde{f} : X/E_\pi \to X/E_\pi$ can be viewed as a nondeterministic system.

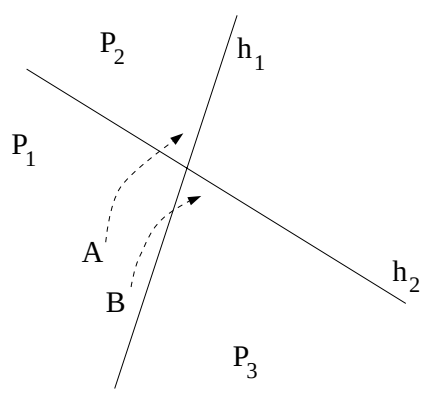

Figure 6: Nondeterminism of the induced dynamical system.

In general, piecewise linear hybrid dynamical systems cannot be induced in finite quotient spaces by preserving the reachability properties [23]. However, there are some cases when a mapping π and the induced system $\tilde{f}$ can be computed. A special case arise when the mapping π is defined using the natural invariants of the continuous dynamics [44]. However, it is very difficult to compute such partitions, and more importantly, the control specifications are not necessarily defined using the invariant sets of the system. It should be noted that the problem of abstracting continuous systems has been studied in [35] and controllability preserving quotient maps have been derived.

The solution we propose is to take advantage of the available control inputs in order to simplify the system. More specifically, we want to formulate conditions on the available control inputs in order to induce piecewise linear hybrid dynamical systems in finite quotient spaces. In order to illustrate our approach, we use the hierarchical architecture shown in Figure 7. The design of hybrid control systems is decomposed in two levels. In the higher level, we are concerned only with the existence of appropriate control inputs. The implementation of the controller and therefore,

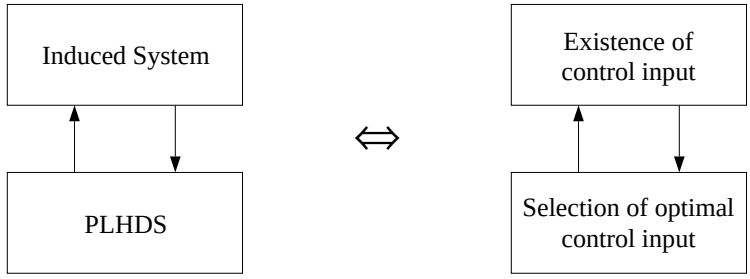

Figure 7: Hierarchical control of hybrid systems.

the selection of the control input signal is done by the lower level. First, we want to formulate efficient algorithms that guarantee the existence of appropriate control inputs for safety and reachability specifications. Second, we want to develop systematic methodologies for the design of the (lower level) controller.

The first problem has been presented in [21] where conditions for the existence of appropriate control inputs for safety and reachability specifications. The conditions are expressed as the feasibility of an optimization problem. In this paper, we are concerned with the second problem of selecting control inputs by solving appropriate optimization problems. The lower level problem is concerned with the selection of the optimal control inputs. A systematic design methodology for the selection of optimal control inputs that results in a feedback control architecture is presented. This control design methodology is formulated as the regulator problem for piecewise linear hybrid dynamical systems in Section 7.

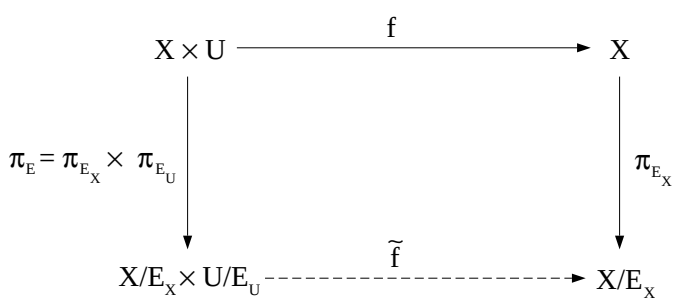

Figure 8: Function diagram including control inputs.

First, we describe our approach using an algebraic system theory setting. Consider the diagram shown in Figure 8. The equivalence relation E is defined by the mapping $\pi_E : X \times U \to X/E_X \times X/E_U$ as follows. The restriction of π_E in the state space X is the mapping which describes the primary partition of the system. The restriction of π_{E_U} separates the input space U into two equivalence classes. The first equivalence class consists of all control inputs available to the system and the second class consists of all the remaining elements of the input space. In practical applications, physical constraints such as saturation constraints restrict the control inputs that can be applied to the system. For example, the current input in the temperature control system example is constrained based on the available current source. Many times, we even consider a finite set of inputs corresponding to specific commands as, for example, in a valve can be closed, half open, open, and so on. Therefore, (x_1, u_1) is equivalent to (x_2, u_2) if and only if $\pi(x_1) = \pi(x_2)$ and the control inputs u_1, u_2 can be applied to the system. Note that the equivalence relation of the input space is defined in accordance with the hierarchical control architecture of Figure 7, since the higher control level is concerned only with the existence of controls. All available control inputs are equivalent at this level of abstraction.

The induced dynamical system $\tilde{f}$ exists if and only if

$$(x_1, u_1)\, E\, (x_2, u_2) \quad \Rightarrow \quad \exists u_1, u_2 \in U, (\pi_{E_X} \circ f)(x_1, u_1) = (\pi_{E_X} \circ f)(x_2, u_2). \tag{22}$$

The interpretation of the above condition is that $\tilde{f}$ exists if and only if there exist control inputs so that states that belong to the same polyhedral equivalence class of the primary partition, will remain equivalent in the next time step.

Of course, it is desirable to consider the dynamic evolution of the system in more than one steps. In order to do that, we consider an upper bound $N \in \mathbf{N}$ on the number of time steps that defines the length of the time horizon of interest. The length is assumed to be finite, since infinite-time problems in piecewise linear systems are, in general, undecidable [41]. We introduce the following notation.

$$\begin{aligned} [t_1, t_2] &= \{t_1, t_1 + 1, \ldots, t_2 - 1, t_2\},\ t_1 \leq t_2, \\ u^*[t_0, t_1] &= \{u(t_0), \ldots, u(t_1)\}. \end{aligned}$$

The function diagram in this case is shown in Figure 9.

$$\begin{tikzcd} X \times U^N \arrow[r, "\varphi"] \arrow[d, "\pi_E"'] & X \arrow[d, "\pi_{E_X}"] \\ X/E_X \times U^N/E_{U^N} \arrow[r, dashed, "\tilde{\varphi}"] & X/E_X \end{tikzcd}$$

Figure 9: Function diagram including control input sequences.

The equivalence relation on the input space is now defined as follows. The input sequences $u_1^*[t_0, t_1]$ and $u_2^*[t_0, t_2]$ are equivalent if and only if $u_1(t), t \in [t_0, t_1]$ and $u_2(t), t \in [t_0, t_2]$ are available control inputs, and in addition we have $t_1 - t_0 \leq N$ and $t_2 - t_0 \leq N$. The system mapping denoted by $\phi : X \times U^N \to X$ is the extension of the map $f : X \times U \to X$, so that it can be applied to sequence segments $u^*[t_0, t],\ t - t_0 \leq N$.

The induced dynamical system $\tilde{\phi}$ exists if and only if

$$(x_1, u_1^*[t_0, t_1])\, E\, (x_2, u_2^*[t_0, t_2]) \quad \Rightarrow \quad \exists u_1^*[t_0, t_1], u_2^*[t_0, t_2] \in U^N, (\pi_{E_X} \circ \phi)(x_1) = (\pi_{E_X} \circ \phi)(x_2). \tag{23}$$

Our objective is to compute a partition of the state space so that the diagram shown in Figure 9 commutes. Our approach is to refine the initial partition that is used to describe the specifications, until we can guarantee that there exist appropriate control resources that guarantee that the specifications are satisfied. Note that we consider only the regions of the state space that appear in the specifications. Consequently, the commutativity of the diagram in Figure 9 is only required with respect to the equivalence classes that are formed from the control specifications. Furthermore, for every region we want to find the optimal control sequence u^* with respected to a cost functional that depends on the region and the control specification. Finally, it is desirable to synthesize a new dynamical system that implements the hybrid controller connected to the plant in a feedback configuration that guarantees that the control specifications are satisfied.

6 Backward Reachability Analysis

In this section,we describe a backward reachability analysis approach for partition refinement. The main contribution is an efficient algorithm for partition refinement of piecewise linear hybrid systems based on the predecessor operator. A *region* of the state space is defined as $R \subset Q \times X$. We are interested in computing the set of all the states that can be driven to R by either continuous or discrete transitions. In the case of piecewise linear hybrid dynamical systems, it suffices to assume that the region is represented by $R = (q, P)$ where $q \in Q$ and $P \subset \Re^n$ is a piecewise linear set. The dynamic evolution of the system is defined by discrete and continuous transitions.

Discrete Transitions

The predecessor operator for discrete transitions is denoted by $\text{pre}_d : 2^{Q \times X} \to 2^{Q \times X}$ and it is used to compute the set of states that can be driven to the region R by a discrete instantaneous transition $q' \to q$ that can be forced by the controller for any uncontrollable event. The predecessor operator in this case is defined as follows:

$$\text{pre}_d(R) = \{(q', x) \in Q \times P | \exists \sigma_c \in \Sigma_c, \forall \sigma_u \in \Sigma_u, q = \delta(q', x, \sigma_c, \sigma_u)\}. \tag{24}$$

It should be noted that the null event is included in the event set $\Sigma = \Sigma_c \cup \Sigma_u$. A controllable and an uncontrollable event may occur at the same time instant. If only a controllable event occurs at a time instant, then the uncontrollable event at this time instant is assumed to be the null event ϵ and vice versa. Furthermore, by the definition of the PLHDS, for the transition $q' \to q$ to be feasible, it is required that $q' \in \text{act}(\pi(x))$.

The refinement partition algorithm consists of recursive applications of the predecessor operator starting with the regions defined by the primary partition. For every discrete transition that can be forced by a controllable event we have that

$$\text{pre}_d(R) = \bigcup_{q' \in \text{act}(P)} G(q', q) \tag{25}$$

where $R = (q, P)$ and $G(q', q)$ is the guard of transition $G(q', q)$. Since we assumed that the guards are described by the polyhedral equivalence classes of the primary partition, no refinement is necessary for the discrete transitions.

Continuous Transitions

In the case of continuous transitions, given the region $R = (q, P)$ we define the predecessor operator $\text{pre}_c : 2^{Q \times X} \to 2^{Q \times X}$ to compute the set of states for which there exists a control input so that the continuous state will be driven in the set P for every disturbance, while the system is at the discrete mode q. The action of the operator is described by

$$\text{pre}_c(R) = \{q\} \times \{x \in X | \exists u \in U, \forall d \in D, A_q x + B_q u + E_q d \in P\}. \tag{26}$$

The set $\text{pre}_c(R)$ is piecewise linear and can be always represented using only linear equalities and inequalities. Such a description is based on the fact that *piecewise-linear algebra* admits elimination of quantifiers [40]. In order to illustrate this result, we consider an alternative way to define PL sets [40].

Definition 2 Let $\mathcal{L}$ be the first-order language defined by (i) a set of (countably many) variables $\{x_1, x_2, \ldots\}$, (ii) the connective symbols $\neg$ and $\to$, (iii) the quantifier $\forall$, the parentheses (and) and the comma, (iv) A set of constants $\{r\}$ for each real number r, (v) A set of unary functions $\{r \cdot ()\}$ for each real number, the binary function $+$, (vi) the relational symbols $>$ and $=$.

Lemma 1 *[40] Every sentence in $\mathcal{L}$ defines a PL set and conversely, every PL subset of $\Re^n$ can be defined in this fashion.*

The conclusion of the above lemma is that any set defined using quantifiers can be also defined using only propositional connectives. Therefore, we can represent the predecessor operator of any piecewise linear region of the hybrid state space without quantifiers. The computation of the predecessor operator is based on combinations of three different mathematical tools. Fourier-Motzkin elimination [30] for computing appropriate projections, linear programming techniques [31] for eliminating redundant constraints, and equivalences from predicate logic [33] to combine the constraints. Consider the region $R = (q, P)$ where $q \in Q$ and P is a PL set. A PL set is not necessarily polyhedral. However, every PL set P can be written as

$$P = \bigcup_{i=1}^{p} P_i \tag{27}$$

where P_i are polyhedral sets, as shown in the following lemma. The proof of the lemma is constructive and is used in the developed algorithms for backward reachability analysis.

Lemma 2 *Every PL set can be written as a finite union of polyhedral sets.*

Proof By Lemma 1, every PL set can be written as a sentence in $\mathcal{L}$. Such a sentence is a logical formula without quantifiers and can be written in *disjunctive normal form* (see for example [33]) as

$$(\phi_{11} \wedge \phi_{12} \wedge \ldots) \vee \ldots \vee (\phi_{p1} \wedge \phi_{p2} \wedge \ldots) \tag{28}$$

where ϕ_{ij} is logical formula describing either a linear equality or inequality.

Consider a linear constraint ϕ_{ij} and assume without loss of generality that can be represented by the linear inequality $g_{ij}^T x < w_{ij}$. Then, equivalently we can represent ϕ_{ij} by the inequality $-g_{ij}^T x > -w_{ij}$. Therefore, every constraint in Equation (28) can be represented using, for example, the relational symbols $>$ and $=$. But every conjunction of linear constraints $(\phi_{i1} \wedge \phi_{i2} \wedge \ldots)$ is a polyhedral set as the intersection of halfspaces and hyperplanes. Therefore, every PL set can be written as a finite union of polyhedral sets by representing the set as a logical formula in disjunctive normal form. □

In order to simplify the notation, we consider only the restriction of the predecessor operator in the continuous state space $\text{pre}_c : 2^X \to 2^X$. It should be noted that in order to show that there exists a constructive algorithm for elimination of quantifiers, we have essentially to consider only the logical formula

$$(\exists u \in U)(\phi_{11}(x, u) \wedge \phi_{12}(x, u) \wedge \ldots) \vee \ldots \vee (\phi_{p1}(x, u) \wedge \phi_{p2}(x, u) \wedge \ldots). \tag{29}$$

Algorithms for elimination of quantifiers for more complicated logical formulas can then be derived using logical equivalences [10]. In the case of PLHDS, we are interested in elimination of quantifiers for formulas of the form $(\exists u \in U)$ and $(\forall d \in D)$ for the control inputs and disturbances respectively. By Lemma 4.2, it suffices to show how the predecessor operator is applied to a union of polyhedral sets. We compute the predecessor operator of a PL set in two steps. First, we consider only polyhedral sets and second, unions of polyhedral sets.

Consider the system

$$x(t+1) = Ax(t) + Bu(t) \tag{30}$$

where $A \in \Re^{n \times n}$ and $B \in \Re^{n \times m}$. It is assumed that the control input takes values in the polytope (bounded polyhedral) U described by

$$U = \{u \in \Re^m | Fu \leq v\}, \;\; F \in \Re^{\mu \times m}, \; v \in \Re^{\mu}. \tag{31}$$

Consider the polyhedral set $P \subseteq \Re^n$ given by

$$P = \{x \in \Re^n | Gx \leq w\}, \;\; G \in \Re^{\nu \times n}, \; w \in \Re^{\nu}. \tag{32}$$

Our objective is to present a systematic methodology to compute the predecessor operator set

$$\text{pre}_c(P) = \{x \in \Re^n | \exists u \in U, Ax + Bu \in P\}. \tag{33}$$

We denote $\text{Pr} : X \times U \rightarrow X$ the *projection* from the set $X \times U = \Re^n \times \Re^m$ to the state space $X = \Re^n$.

Proposition 1 *The set pre$_c(P)$ is given by*

$$\textit{pre}_c(P) = \text{Pr}(Q) \tag{34}$$

where $Q \subseteq X \times U$ is defined as

$$Q = \{(x, u) | (GAx + GBu \leq w) \wedge (Fu \leq v)\}. \tag{35}$$

Proof By direct substitution, we have that

$$\text{pre}_c(P) = \{x | \exists u \in U, GAx + GBu \leq w\} \tag{36}$$

Then, we have that if $x \in \text{Pr}(Q)$, there exits $u \in U$ such that $(x, u) \in Q$, and therefore $x \in \text{pre}_c(P)$. Conversely, if $x \in \text{pre}_c(P)$, then by definition of the predecessor operator there exists control input $u \in U$ such that $(x, u) \in Q$, which implies that $x \in \text{Pr}(Q)$. Therefore, we have shown that $\text{pre}_c(P) = \text{Pr}(Q)$. □

The projection of the set Q into the continuous state space $X = \Re^n$ can be computed using the *Fourier-Motzkin elimination method* [30, 13, 47]. We project the polyhedron $Q \subset \Re^n \times \Re^m$ into the space $\Re^n$ by eliminating the variables u_i of the control input vector. According to Fourier's method, in order to eliminate a variable from a set of inequalities, we must consider all pairs of inequalities in which the variable has opposite sign and eliminate the variable between each pair. Since U is bounded, all the control variables u_i will appear with opposite sign in at least one pair of inequalities from the constraints $Fu \leq v$. In order to see that, consider that there exists u_i that appear with the same sign in all the constraints. Assume without loss of generality that u_i appears with a positive sign in all the constraints $Fu \leq v$. Then, u_i can be decreased indefinitely without violating any of the constraints. Therefore, the set U is unbounded which is a contradiction. A piecewise linear set, however, is not necessarily polyhedral, but it can be written as the union of polyhedral sets using the proof of Lemma 4.2. Consider, the set $P = \bigcup_{i=1}^{p} P_i$ where P_i are polyhedral sets. Then, the set $\text{pre}_c(P)$ can be computed by the following lemma.

Lemma 3 *Consider the piecewise linear set $P = \bigcup_{i=1}^{p} P_i$, where P_i are polyhedral sets, then the predecessor operator of P can be computed by pre$_c(P) = \bigcup_{i=1}^{p}$ pre$_c(P_i)$.*

Proof

$$\begin{aligned}
\text{pre}_c(P') &= \text{pre}_c\left(\bigcup_{i=1}^{p} P_i\right) && (37)\\
&= \{x | \exists u \in U, Ax + Bu \in \bigcup_{i=1}^{p} P_i\} && (38)\\
&= \{x | \exists u \in U, Ax + Bu \in P_1 \vee \ldots \vee \exists u \in U, Ax + Bu \in P_p\} && (39)\\
&= \bigcup_{i=1}^{p} \text{pre}_c(P_i) && (40)
\end{aligned}$$

□

Therefore, the predecessor operator commutes with unions of piecewise linear sets. Note that this lemma is a consequence of the equivalence $(\exists x)(\phi(x) \vee \psi(x)) \leftrightarrow (\exists x)\phi(x) \vee (\exists x)\psi(x)$ in predicate logic.

Continuous Disturbances

Here, we consider that continuous disturbances are present in the description of the system which for a fixed discrete mode is given by

$$x(t+1) = Ax(t) + Bu(t) + Ed(t) \tag{41}$$

where $A \in \Re^{n\times n}, B \in \Re^{n\times m}$, and $E \in \Re^{n\times p}$. It is also assumed that the control input u and the disturbance d take values in the polyhedral and bounded sets U and D respectively.

Consider the polyhedral set P represented by the following set of linear inequalities:

$$\begin{array}{ccc} g_1^T x & \leq & w_1 \\ \vdots & & \vdots \\ g_\nu^T x & \leq & w_\nu \end{array}$$

In this case, the predecessor operator takes the form

$$\text{pre}_c(P) = \{x \in X | \exists u \in U, \forall d \in D, Ax + Bu + Ed \in P\}. \tag{42}$$

Consider the following linear programming problems:

$$\begin{array}{cc} \min & -g_i^T Ed \\ \text{s.t.} & d \in D \end{array} \tag{43}$$

Since D is a bounded set the above linear programming problems have finite solutions. The corresponding solutions are denoted by $d_i^* = \text{argmin}_{d\in D}\{-g_i^T Ed\}, i = 1, \ldots, \nu$.

Proposition 2 *The set pre$_c(P)$ is given by*

$$\textit{pre}_c(P) = \Pr(Q) \tag{44}$$

where $Q \subseteq X \times U$ is defined as

$$Q = \{(x,u) | \bigwedge_{i=1,\ldots,\nu} g_i^T Ax + g_i^T Bu \leq w_i - g_i^T Ed_i^*\}. \tag{45}$$

Proof If $x \in \text{pre}_c(P)$ then by definition there exists control input $u \in U$ such that the following set of inequalities holds for every $d \in D$, and therefore for $d = [d_1^*, \ldots, d_\nu^*]^T$ we have that

$$\begin{array}{ccc} g_1^T Ax + g_1^T Bu & \leq & w_1 - g^T Ed_1^* \\ \vdots & & \vdots \\ g_\nu^T Ax + g_\nu^T Bu & \leq & w_\nu - g_\nu^T Ed_\nu^* \end{array}$$

Therefore, there exists $u \in U$ such that $(x,u) \in Q$, which implies that $x \in \Pr(Q)$.

Conversely, assume that $x \in \Pr(Q)$ but $x \notin \text{pre}_c(P)$. Then, there exists $\bar{d} \in D$ and $i \in [1, \ldots, \nu]$ such that for every $u \in U$

$$g_i^T Ax + g_i^T Bu > w_i - g_i^T E\bar{d}. \tag{46}$$

But by the assumption that $x \in \mathrm{Pr}(Q)$ we have that there exists $u \in U$ such that

$$g_i^T Ax + g_i^T Bu \leq w_i - g_i^T Ed_i^* \leq w_i - g_i^T E\bar{d} \tag{47}$$

which is a contradiction. □

Note that we could first apply the Fourier-Motzkin elimination method for the elimination of control variables, and then solve the linear programming problems for the disturbance. In the case the set P is piecewise linear but not polyhedral, then we can can compute the set $\mathrm{pre}_c(P)$ without quantifiers by using appropriate equivalences from predicate logic. For example, in order to eliminate the universal quantifier of the disturbances for the set $P_1 \cup P_2$, we can use the logical equivalence

$$\forall d \in D, Ax + Bu + Ed \in P_1 \cup P_2 \leftrightarrow \neg(\exists d \in D, Ax + Bu + Ed \in P_1^c \cap P_2^c). \tag{48}$$

Then, the existential quantifier can be eliminated by writing the set $P_1^c \cap P_2^c$ in disjunctive normal form and apply the Fourier-Motzkin elimination method for each set of conjunctive constraints. Note that since the control variables $u \in U$ are assumed to be independent of the disturbance variable $d \in D$, we can select the order for the elimination of quantifiers.

Example In order to illustrate, that the predecessor operator can be computed in a closed-form in a straightforward manner, we consider a piecewise linear set described by the logical formula

$$(\phi_1(x,u,d) \wedge \phi_2(x,u,d)) \vee (\phi_3(x,u,d) \wedge \phi_4(x,u,d)) \tag{49}$$

where ϕ_i corresponds to the linear constraint $g_i^T Ax + g_i^T Bu + g_i^T Ed \leq w_i$.

The computation of the set $\mathrm{pre}_c(P)$ is equivalent to the quantifier elimination for the formula

$$(\exists u)(\forall d)(\phi_1(x,u,d) \wedge \phi_2(x,u,d)) \vee (\phi_3(x,u,d) \wedge \phi_4(x,u,d).$$

By applying simple logical equivalences we have

$$\begin{aligned}
& (\exists u)(\forall d)(\phi_1(x,u,d) \wedge \phi_2(x,u,d)) \vee (\phi_3(x,u,d) \wedge \phi_4(x,u,d)) \\
\Leftrightarrow \quad & (\forall d)\left((\exists u)(\phi_1(x,u,d) \wedge \phi_2(x,u,d)) \vee (\phi_3(x,u,d) \wedge \phi_4(x,u,d)\right) \\
\Leftrightarrow \quad & (\forall d)\left((\exists u)(\phi_1(x,u,d) \wedge \phi_2(x,u,d))\right) \vee (\exists u)(\phi_3(x,u,d) \wedge \phi_4(x,u,d))\,.
\end{aligned}$$

The elimination of the control variables can be accomplished by applying Fourier-Motzkin elimination. The resulting set can be written in disjunctive normal form to obtain the logical formula $\Psi(x,d)$. Then, the disturbance variables can be eliminated using the logical equivalence $(\forall d \in D)(\Psi(x,d)) \Leftrightarrow (\neg(\exists d \in D)\neg(\Psi(x,d))$. □

We have presented constructive algorithms for the computation of the predecessor operator for any piecewise linear region of the continuous state space. These algorithms use the Fourier-Motzkin elimination method, linear programming techniques, and simple equivalences from predicate logic. The algorithms were presented in analytical form and they can be implemented by software in a straightforward manner. These algorithms have been applied for reachability analysis of practical examples using MATLAB and are illustrated in this paper using the temperature control system.

Remark A special case of particular interest is the class of hybrid systems for which the control inputs take values in a finite set. This is a rather important class of systems since it can be used to model many practical applications.

For example, chemical processes usually involve valves and compressors that can be modeled using discrete variables. Discrete control inputs arise also in the motion control of many systems such as satellites or underwater vehicles. Note that in this case the projection $\mathrm{Pr}(Q)$ can be computed as the union of the sets that result by substituting each possible value for the control input. This method, however, will lead to many redundant constrains. The procedure to eliminate these redundant constrains requires additional computational effort. A methodology for reachability analysis in the case of discrete control inputs based on mathematical programming techniques has been presented in [20].

Algorithm for Backward Reachability Analysis

Consider a PLHDS described by the equations (13) - (15) and a region $R = (q, P)$. We denote the quotient space X/E_π induced by the primary partition as $X/E_\pi = \{P_i\}, i = 1, \ldots, |\pi|$. In addition, let $\text{pre}_{c,q} : 2^X \to 2^X$ denote the predecessor operator for a continuous transition described by the discrete mode q. The following algorithm computes all the states of the hybrid system that can be driven to R in one time-step. The algorithm is implemented using the technical results presented earlier in this section.

Algorithm for the computation of pre(R)

INPUT: $R = (q, P),\ S = \emptyset,\ T = \emptyset;$
for $i = 1, \ldots, |\pi|$
  $Q_i = P \cap P_i;$
  if $Q_i \neq \emptyset$
    for $q' \in \text{act}(P_i)$
      $S = S \cup \text{pre}_{c,q'}(Q_i);$
      $T = T \cup \{q'\};$
    end
end
OUTPUT: $\text{pre}(R) = (T, S)$

The algorithm computes all the regions of the state space for which the state can be driven to R by a continuous transition. In order to consider only the discrete modes that are feasible at each region of the state space, we write the set P as a union of regions of the initial partition. Then, we add also the states that can be driven to R by discrete transitions. If the initial region R contains more than one discrete states, then the algorithm is applied for each individual state. We have shown that the set $\text{pre}(R)$ is piecewise linear and is described using a finite set of linear inequalities. Therefore, we can apply the predecessor operator to compute the set of all states that can be driven to $\text{pre}(R)$ to get $\text{pre}(\text{pre}(R))$. Following the same procedure, we define successive applications of the predecessor operator as

$$\text{pre}^N(R) = \overbrace{\text{pre}(\cdots \text{pre}(R))}^{N \text{ times}}. \tag{50}$$

For a given region R, we define the *coreachable* set $CR(R)$ as the set of all states that can be driven to R. The coreachable set for a region of the hybrid state space can be computed by successive application of the predecessor operator

$$CR(R) = \text{pre}^*(R). \tag{51}$$

It should be noted that the algorithm for the computation of the coreachable set for a region R is semi-decidable. The procedure produces the correct answer if it terminates, but its termination is not guaranteed.

In this section, we have presented a systematic approach for the computation of the predecessor operator for piecewise linear hybrid dynamical systems. The developed algorithms can be used for backward reachability analysis and partition refinement. The predecessor operator can be computed in a closed form for any piecewise linear region of the hybrid state space. Here, we comment on the computational complexity of the presented algorithms for backward reachability analysis. Infinite time problems for piecewise linear systems are, in general, undecidable [41]. We have presented semi-decidable procedures for backward reachability analysis. For finite time problems, backward reachability algorithms for piecewise linear hybrid systems are NP-complete [41]. This follows from the definition of the predecessor operator which is formulated using the existential quantifier over all possible inputs. Practically, the number of linear constraints that are used to represent the coreachable region grows exponentially at every iteration of the algorithm. The developed algorithms can be used for practical applications if they involve only a reasonable number of iterations.

7 Hybrid system regulator

In this section, the regulator problem for hybrid systems is formulated. In general, a regulator requests certain types of outputs from the plant so that these are attained in the presence of disturbances. The desired outputs can be described as the outputs of another dynamical system, called the *exosystem*. We model the control specifications using an input-output deterministic finite automaton. We are interested on either static specifications that do not change as time progresses or dynamic specifications that include, for example, sequencing of events. The dynamic behavior of the exosystem is described by the set of output sequences it can generate. Our control objective is that the closed loop system consisting of the plant and the controller shown in Figure 10 exhibits the same behavior as the exosystem.

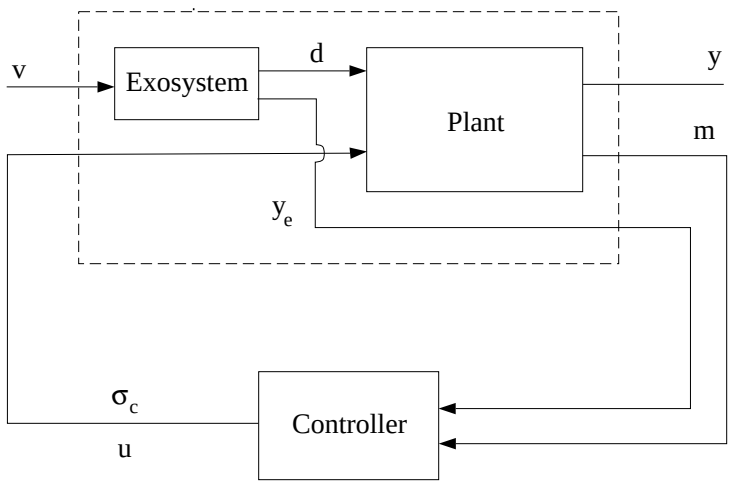

Figure 10: Hybrid system regulator.

In order to define the closed loop system, we model the constituent systems as *set-dynamical systems* [38]. The main question is if there exists a controller so that the closed loop system follows the behavior of the exosystem. This question is directly related to the existence of appropriate control resources in order for the plant to achieve the desired behavior. We formalize this notion using the *attainability* of the specified behavior. In this work, *attainable*

behavior refers to behavior that can be forced to the plant by a control mechanism. Based on the proposed notion of attainability for the desired behavior of piecewise linear hybrid systems, we present a systematic procedure for controller design. We present a convenient representation for the controller as a dynamical system which consists of three agents, the *event generator*,the *control automaton*, and the *actuator* described in detail later in this Section. The main characteristic of the controller is that it contains a copy of the exosystem "in an appropriate sense" in accordance to the *internal model principle*.

7.1 Control Specifications

In this section, we present a modeling formalism for control specifications based on finite automata models. We consider both static and dynamic specifications. Static specifications describe desired outputs that do not change as time progresses. For example, safety and reachability are static specifications. Dynamic specifications involve sequencing of events and eventual execution of actions. In a manufacturing system, for example, the assembly of a component may require that a set of tasks is executed in a specific order.

Remark Often, we are also interest in the real-time behavior of a system. For example, we may require that each component is assembled in a manufacturing system in less than 1 min. Real-time specifications can be incorporated into the framework by including explicitly timers in the model of the plant. The timers can be included either in the continuous part using linear oscillators [42] or in the discrete part using finite state machine models [34, 36]. Note that in the case when the timers are described by linear continuous models or finite state machines, the resulting system is still piecewise linear and our analysis and synthesis methodologies can be still applied. However, the disadvantage of such an approach is that the number of states for the exosystem and the controller becomes very large. In this work, we do not consider real-time specifications but we concentrate only on static and (discrete-event) dynamic specifications.

7.1.1 Safety

In the following, we focus on the safety problem and we show how the refinement of the state space partition can be used to formulated conditions for safety.

Definition 3 Given a set of safe states described by the region $R \subset Q \times X$ and an initial condition $(q_0, x_0) \in R$, we say that the system is *safe* if $(q(t), x(t)) \in R$ for every t.

Our first objective is to formulate conditions on the available controls, so that a given set is safe for a PLHDS. We formulate conditions that guarantee that a given region of the hybrid state space is safe. The conditions can be efficiently tested using linear programming techniques.

Theorem 1 *A PLHDS is safe with respect to the region $R \subseteq Q \times X$ if and only if $R \subseteq$ pre(R).*

Proof If $R \subseteq \text{pre}(R)$, every state $(q, x) \in \text{pre}(R)$ and therefore every state $(q, x) \in R$ can be driven in R, either by selection of appropriate control input $u \in U$ or by triggering a discrete transition and therefore, the system is safe.

Conversely, assume that the system is safe and consider there exists control policy such that $x(t) \in R$ for every t. By definition, the set pre(R) is the set of all the states for which there exists control policy so that the next state will be in R. Therefore, since the system is safe for every $x \in R$ we have that $x \in \text{pre}(R)$. $\square$

Furnace ON	*Furnace OFF*
q_1	q_0
$R_1 = 2$	$R_3 = 10$
$R_2 = 1$	$R_4 = 2$
$C_1 = 1$	$C_3 = 0.5$
$C_2 = 1$	$C_4 = 1$
$U_1 = [0.5, 5]$	$U_0 = 0$
$D_1 = [0, 1]$	$D_0 = [-1, 0]$

Table 1: Parameters for the temperature control system

In the following, we present a constructive algorithm which is used to test the condition $R \subseteq \text{pre}(R)$. Let $R|X$ and $\text{pre}(R)|X$ be the projection of R and $\text{pre}(R)$ into the continuous state space X. Similarly $R|Q$ and $\text{pre}(R)|Q$ for the discrete state space Q. Since, the sets $R|Q$ and $\text{pre}(R)|Q$ are finite, we can test whether $R|Q \subseteq \text{pre}(R)|Q$ in a straightforward manner. Next, we concentrate on the continuous part of the regions R and $\text{pre}(R)$. The sets $R|X$ and $\text{pre}(R)|X$ are piecewise linear but not polyhedral, and therefore they are not necessarily convex. In order to test whether $R|X \subseteq \text{pre}(R)|X$, we represent the constraints in disjunctive normal form (DNF) and we test the feasibility of finite set of linear programming problems.

Using Lemma 2, every PL set can be written as a union of polyhedral sets using the disjunctive normal form representation. Therefore, we can assume that the set $R|X$ and the complement of the set $\text{pre}(R)|X$ can be written as

$$R|X = \bigcup_{i=1,\ldots,|P|} P_i \tag{52}$$

and

$$[\text{pre}(R)|X]^c = \bigcup_{j=1,\ldots,|Q|} Q_j \tag{53}$$

where P_i and Q_j are polyhedral, and therefore convex sets in $\Re^n$. For each pair (i, j) the set $C_{ij} = P_i \cap Q_j$ is polyhedral as the intersection of polyhedral sets. Furthermore, the condition $P_i \cap Q_j = \emptyset$ can be tested by solving the following linear programming problem for an arbitrary cost vector $c \in \Re^n$:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in C_{ij} \end{array} \tag{54}$$

We have that $P_i \cap Q_j = \emptyset$ if and only if the above linear programming is infeasible. Therefore, we have that $R \subseteq \text{pre}(R)$ and the PLHDS is safe if and only if $P_i \cap Q_j = \emptyset$ for every $i = 1, \ldots, |P|$ and $j = 1, \ldots, |Q|$.

Example - Temperature Control System In the following, we use the temperature control system presented in Section 4 to illustrate how we can formulate the safety conditions. The temperature control system is modeled by the PLHDS described in Equations (18)- (20). We consider the system parameters shown in Table 1.

The discrete state q_1 corresponds to the case the furnace is on. Using zero-order hold sampling with $T = 1$, the continuous dynamics are described by the difference equation

$$x(t+1) = A_1 x(t) + B_1 u(t) + E_1 d(t) \tag{55}$$

where

$$A_1 = \begin{bmatrix} -0.6634 & 0.1997 \\ 0.1997 & 0.2641 \end{bmatrix}, \; B_1 = \begin{bmatrix} 0.8101 \\ 0.1369 \end{bmatrix}, \; E_1 = \begin{bmatrix} 0.1369 \\ 0.5363 \end{bmatrix}, \tag{56}$$

and $u(t) \in U_1, d(t) \in D_1$.

Similarly, for the discrete state q_0 (furnace off), we have

$$x(t+1) = A_0 x(t) + B_0 u(t) + E_0 d(t) \tag{57}$$

where

$$A_0 = \begin{bmatrix} 0.8259 & 0.1354 \\ 0.0677 & 0.5551 \end{bmatrix}, \; B_0 = \begin{bmatrix} 1.8179 \\ 0.0773 \end{bmatrix}, \; E_0 = \begin{bmatrix} 0.0387 \\ 0.3772 \end{bmatrix}, \tag{58}$$

and $u(t) \in U_0, d(t) \in D_0$.

The partition of the state space is obtained by considering the following hyperplanes

$$h_1(x) = x_1 - M, \; M = 20 \tag{59}$$

$$h_2(x) = x_2 - ht, \; ht = 5 \tag{60}$$

$$h_3(x) = x_2 - lt, \; lt = 0 \tag{61}$$

$$h_4(x) = x_1 \tag{62}$$

and it is shown in Figure 11.

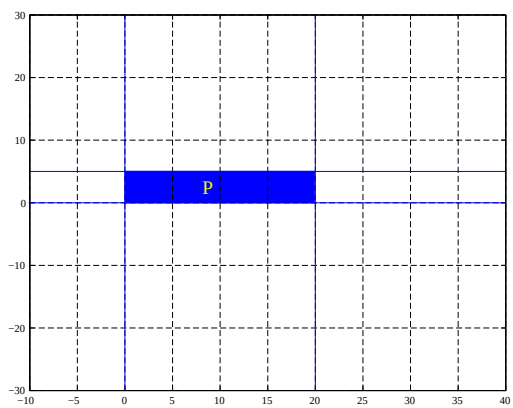

Figure 11: Primary partition for the temperature control system.

It is assumed that the safe region is described by the set $R = \{(q_0, q_1), P\}$ where P is given by

$$P = \{x \in \Re^2 | (0 \leq x_1 \leq M) \wedge (lt \leq x_2 \leq ht)\}. \tag{63}$$

Next, we describe in detail the algorithm for the computation of the set pre(R). We represent the set P as $P = \{x | Gx \leq w\}$ where

$$G = \begin{bmatrix} g_1^T \\ g_2^T \\ g_3^T \\ g_4^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \tag{64}$$

and

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 20 \\ 5 \\ 0 \\ 0 \end{bmatrix}. \tag{65}$$

First, we compute the set

$$\text{pre}_{c,q_0}(P) = \{x | A_0 x + B_0 u + E_0 d \in P\}. \tag{66}$$

Note that if the system is at mode q_0, the input is $u = 0$. Using Proposition 4.2, we consider the following set of linear inequalities:

$$GA_0 x \leq w - GE_0 d \tag{67}$$

We solve the linear programming problems

$$\begin{array}{ll} \min & -g_i^T E_0 d \\ \text{s.t.} & d \in D_0 \end{array} \tag{68}$$

for $i = 1, 2, 3, 4$ and we obtain $[d_1^*, d_2^*, d_3^*, d_4^*] = [0, 0, -1, -1]$. By substituting in Equation (67) we get

$$\text{pre}_{c,q_0}(P) = \left\{ x \in \Re^n | \begin{bmatrix} 0.8259 & 0.1354 \\ 0.0677 & 0.5551 \\ -0.0677 & -0.5551 \\ -0.8259 & -0.1354 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 20 \\ 5 \\ -0.3772 \\ -0.0387 \end{bmatrix} \right\}. \tag{69}$$

Next, we compute the set

$$\text{pre}_{c,q_1}(P) = \{x | A_1 x + B_1 u + E_1 d \in P\}. \tag{70}$$

We consider the following set of linear inequalities:

$$GA_1 x + GB_1 u \leq w - GE_1 d \tag{71}$$

$$u \leq 1 \tag{72}$$

$$-u \leq -0.5 \tag{73}$$

We apply the Fourier-Motzkin elimination method in order to eliminate the control variable u. We also solve the linear programming problems

$$\begin{array}{ll} \min & -g_i^T E_1 d \\ \text{s.t.} & d \in D_1 \end{array} \tag{74}$$

for $i = 1, 2, 3, 4$ and we obtain $[d_1^*, d_2^*, d_3^*, d_4^*] = [1, 1, 0, 0]$. Using Proposition 2 we have that

$$\text{pre}_{c,q_1}(P) = \left\{ x \in \Re^2 | \begin{bmatrix} 0.6634 & 0.1997 \\ 0.1997 & 0.2641 \\ -0.1997 & -0.2641 \\ -0.6634 & -0.1997 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 19.4580 \\ 4.3953 \\ 0.6847 \\ 4.0507 \end{bmatrix} \right\}. \tag{75}$$

The sets $\text{pre}_{c,q_0}(P)$ and $\text{pre}_{c,q_1}(P)$ are shown in Figure 12.

The set $\text{pre}(R)$ is computed using the algorithm presented in Subsection 6. Since both discrete transitions (q_0, q_1) and (q_1, q_0) are feasible we get

$$\text{pre}(R) = \{(q_0, q_1), \text{pre}_{c,q_0}(P) \cup \text{pre}_{c,q_1}(P)\}. \tag{76}$$

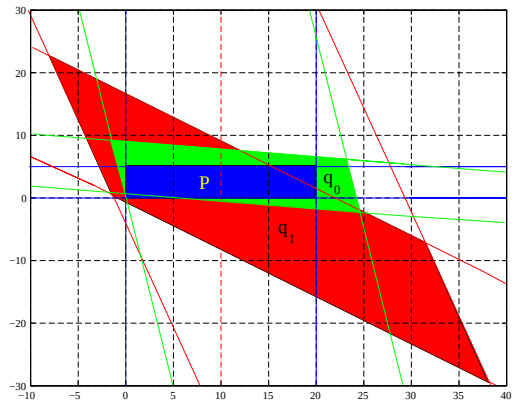

Figure 12: Final partition for the temperature control system.

In the following, we illustrate how we can test the safety condition $R \subseteq \text{pre}(R)$. The set $\text{pre}(R)|X$ can be represented by the logical formula

$$(\phi_{01} \wedge \phi_{02} \wedge \phi_{03} \wedge \phi_{04}) \vee (\phi_{11} \wedge \phi_{12} \wedge \phi_{13} \wedge \phi_{14}) \tag{77}$$

where the atomic formulas ϕ_{ij} correspond to the linear inequalities that define the sets $\text{pre}_{c,q_0}(P)$ and $\text{pre}_{c,q_1}(P)$ in Equations (69) and (75) respectively. We define the set $Q = [\text{pre}(R|X)]^c$. Using DeMorgan's laws, the set Q can be represented by

$$\begin{aligned}
& \neg[(\phi_{01} \wedge \phi_{02} \wedge \phi_{03} \wedge \phi_{04}) \vee (\phi_{11} \wedge \phi_{12} \wedge \phi_{13} \wedge \phi_{14})] \\
\Leftrightarrow \quad & \neg(\phi_{01} \wedge \phi_{02} \wedge \phi_{03} \wedge \phi_{04}) \wedge \neg(\phi_{11} \wedge \phi_{12} \wedge \phi_{13} \wedge \phi_{14}) \\
\Leftrightarrow \quad & (\neg\phi_{01} \vee \neg\phi_{02} \vee \neg\phi_{03} \vee \neg\phi_{04}) \wedge (\neg\phi_{11} \vee \neg\phi_{12} \vee \neg\phi_{13} \vee \neg\phi_{14}) \\
\Leftrightarrow \quad & \bigcup_{i=1,2,3,4,\ j=1,2,3,4} (\neg\phi_{0i} \wedge \neg\phi_{1j})
\end{aligned}$$

Therefore, the set Q can be written as $Q = \bigcup_{i,j} Q_{ij}$. Each set Q_{ij} is described by the logical formula $(\neg\phi_{0i} \wedge \neg\phi_{1j})$ and therefore, it is polyhedral. The condition $R|X \subseteq \text{pre}(R)|X$ can be checked by testing the feasibility of the linear programming problems:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in P \cap Q_{ij} \end{array} \tag{78}$$

For the temperature control system, we have that $R|Q = \text{pre}(R)|Q = \{q_0, q_1\}$ and $P \cap Q_{ij} = \emptyset$ for $i = 1, 2, 3, 4$ and $j = 1, 2, 3, 4$. Therefore, the region R is safe as it can be seen in Figure 12. □

7.1.2 Reachability

In this section, we study the reachability problem for piecewise linear hybrid dynamical systems. We present a reachability algorithm based on the successive computation of the predecessor operator. It should be emphasized that we are interested only in the case when reachability between two regions R_1 and R_2 is defined so that the state is driven to R_2 directly from the region R_1 without entering a third region. This is a problem of practical importance in hybrid systems since it is often desirable to drive the state to a target region of the state space while satisfying constraints on the state and input during the operation of the system. Consider, for example, an unmanned underwater vehicle with control policies that allow various combinations of screw speeds (on and off), stern plane positions (up, level, down),

and rudder positions (left, right, straight). A control goal for such a system can be described by a target region of the state space which represents a desirable set of displacements and velocities for the vehicle. However, while the system is driven to the target regions the dispacements and velocities must be appropriety constrained to guarantee safe operation.

Definition 4 Given two regions $R_1, R_2 \subseteq Q \times X$, we say that R_2 is *directly reachable* from R_1, if every state $(q, x) \in R_1$ can be driven in R_2 in finite time without entering a third region.

The problem of deciding if a region R_2 is directly reachable from R_1 can be solved by recursively computing all the states that can be driven to R_2 from R_1 using the predecessor operator. We only consider regions of the form $R_1 = (Q_1, P_1)$ and $R_2 = (Q_2, P_2)$ for which P_1 and P_2 are adjacent polyhedral regions of the primary partition. In this case, the regions P_1 and P_2 have a common boundary which is represented by a $(n-1)$-dimensional hyperplane $h(x) = g^T x - w$. The reachability problem between any two regions can be solved by finding a path consisting of adjacent reachable regions. Note that if the regions $R_1 = (q_1, P)$ and $R_2 = (q_2, P)$ have identical continuous parts, then the reachability problem can be solved by considering the set of feasible transitions for the polyhedral region P.

Consider the regions R_1 and R_2 and the initial state $(q, x) \in R_1$ and assume that we can disable the state from crossing all the boundaries of R_1 but $h(x)$. It is still possible that the hybrid system will be blocked in the sense that the state will never exit the region R_1 through the hyperplane $h(x)$. Note that this can happen since we want to drive the state from R_1 to R_2 without entering a third region. In this case there is a trade-off between driving the state into the target region and satisfying the constraints for the state trajectory. The risk of violating the operational conditions of a system while stirring the state to a desired operating point must be addressed. Thus, this formulation of the reachability problem that takes into consideration constraints in the state trajectory is more important than considering only the state into the target region, both in theory and in practice.

Our approach is based on conditions that guarantee that state can be forced to cross the hyperplane $h(x)$ in finite time by selecting appropriate controls. For this purpose, we consider a finite time horizon defined by NT where T is the sampling period and $N \in \mathbf{N}$. Consider a PLHDS described by the equations (13)-(15) and assume that the initial condition is $(q(t_0), x(t_0)) \in R_1$.

Definition 5 The region R_2 is *directly N-reachable* from R_1 if for every initial state $(q(t_0), x(t_0)) \in R_1$ there exist control inputs for the PLHDS and $k \in \mathbf{N}, 0 < k \leq N$ so that $(q(t), x(t)) \in R_1$ for $t_0 \leq t < t_0 + kt$ and $(q(t_0 + kt), x(t_0 + kt)) \in R_2$..

We define the *coreachable set* $CR^N_{R_1}(R_2)$ of all states that can be driven from the region R_1 to R_2 in the finite time $t \leq NT$ without entering a third region. The predecessor operator pre : $2^{Q \times X} \to 2^{Q \times X}$ can be used to compute the set $CR^N_{R_1}(R_2)$ using the following algorithm.

Algorithm for the computation of $CR^N_{R_1}(R_2)$

$R^0 = R_2$;
$CR^N_{R_1}(R_2) = \emptyset$;
$k = 0$;
while $\neg(R^{k+1} \subseteq R^k)$ AND $k \leq N$
    $R^{k+1} = \text{pre}(R^k) \cap R_1$;
    $CR^N_{R_1}(R_2) = CR^N_{R_1}(R_2) \cup R^{k+1}$;

$k = k + 1;$

end

Given the regions R_1 and R_2, we compute all the states that can driven from R_1 to R_2. Note that at every iteration k of the algorithm we consider the intersection of the set pre(R^k) with the set R_1 since we are interested only in states that can be driven to R_2 directly from the region R_1 without entering a third region. At every iteration of the algorithm we have to apply the predecessor operator to a piecewise linear region of the state space. The resulting region is still piecewise linear, it can be represented using only linear equalities and inequalities, and it can be computed using the algorithms for elimination of quantifiers presented in Section 6. The above algorithm can be used to determine if the region R_2 is N-reachable from R_1 using the following theorem.

Theorem 2 *Consider a PLHDS described by (13)-(14) and the regions $R_1 = (Q_1, P_1)$ and $R_2 = (Q_2, P_2)$. Then, the set set $CR^N_{R_1}(R_2)$ is piecewise linear and the region R_2 is directly N-reachable from R_1 if and only if $R_1 \subseteq CR^N_{R_1}(R_2)$.*

Proof The set set $CR^N_{R_1}(R_2)$ is piecewise linear since it is computed using finite unions and intersections of piecewise linear sets. At the k iteration of the algorithm, the set R^k contains all the states in R_1 that can be driven in R_2 in $t \leq kT$. If $R_1 \subseteq CR^N_{R_1}(R_2)$ then there exists controls so that every state $(q, x) \in R_1$ can be driven to R_2 in $t \leq NT$ without entering a third region. □

Furthermore, since the set $CR^N_{R_1}(R_2)$ is piecewise linear, the reachability problem between R_1 and R_2 can be solved using linear programming techniques similarly to the safety conditions. For regions that are not adjacent, a feasible path connecting these regions consisting of adjacent regions must be established. Note that this can be done at the higher level of abstraction, since the necessary information is the existence of a control policy and not the actual policy.

Example - Temperature Control System We illustrate the reachability algorithm using the temperature control system.

Consider the regions $R_1 = (\{q_0, q_1\}, P_1)$ and $R_2 = (\{q_0, q_1\}, P_2)$ where

$$P_1 = \{x \in \Re^2 | (0 \leq x_1 \leq 20) \wedge (-20 \leq x_2 \leq 0)\} \tag{79}$$

and

$$P_2 = \{x \in \Re^2 | (0 \leq x_1 \leq 20) \wedge (0 \leq x_2 \leq 5)\}. \tag{80}$$

It is desirable that every state from R_1 can be driven to R_2 without entering a third region. Such a specification may arise, for example, at the startup procedure of the system, where it is required for the state of the system to reach the safe region $0 \leq x_2 \leq 5$ without entering the unsafe region $x_1 > 20$.

In order to compute the set of states in the region R_1 that can be driven to R_2 using appropriate control inputs, we apply the reachability algorithm. The coreachable set of states for three iterations of the algorithm is shown in Figure 13.

The region R_2 is directly reachable from R_1 in $t = 3T$. Therefore, there exists control policy which selects the control input $u \in U$ and possibly forces appropriate discrete transitions so that every state $(q, x) \in R_1$ can be driven to the region R_2. □

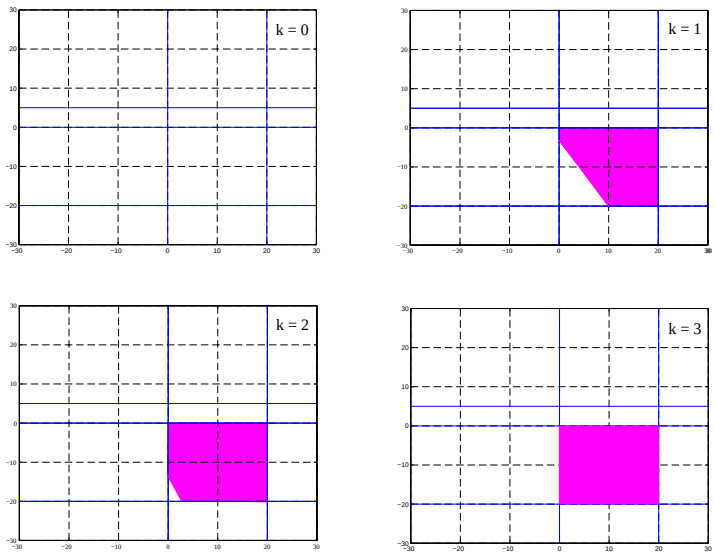

Figure 13: Coreachable set for the temperature control system.

Remark In order to guarantee that the reachability algorithm will always terminate we have also formulated a practical termination condition based on quantization of the state space. The basic idea is that the algorithm should terminate if the set pre(R^k) is not "substantially different" from the set pre(R^{k-L}). By "substantially different" we mean whether new cells of the quantized space have been added to the set of states that can be driven to R. L is a parameter selected by the designer and depends on the sampling period and the quantization of the state space. This grid-approximation method is used only in the termination condition, and therefore there is no accumulation of error due to the approximation; details can be found in [21].

7.1.3 Dynamic specifications

The main advantages of the formal modeling of the specifications are the following. First, formal modeling allows to compose complex specifications by simpler ones. The controller synthesis relies directly on the finite automaton model of the specifications. Therefore, by putting the emphasis on the specifications, we can develop a systematic methodology for controller synthesis. Finally, the formal models of the specifications are necessary for simulation and verification tools.

As it was discussed in Section 4, the primary partition is designed based on the specifications. We consider specifications that are described with respect to regions of the hybrid state space. We define the set X_e as

$$X_e = \{R_1, R_2, \ldots, R_M\} \tag{81}$$

where $R_i = (Q_i, P_i)$ are piecewise linear regions of the hybrid state space.

Since we assume that the primary partition is fine enough to describe the specifications, for every region we can write $R_i \subseteq Q \times X/E_\pi$. Note that in order to reduce the number of states of the finite automaton that models the specifications, we assume that each region R_i may contain more than one discrete modes and/or more than one regions of the continuous state space if those are adjacent. It is assumed that the regions R_i are disjoint as subsets of the hybrid state space $Q \times X$. Therefore, each state (q, x) corresponds to exactly one region R_i.

Example

Consider the temperature control system presented in Section 4. The set of discrete states is $Q = \{q_0, q_1\}$. The primary partition of the continuous state space is defined by the hyperplanes

$$h_1(x) = x_1 - M \tag{82}$$

$$h_2(x) = x_2 - lt \tag{83}$$

$$h_3(x) = x_2 - ht \tag{84}$$

$$h_4(x) = x_1 \tag{85}$$

and it is shown in Figure 14.

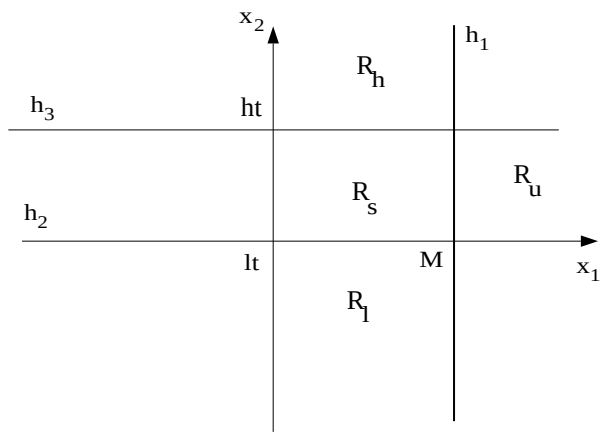

Figure 14: Primary partition for the temperature control system.

As it was described in Section 4, we have an unsafe region describing that the system cannot be on (q_1) while $x_1 > M$. For this specification we consider the region of the state space defined as

$$R_u = \left(\{q_1\}, \{x \in \Re^2 | x_1 > M\}\right). \tag{86}$$

Note that the continuous part $\{x \in \Re^2 | x_1 > M\}$ is defined with respect to the primary partition. However, it corresponds to a union of elements from the quotient set X/E_π.

Next, consider a safety specification for the system, for which we require that $lt \leq x_2 \leq ht$ during the operation of the system. This is a safety specification described by the region

$$R_s = \left(\{q_0, q_1\}, \{x \in \Re^2 | (0 \leq x_1 \leq M) \wedge (lt \leq x_2 \leq ht)\}\right). \tag{87}$$

Dynamic specifications involve also sequencing of events. For example, in the startup procedure for the temperature control system, we may require first to drive the system to a high temperature and then in the safe region. We describe the additional regions as

$$R_l = \left(\{q_0, q_1\}, \{x \in \Re^2 | (0 \leq x_1 \leq M) \wedge (x_2 < lt)\}\right) \tag{88}$$

and

$$R_h = \left(\{q_0, q_1\}, \{x \in \Re^2 | (0 \leq x_1 \leq M) \wedge (x_2 > ht)\}\right). \tag{89}$$

Then, the specification for the startup procedure is described by the sequence $R_l \to R_s \to R_h \to R_s$.

Further, we may require that during the operation of the system the temperature follows the periodic behavior $R_l \to R_s \to R_h \to R_s \to R_l$. □

In the following, we use a formal automaton model to represent the specifications of interest.

Definition 6 The control specifications are modeled by an input-output (I/O) deterministic finite automaton described by

$$\mathcal{E} = (X_e, V_e, Y_e, \delta_e, \lambda_e, R_0) \tag{90}$$

where

- X_e is the set of states,
- V_e is the input alphabet,
- Y_e is the output alphabet,
- $\delta_e : X_e \times V_e \to X_e$ is the state transition function,
- $\lambda_e : X_e \to Y_e$ is the output function returning the output associated with each state, and
- R_0 is the initial state.

We assume that the function δ_e is *non-total*, which means that not every input can be applied to every state of the automaton. We also assume that every state is reachable and therefore, there exists appropriate input sequences so that every state can be reached. The I/O finite automaton which describes the specifications is a deterministic Moore automaton [15] and is called the *exosystem* [38]. An I/O framework for modeling the specifications for hybrid systems arises in a natural manner as in classical control design and it can be more appropriate for the design of hybrid systems than the supervisory control framework [22].

Our objective is to control the plant so that it has the same behavior as the exosystem. In order to describe the dynamic behavior of the exosystem, a notion of time must be included in the system's representation. Following [38], this is accomplished by introducing an *index set* J equipped with a simple order relation and an *index function* $\alpha : \mathbf{N} \to J$.

An index function is said to be *admissible* if

1. α is *order preserving*, that is if $n_1 \leq n_2 \Rightarrow \alpha(n_1) \leq \alpha(n_2)$, and
2. α is *injective*, that is $n_1 \neq n_2 \Rightarrow \alpha(n_1) \neq \alpha(n_2)$.

Given the pair (α, J) with α admissible, denote $I = Im\alpha$ the image of the function α. The dynamic evolution of the system is defined as follows. The state $x_e \in X_e$ is associated with an index $j(n) \in Im\alpha$ meaning the state at "time" $j(n)$. Similarly, we associate an index with the input and output of the exosystem. The transition function δ_e leads to the "next" state where the interpretation is taken with respect to the index function α, and we write $x_e(j(n+1)) = \delta_e(x_e(j(n)), v_e(j(n)))$. Similarly, the output function represents the "current" output and we write $y_e(j(n)) = \lambda_e(x_e(j(n)))$. By abuse of notation, we can abbreviate the index and we can write

$$x_e[n+1] = \delta_e(x_e[n], v_e[n]) \tag{91}$$

$$y_e[n] = \lambda_e(x_e[n]). \tag{92}$$

We are only interested on either static specifications or dynamic specifications that include sequencing of events and not in real-time specifications. Therefore, we consider that the index n advances only when a state transition occurs and consequently, the index set J represents the ordering of the input sequence. In the following, we formally define the notion of behavior for the exosystem. A *dynamical system* can be described as a triple (T, W, B) where T is the time axis, W is the signal space, and $B \subset W^T$ (the set of all functions $f : T \to W$) the *behavior* [46].

The exosystem is modeled as a dynamical system. Its behavior is described by $B_e \subseteq (V_e \times Y_e)^I$ and consists of all the input-output pairs that the exosystem can generate. The "time" axis is described by the index set I defined above. A specification is described by the behavior $B_{sp} \subset B_e$. Therefore, B_{sp} consists of input-output sequences representing the desired behavior. Since we assume that the exosystem is represented by a deterministic finite automaton, for each input sequence we can obtain a unique output sequence. Furthermore, given the output sequence we can reconstruct the input sequence. Therefore, it is reasonable to assume that the specification is defined only with respect to the output set, and is given by $B_{sp} \subset Y_e^I$. The dynamic behavior of the exosystem can be described by the set of sequences of symbols from Y_e that it can generate. Denote by Y^* the set of all strings formed by concatenation of symbols from the output alphabet Y; the * operation is called the Kleene closure. A *language* is formally defined as a subset of Y^*. The behavior of the exosystem can be described by the output language $L \subset Y^*$.

The usual set operations, such as union, intersection, difference, complement (with respect to Y^*) are applicable to languages. In addition, the *prefix-closure* of L, denoted by $\bar{L}$ is defined as the set of all prefixes of strings in L. The language L is said to be *prefix-closed* if all the prefixes of the language are also in L, or equivalently if $L = \bar{L}$. Note that the language describing the output behavior is different than the language *accepted* by an automaton. The former is defined with respect to the output symbols generated by the output function, while the latter is defined with respect to the input symbols and the state transition function [15]. The language generated by the exosystem is defined as follows.

Definition 7 Given the finite set Y and the sequence $y : I \to Y$, then $y \in L$ if and only if there exist $x_e \in X_e$ and $v_e \in V_e$ such that the following conditions hold:

$$x_e[n+1] = \delta_e(x_e[n], v_e[n]), \quad \forall n \in I \tag{93}$$

$$y[n] = \lambda_e(x_e[n]), \forall n \in I \tag{94}$$

Each specification behavior can be described by a language $K \subset L$. Such a language may contain two types of symbols. First, it may contain *terminating* output symbols that represent safe regions of the state space. If the hybrid system reaches a terminating region then it remains in the state until the safety specification is not of interest. Note that this blocking behavior describes the safety of a region and it is not undesirable as in discrete event systems. Second, the language K may contain *non-terminating* output symbols. The transition from one state to the next represents reachability for the corresponding regions of the hybrid state space.

Example

The specifications for the temperature control system are modeled by the finite automaton shown in Figure 15. The state transition function can be obtained from the Figure 15. The output function is defined as follows:

$$\begin{aligned} \lambda(R_l) &= a \\ \lambda(R_s) &= b \\ \lambda(R_h) &= c \\ \lambda(R_u) &= d \end{aligned}$$

We consider the following specifications that can be described using the exosystem shown in Figure 15:

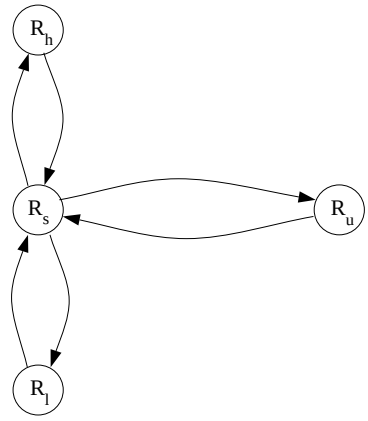

Figure 15: Exosystem for the temperature control system.

- For safety the desired behavior is described by the language $K_1 = b$, where b is viewed here as a constant function from I to Y_e.
- During the startup procedure the desired behavior can be described by the language $K_2 = abcb$. This a piecewise constant function from I to Y_e.
- During the operation of the system we may require a periodic behavior described by the language $K_3 = \overline{(abcb)^*}$.

□

The inputs for the finite automaton of the temperature control system correspond to control commands. The control objective is for the closed loop system to follow the output of the exosystem for a given input sequence. Note that these inputs are not actual inputs for the plant. However, there are certain cases where a subset of the inputs of the exosystem may appear as inputs for the discrete part of the plant. These inputs represent events generated by the environment and they are included to describe how we want the system to react to external events. Note that we still assume that the exosystem is deterministic.

7.2 Attainability and the Regulator Problem

Our control objective is that the closed loop system consisting of the plant and the controller shown in Figure 10 exhibits the same behavior as the exosystem. Note that by convention it is assumed that the disturbances and the events generated by the environment can be described as outputs of the exosystem. Next, we formally define the composition of the plant with the controller and the behavior of interest of the closed loop system.

In order to define the closed loop system we model the constituent systems as *set-dynamical systems*. A *set-dynamical system* (SDS) is denoted as $(X, U, Y; f, g)$ where X is the state set of the system, U is the input set, Y is the output set, $f : X \times U \to X$ is the state transition function, and $g : X \times U \to Y$ is the output function. It is important to distinguish between the controlled and the uncontrolled inputs (disturbances) of an SDS. Furthermore, in the case when the measurements are different than the outputs, a measurement set M and a measurement function m can be included in the system's description. In order to describe the behavior of a dynamical system, the notion of time must be included in the system's representation and this is accomplished with an index set J and an index function α.

Plant

The plant is a piecewise linear hybrid dynamical system described by Equations (13)-(15) and can be represented by the set-dynamical system

$$\mathcal{P} = (X_p, U_p, Y_p, M_p; f_p, g_p, m_p) \tag{95}$$

where

- $X_p = Q \times X$ is the hybrid state space.
- $U_p = (\Sigma_u \times \Sigma_c) \times (D \times U)$ is the input set. Σ_u is the set of events generated by the environment and Σ_c the set of events generated by the controller; the controller events are used to force a discrete transition (q, q') in the plant under the assumption that this transition is feasible ($q \in$ act$(\pi(x))$); D is the set of continuous disturbances and U is the set of continuous control inputs.
- $Y_p = \{R_1, \ldots, R_M\}$ is the output set consisting of the regions of the hybrid state space which are used to describe the control specifications.
- $M_p = Q \times X$ is the measurement set.
- $f_p : X_p \times U_p \to X_p$ is the hybrid state transition function described by the following equations:

$$x(t+1) = A_{q(t)}x(t) + B_{q(t)}u(t) + E_{q(t)}d(t) \tag{96}$$

$$q(t+1) = \delta(q(t), \pi(x(t)), \sigma_c(t), \sigma_u(t)),\ q(t+1) \in \text{act}(\pi(x(t))) \tag{97}$$

- $g_p : X_p \to Y_p$ is the output function; the output function is implemented by a filter that determines the membership of the state into a region R_i of the state space.
- $m_p : X_p \to M_p$ is the measurement function; it is assumed that the state is measurable, and the function m_p can be viewed as the identity function.

The evolution of the plant is defined in discrete-time. Therefore, it is assumed that the index set J describes real-time which is discretized using a sampling period T. The index function α is used to assign the time instant $t = nT$ to the index n.

Controller

The controller is represented by the SDS

$$\mathcal{C} = (X_c, U_c, Y_c; f_c, g_c) \tag{98}$$

where

- X_c is the set of states,
- $U_c = M_p$ is the input set which coincides with the measurement set of the plant,
- $Y_c = \Sigma \times U$ is the output of the controller consisting of the continuous control input and the controller events that may trigger a (feasible) discrete transition,
- $f_c : X_c \times U_c \to X_c$ is the state transition function, and

- $g_c : X_c \times U_c \to U_c$ is the output function of the controller.

In order to define the composition of the plant and the controller, it is reasonable to assume that the controller also evolves in discretized real-time.

Composition

The plant and the controller are connected in a feedback configuration as shown in Figure 10. The closed loop system is described by

$$\mathcal{CL} = (X_{cl}, U_{cl}, Y_{cl}; f_{cl}, g_{cl}) \tag{99}$$

where

- $X_{cl} = X_p \times X_c$ is the state set of the closed loop system,
- $U_{cl} = \Sigma_u \times D$ is the set of exogenous inputs,
- $Y_{cl} = Y_p$ is the output set,
- $f_{cl} : X_{cl} \times U_{cl} \to X_{cl}$ is the state transition function, and
- $g_{cl} : X_{cl} \times U_{cl} \to Y_{cl}$ is the output function of the closed loop system.

Since the control requirement is for the closed loop system to exhibit the same (output) behavior as the exosystem, we define the behavior of the plant with respect to the output set $Y_{cl} = \{R_1, \ldots . R_M\}$. The time-axis of the closed loop system coincides with the time-axis of the plant. Therefore, the closed loop system is a discrete-time dynamical system. In order to compare the behaviors of the exosystem and the closed loop system, we abstract the time information by defining a new index function α_{cl} for the closed loop system. Consider the index $k \in \mathbf{N}$. The state of the discrete-time dynamical system is associated with the index k meaning the state at time $t = kT$. Then define $\alpha_{cl} : \mathbf{N} \to I$ as follows:

$$\alpha_{cl}[0] = 0 \tag{100}$$

$$\alpha_{cl}[n] = \min\{t = kT > \alpha_{cl}[n-1] : y_{cl}(t) \neq y_{cl}(\alpha_{cl}[n-1])\} \tag{101}$$

We denote $I = \mathrm{Im}\alpha_{cl}$ the image of the function α_{cl}. Then the behavior of the closed loop system is defined as $B_{cl} \subseteq Y_{cl}^I$. As usually, we abbreviate the index $\alpha_{cl}[n]$ by n.

Definition 8 Given the exosystem $\mathcal{E}$ and the piecewise linear hybrid dynamical system $\mathcal{P}$, we say that the *regulator problem* has a solution if there exists a controller $\mathcal{C}$ such that the output behavior of the closed loop system follows the specified behavior of the exosystem, that is $B_{cl} = B_{sp}$.

The main question is if there exists a controller so that the closed loop system follows the behavior of the exosystem. This questions is directly related to the existence of appropriate control resources in order for the plant to achieve the behavior B_{sp}. We formalize this notion using the *attainability* of the specification behavior B_{sp}. In this work, *attainable behavior refers to behavior that can be forced to the plant by a control mechanism*.

We represent a sequence of symbols from the specification language K by

$$y = y_0 y_1 \ldots y_{n-1} \in K \tag{102}$$

where the index of the individual symbols satisfy equations (100)-(101).

Definition 9 The specification behavior B_{sp} is *attainable* if and only it is described by a language K that satisfies the following conditions:

(i) $y_0 = g(q_0, x_0) = g(q(0), x(0))$;

(ii) if y_{n-1} is a terminating output symbol, then for every $t > \alpha_{cl}[n-1]$ there exist control input $u(t)$ and controllable events $\sigma_c(t)$ so that for every disturbance $d(t)$ and uncontrollable event $\sigma_u(t)$, we have that

$$y(t) = g(q(t), x(t)) = y_{n-1}. \tag{103}$$

(iii) If y_{n-1} is not a terminating output symbol, then there exists $t' > \alpha_{cl}[n-1]$ so that for every t such that $\alpha_{cl}[n-1] < t < t'$, there exist $u(t)$ and $\sigma(t)$ such that for every $d(t)$ and $\sigma_u(t)$ we have

$$yy_n \in K, \;\; \alpha_{cl}[n] = t' \tag{104}$$

where

$$y(t') = g(q(t'), x(t')) = y_n \tag{105}$$

$$y(t) = g(q(t), x(t)) = y_{n-1}, \; \alpha_{cl}[n-1] < t < t'. \tag{106}$$

The above definition requires that for every prefix of the desired behavior, there exist controls, that will force the output of the system to remain in B_{sp}. The definition of attainability implies the next technical result for the existence of a controller that guarantees that the control specifications are satisfied.

Theorem 3 *The specification behavior B_{sp} is attainable if and only if the following conditions hold:*

(i) *Every terminating state y_{n-1} corresponds to a region R_{n-1} that is safe, and*

(ii) *For every non-terminating state y_{n-1}, there exists y_n so that, for the corresponding regions we have that R_n is reachable from R_{n-1}.*

Furthermore, if B_{sp} is attainable then there exists a controller $\mathcal{C}$ so that the regulator problem has a solution.

Proof We have

(i) $y_0 = g(q_0, x_0) \in K$ by the definition of attainability.

(ii) If y_{n-1} is a terminating output symbol, consider $R_{n-1} \subset Q \times X$ the corresponding region of the state space. Then, the attainability of the language K implies that the region R_{n-1} is safe and by the definition of safety, there exists a control policy that will force the state to remain in R_{n-1} for every disturbance.

(iii) If y_{n-1} is not a terminating output symbol, consider the regions R_{n-1} and R_n corresponding to the output symbols y_{n-1} and y_n respectively. Since K is attainable, there exist a control policy so that R_n is reachable from R_{n-1}, and therefore there exist a control policy so that the sequence $y = y_0 y_1 \ldots y_{n-1} y_n \in K$.

□

8 Controller Design

In this section, we present a systematic procedure for controller design. It is assumed that the desired behavior is attainable and therefore there exists a control policy so that the plant will follow the output of the exosystem. A controller is designed as a dynamical system to implement the desired control policy. We have already shown that if the specification behavior is attainable, the there exists control policy so that the closed loop system will satisfy the specification. Our objective is to build a convenient representation of the controller. The design of the controller is based on the regions $\{R_1, \ldots, R_M\}$ that are used to define the control specifications. The proposed representation for the controller is shown in Figure 16.

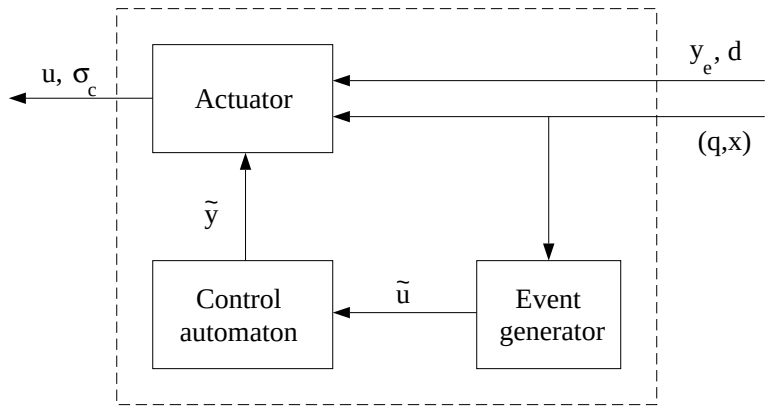

Figure 16: Controller diagram.

The controller consists of three agents. The *event generator* receives the discrete-time measurement signal of the hybrid plant, and issues appropriate events when the state $(q(t), x(t))$ enters a new region R_i of the hybrid state space. The *control automaton* is a finite automaton whose states correspond to the regions R_i and its main purpose is to select an appropriate cost functional based on the control objective. Finally, the *actuator* determines the control input which is applied to the hybrid plant. The control input consists of a continuous component $u \in U$ and a discrete component $\sigma_c \in \Sigma_c$ which triggers feasible discrete transitions. In the following, we formally define the controller.

Event Generator

The event generator abstracts the discrete-time signal $(q(t), x(t))$ from the plant to a sequence of events that describe the membership of the state to the regions R_i. The event generator is defined by the following equations:

$$\begin{aligned}
\alpha_c[0] &= 0 \\
\alpha_c[n] &= \min\{t > \alpha_c[n-1] : (q(t), x(t)) \in R_j \neq R_i \ni (q(\alpha_c[n-1]), x(\alpha_c[n-1]))\} \\
\widetilde{u}[n] &= \ell_c(R_i, R_j)
\end{aligned}$$

where $\alpha_c : \mathbf{N} \to I$ is an index function representing the order of events and $\ell_c : X_e \times X_e \to \widetilde{U}$ a (non-total) label function assigning an event to every pair of regions with adjacent continuous parts.

Control Automaton

The control automaton is an I/O deterministic (Moore) finite automaton [15] defined as

$$(X_c, \widetilde{U}, \widetilde{Y}, \widetilde{\delta}, \widetilde{\lambda})$$

where

- $X_c = \{R_1, \ldots, R_M\}$ is the set of states,
- $\widetilde{U}$ is the set of input events,
- $\widetilde{Y} = \{\widetilde{y}_1, \ldots, \widetilde{y}_M\}$ is the set of output events,
- $\widetilde{\delta} : X_c \times \widetilde{U} \to X_c$ is the state transition function, and
- $\widetilde{\lambda} : X_c \to \widetilde{Y}$ is the output function.

The control automaton is deterministic and therefore, the next state can be uniquely determined from the current state and the input event which is an abstraction of the state of the hybrid plant. This is a realistic assumption for practical applications of hybrid systems. The input events represent the measurements from the hybrid plant. An event $\widetilde{u}$ is generated when the state crosses to a new region R_i of the state space.

The Actuator

The actuator determines the control input to be applied to the plant using an optimization algorithm based on the desired output provided by the exosystem. The output of the actuator is a discrete-time control signal $(\sigma_c(t), u(t))$. At every time step, the control input is selected as the solution to a mathematical programming problem. In the following, we formulate the optimization problem that is used by the actuator. Consider the specification behavior described by the language $K = y_0 y_1 \ldots y_{n-1},\ y_i \in Y_e$ and let R_i be the corresponding region to the output symbol y_i.

First, we consider terminating output symbols that represent safety conditions for the corresponding region of the state space. Let y_{n-1} be a terminating state and $R_{n-1} = (S_{n-1}, P_{n-1}) \subset Q \times X$ the corresponding region. We define the cost functional $J_{n-1} : Q \times \Re^n \times \Re^m \times \Re^p \to \Re$

$$J_{n-1}(q, x, u, d) = c^T [A_{q(t)} x(t) + B_{q(t)} u(t) + E_{q(t)} d(t) - \bar{x}_{n-1}] \tag{107}$$

where $\bar{x}_{n-1} \in P_{n-1}$ is a fixed point selected by the designer and $c \in \Re^n$ a cost vector. For example, if P_{n-1} is polyhedral, $\bar{x}_{n-1}$ can be selected as the epicenter of P_{n-1}. The control signal is selected as the solution to the following optimization problem:

$$\begin{array}{cc} \min & J_{n-1}(q, x, u, d) \\ q \in \text{act}(\pi(x(t))) & \\ u \in U & \text{s.t. } A_q x(t) + B_q u(t) + E_q d(t) \in P_{n-1} \end{array} \tag{108}$$

At every time step, the constraint can be computed by substituting the state $x(t)$ and the disturbance $d(t)$. The above problem can be solved very efficiently by solving a linear programming problem for each feasible discrete mode $q \in \text{act}(\pi(x(t)))$. Let q' be the mode that corresponds to the minimum cost, then the control input is selected as $(\sigma_c(t), u(t))$ where $q' = \delta(q(t), \pi(x(t)), \sigma_c(t), \epsilon)$ and $u = \text{argmin}_{u \in U} J_{n-1}(q', x, u, d)$.

Next, we consider two non-terminating output symbols y_k and y_{k+1} which describe a reachability specification between the regions $R_k = (S_k, P_k)$ and $R_{k+1} = (S_{k+1}, P_{k+1})$. The control objective is to drive every state in R_k to R_{k+1}. As it is explained in Section 6, we can assume that P_k and P_{k+1} are adjacent polyhedral regions of the state space and we denote $h(x) = g^T x - w$ their common boundary. Let P be the polyhedral set defined by all the linear constraints that define P_k except $h(x)$. It is assumed without loss of generality that $h(x) > 0$ for every $x \in P$. We define the cost functional $J_h : Q \times \Re^n \times \Re^m \times \Re^p \to \Re$

$$J_h(q, x, u, d) = g^T[A_{q(t)}x(t) + B_{q(t)}u(t) + E_{q(t)}d(t)] - w. \tag{109}$$

The control signal is selected as the solution to the following optimization problem:

$$\begin{array}{cc} \min & J_h(q, x, u, d) \\ q \in \text{act}(\pi(x(t))) & \\ u \in U & \text{s.t. } A_q x(t) + B_q u(t) + E_q d(t) \in P \end{array} \tag{110}$$

Similarly, this problem can be solved very efficiently by solving a linear programming problem for each feasible discrete mode $q \in \text{act}(\pi(x(t)))$, Let q' be the mode that corresponds to the minimum cost, then the control input is selected as $(\sigma_c(t), u(t))$ where $q' = \delta(q(t), \pi(x(t)), \sigma_c(t), \epsilon)$ and $u = \text{argmin}_{u \in U} J_h(q', x, u, d)$.

The action of the actuator can be described by the following algorithm by combining terminating and non-terminating output symbols.

Optimization algorithm for the actuator

$t = t_0$;
while $t < \infty$
    INPUT: $y_i, \widetilde{y}, (q, x), d$;
    while $y_i \neq \widetilde{y}$
        $h(x) = g^T x - w$;
        if $h(x) \leq 0$ then exit;
        for $q \in \text{act}(\pi(x(t))$
            $J_h^q = \min_{u \in U} J_h(q, x, u, d)$;
        end
        $J_h^* = \min_{q \in \text{act}(\pi(x(t)))} J_h^q$;
        $q' = \text{argmin}(J_h^*)$;
        $u = \text{argmin}_{u \in U} J_h^*$;
    end
    if $y_i = \widetilde{y}$
        for $q \in \text{act}(\pi(x(t))$
            $J_i^q = \min_{u \in U} J_i(q, x, u, d)$;
        end
        $J_i^* = \min_{q \in \text{act}(\pi(x(t)))} J_i^q$;
        $q' = \text{argmin}(J_i^*)$;
        $u = \text{argmin}_{u \in U} J_i^*$;
    end
$t = t + 1$;
end

Corollary 1 *Consider the controller shown in Figure 16 with the event generator, control automaton, and actuator as defined above. If the specification behavior is attainable, the output behavior of the closed loop system follows the specified behavior of the exosystem, that is* $B_{cl} = B_{sp}$.

Proof Each terminating output symbol y_i represents a safety specification for the region R_i. By the attainability assumption, there exists a control policy that guarantees safety. Therefore, there exists a solution to the optimization problem (108) and the corresponding control input satisfies the safety objective.

Similarly, for non-terminating output symbols y_k and y_{k+1}. By the attainability assumption, there exists a control policy that guarantees that the region R_{k+1} is reachable from R_k. Therefore, there exists a solution to the optimization problem (110) and the corresponding control input satisfies the reachability objective. □

8.1 Internal Model Principle

The main characteristic of the controller is that it contains a copy of the dynamical action of the exosystem. This characteristic is known as the *internal model principle*. In this section, we will show that when the hybrid system regulator problem has a solution, then the controller designed in Section 8 satisfies the internal model principle in "an appropriate sense". The following discussion is based on the formulation of the internal model principle presented in [38].

The regulator is defined as the interconnection of the exosystem, the plant, and the controller as shown in Figure 10. The regulator can be represented by the SDS

$$\mathcal{R} = (S, Y^2; f, g) \tag{111}$$

where

- $S = X_e \times X_p \times X_c$ is the state set,
- $Y^2 = Y_e \times Y_p$ is the output set,
- $f : S \to S$ is the state transition function, and
- $g : S \to Y^2$ is the output function.

The functions f and g are derived by the composition of the exosystem, the plant, and the controller in a straightforward manner. We also consider that the system is autonomous and therefore there are not any exogenous inputs.

In order to demonstrate that the regulator problem satisfies the internal model principle, we need to find a convenient representation of the dynamic actions of the exosystem and the controller in the dynamical system $\mathcal{R}$. First, we consider the exosystem described by the finite automaton $\mathcal{E}$. Observe that the states of the exosystem correspond to piecewise linear regions of the state space $Q \times X$ of the plant. The main difficulty for representing the dynamic action of the exosystem as it appears in the regulator is that the regulator is a dynamical system evolving in discrete-time, while the exosystem is a discrete (event) dynamical system. Here, we are interested in the discrete (event) dynamic action of the exosystem and we represent this action using the induced continuous dynamics in the finite quotient space X/E_π. We define the set $\widetilde{S} = X_e \times \widetilde{X}_p \times X_c$ where $\widetilde{X}_p = Q \times X/E_\pi$ and we consider the state transition function $\widetilde{f} : \widetilde{S} \to \widetilde{S}$ by abstracting the continuous dynamics as described in Section 5. Note that by abuse of notation we use the quotient space X/E_π, although it is possible to group regions of the primary partition together in order to reduce

the number of states of the exosystem and the controller. The regulation condition can be stated as $X_d \subset \widetilde{S}$ where X_d is understood as the inverse image under the output function g of the regulator. Then, the dynamic action of the exosystem in the regulator can be described by $\widetilde{f} : X_d \to X_d$.

The controller is also a dynamical system that evolves in discrete-time. Here, we are only interested in the dynamic action of the control automaton which is the discrete (event) part of the controller. The dynamic action of the discrete automaton is described by $\widetilde{\delta} : X_c \to X_c$. First, we introduce some necessary notation. Consider the product projection $p : \widetilde{S} \to X_c$ defined by $p(x_e, \widetilde{x}_p, x_c) = x_c$. We denote the restriction of the mapping p to the set $X_d \subset \widetilde{S}$ as $p|X_d$, the image of this mapping p as $\operatorname{Im} p|X_d$, and $\approx$ a bijective relation between two sets.

Definition 10 The controller is said to be an *internal model* of the exosystem if there exists $\widetilde{\delta} : X_c \to X_c$ such that

$$\widetilde{\delta} \circ p|X_d = (p|X_d) \circ \widetilde{f}|X_d \tag{112}$$

and

$$\operatorname{Im} p|X_d \approx X_d. \tag{113}$$

Equation (112) can be represented by the commutativity of the diagram in Figure 17 and describes the internal model in the controller. The condition described by Equation (113) insures that every state of the exosystem is contained in the controller.

$$\begin{tikzcd}
X_d \arrow[r, "\widetilde{f}"] \arrow[d, "p|X_d"'] & X_d \arrow[d, "p|X_d"] \\
X_c \arrow[r, dashed, "\widetilde{\delta}"'] & X_c
\end{tikzcd}$$

Figure 17: Internal model.

The dynamic evolution of the control automaton presented in Section 8 is defined as follows. The next state of the automaton is determined by measuring the state of the hybrid plant after the application of the regulator transition function. Therefore, the state transition function of the control automaton satisfies the condition

$$\widetilde{\delta} \circ p|X_d = (p|\widetilde{S}) \circ \widetilde{f}|X_d \tag{114}$$

which is described by the diagram in Figure 18

Furthermore, we have

$$(p|\widetilde{S}) \circ \widetilde{f}|X_d = (p|X_d) \circ \widetilde{f}|X_d \tag{115}$$

since when the regulation condition is satisfied we have that $\widetilde{f} : X_d \to X_d$. Therefore, the condition (112) is established.

In order to show that the condition (113) also holds, observe that the state set of the control automaton is defined as $X_c = X_e$ and furthermore, the controller can uniquely determine the state of the exosystem by the overall dynamics of the regulator. Therefore, the function $p|X_d$ is injective and condition (113) is also established.

We have demonstrated that if the hybrid system regulator has a solution then the controller contains a copy of the dynamic action of the exosystem according the internal model principle.

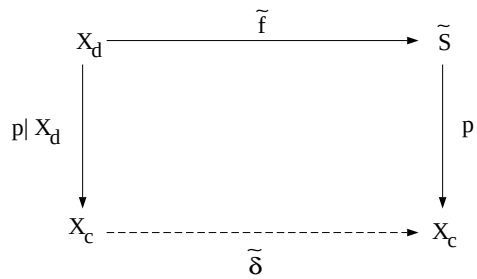

Figure 18: Dynamic action of the control automaton.

9 Simulation Results

In the following, we illustrate the controller design methodology using the temperature control system. The temperature control system is described in Section 4. Safety and reachability conditions for the temperature control system were investigated in Section 7. The controller for the temperature control system consists of the event generator, the control automaton, and the actuator. The event generator determines the membership of the state (q, x) in one of the regions R_l, R_s, R_h, or R_u; see the formal model of the specifications for the temperature control system in Section 7. The control automaton is shown in Figure 19. Every output of the of the control automaton corresponds to a cost functional, and the control input is selected by minimizing this cost functional at every time step over all the possible control actions. At every time step, a linear programming problem is solved for the mode q_0. For the mode q_1 the control input is always $u = 0$.

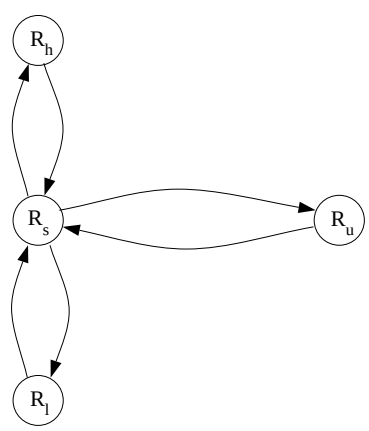

Figure 19: Control automaton for the temperature control system.

We consider the regions $R_1 = (\{q_0, q_1\}, P_1)$ and $R_2 = (\{q_0, q_1\}, P_2)$ where

$$P_1 = \{x \in \Re^2 | (0 \leq x_1 \leq 20) \wedge (-20 \leq x_2 \leq 0)\} \tag{116}$$

and

$$P_2 = \{x \in \Re^2 | (0 \leq x_1 \leq 20) \wedge (0 \leq x_2 \leq 5)\}. \tag{117}$$

The controller guarantees that every state in the region R_1 can be driven to the safe region R_2 and remain safe for every t. The control input is selected so that it minimizes the cost functional

$$J(q, x, u, d) = (A_q x + B_q u + E_q d) - \bar{x} \tag{118}$$

where $\bar{x} = [10, 2.5]^T$.

In the following, we present simulation results for different initial conditions to illustrate the validity of the controller. In Figures 20 - 23, we consider different initial conditions in the region R_1, and we show how the the state is driven to the region R_2. Finally, in Figures 24 - 29 we consider different initial conditions inside the safe region R_2, we consider the safety specification for R_2 and we plot $x(t), q(t)$, and $u(t)$. For the simulation, the disturbance d was defined as a sinusoidal signal of appropriate magnitude (see Table 1).

10 Conclusions

In this paper, we consider hybrid systems in which the continuous dynamics are described by linear difference equations, the discrete dynamics by finite automata, and the interaction between the continuous and discrete part is defined by piecewise linear maps. The proposed modeling formalism separates the physical plant to be controlled from the control specifications and the controller. It provides the necessary mathematical tools to describe explicitly what control actions are available in order to influence the behavior of the plant so that the control specifications are satisfied. We present a novel methodology for the control design of piecewise linear hybrid dynamical systems based on a formulation of the regulator problem. We present a formal control design framework for both static specifications that do not change as time progresses and dynamic specifications that involve sequencing of events and eventual execution of actions. More specifically, we develop control design algorithms for safety and reachability of piecewise linear hybrid systems as well as dynamic specifications describing membership of the state to regions of the state space. Our control design methodology leads to dynamical controllers that guarantee that the closed loop system consisting of the plant and the controller satisfies the formal specifications. Control design is implemented using finite automata and linear programming techniques. The main characteristic of the feedback controller is that it contains the control automaton that is used to select appropriate cost functionals that are minimized by selecting specific control actions. Furthermore, the control automaton is a representation of the formal specifications according to the internal model principle. The control design methodology is illustrated in detail using a temperature control system.

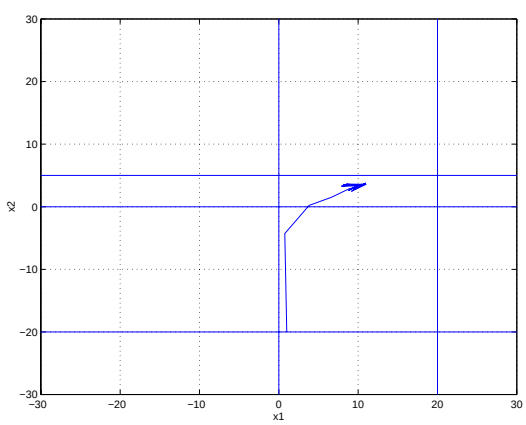

Figure 20: Temperature control system simulation 1: $x_0 = [1, -20]^T$.

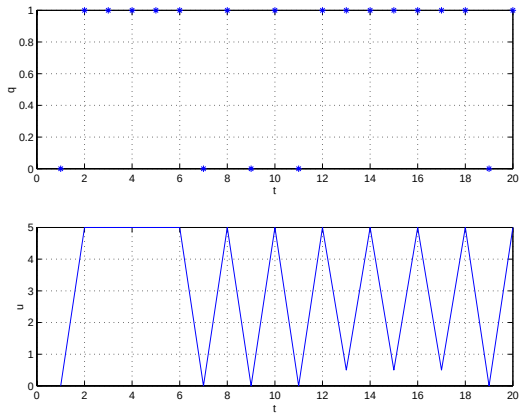

Figure 21: Temperature control system simulation 1: Control input.

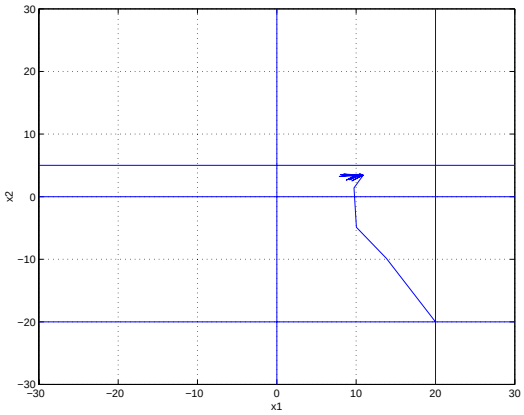

Figure 22: Temperature control system simulation 2: $x_0 = [20, -20]^T$.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P.-H. Ho, X. Nicollin, A. Oliveiro, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical and Computer Science*, 138:3–34, 1995.

[2] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of IEEE*, 88(7):971–984, July 2000.

[3] P. Antsaklis, editor. *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications*, July 2000.

[4] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In N. Lynch and B. Krogh, editors, *Hybrid Systems—Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 20–31. Springer-Verlag, 2000.

[5] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of IEEE*, 88(7):1011–1025, July 2000.

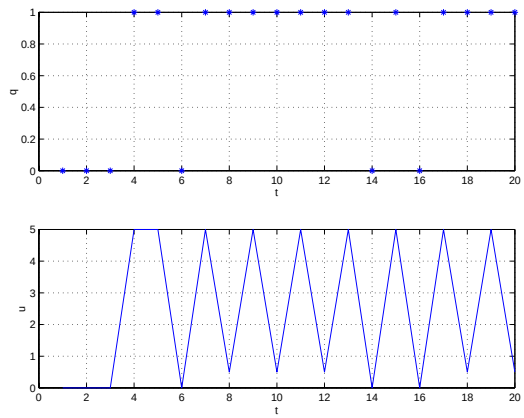

Figure 23: Temperature control system simulation 2: Control input.

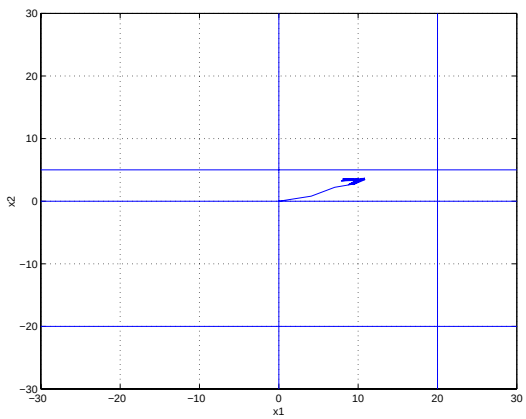

Figure 24: Temperature control system simulation 3: $x_0 = [0, 0]^T$.

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

[7] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In F. Vaandrager and J. van Schuppen, editors, *HSCC 99: Hybrid Systems—Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 31–45. Springer-Verlag, 1999.

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

[9] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *HSCC 99: Hybrid Systems—Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 46–60. Springer-Verlag, 1999.

[10] C. Chang. *Model Theory*. Elsevier, 1990.

[11] A. Chutinan and B. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 2089–2094, Tampa, FL, December 1998.

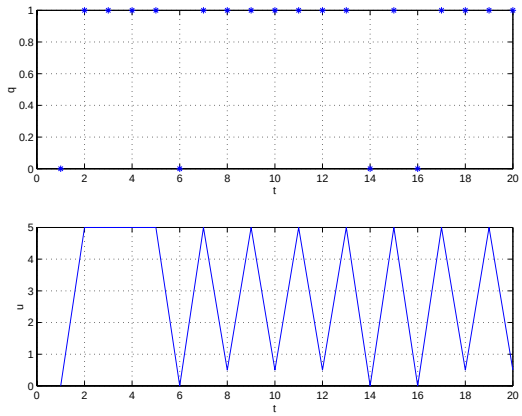

Figure 25: Temperature control system simulation 3: Control input.

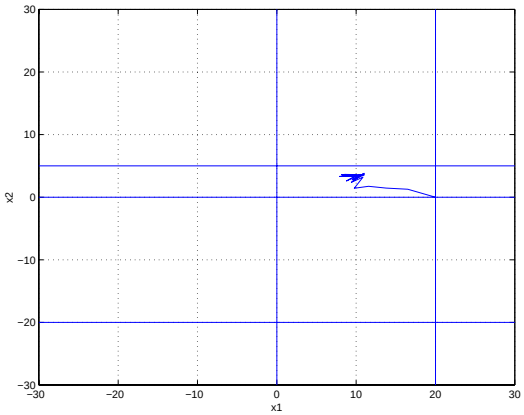

Figure 26: Temperature control system simulation 4: $x_0 = [20, 0]^T$.

[12] J. Cury, B. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control*, 43(4):564–568, 1998.

[13] R. Duffin. On Fourier's analysis of linear inequality systems. *Mathematical Programming Study I*, pages 71–95, 1974.

[14] T. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and G. Gécgeg, editors, *ICALP'95: Automata, Languages, and Programming*. Springer-Verlag, 1995.

[15] J. E. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[16] M. Johansson. *Piecewise Linear Control Systems.* PhD thesis, Lund University, Sweden, 1999.

[17] X. Koutsoukos. *Analysis and Design of Piecewise Linear Hybrid Dynamical Systems.* PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 2000.

[18] X. Koutsoukos and P. Antsaklis. Design of hybrid system regulators. In *Proceedings of the 38th IEEE Conference on Decision and Control*, pages 3990–3995, Phoenix, AZ, December 1999.

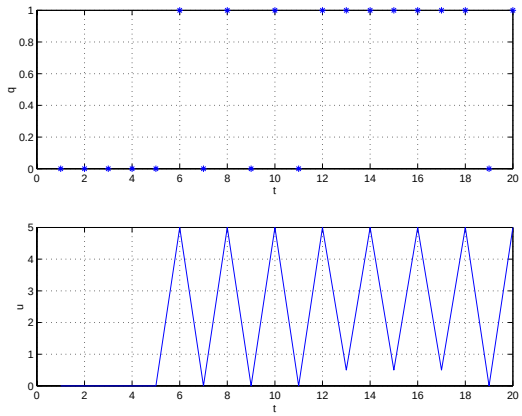

Figure 27: Temperature control system simulation 4: Control input.

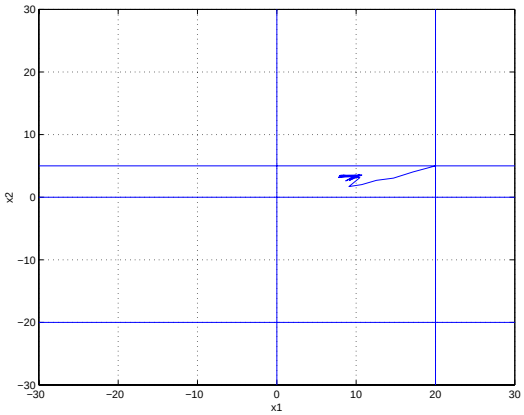

Figure 28: Temperature control system simulation 5: $x_0 = [20, 5]^T$.

[19] X. Koutsoukos and P. Antsaklis. Hybrid control of a robotic manufacturing system. In *Proceedings of the 7th IEEE Mediterranean Conference on Control and Automation*, pages 144–159, Haifa, Israel, June 1999.

[20] X. Koutsoukos and P. Antsaklis. A hybrid feedback regulator approach to control an automotive suspension system. In N. Lynch and B. Krogh, editors, *Hybrid Systems—Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 188–201. Springer-Verlag, 2000.

[21] X. Koutsoukos and P. Antsaklis. Hierarchical control of piecewise linear systems based on discrete abstractions. Technical Report ISIS-2001-001, ISIS Group at the University of Notre Dame, February 2001. Submitted for publication.

[22] X. Koutsoukos, P. Antsaklis, J. Stiver, and M. Lemmon. Supervisory control of hybrid systems. *Proceedings of IEEE*, 88(7):1026–1049, July 2000.

[23] G. Lafferriere, G. Pappas, and S. Sastry. Hybrid systems with finite bisimulations. 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 186–203. Springer, 1999.

[24] D. Leenaerts and W. van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer, 1998.

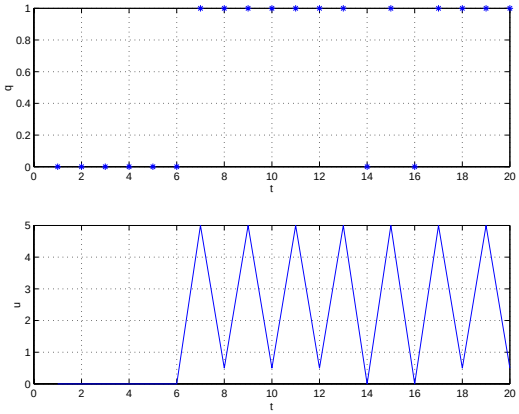

Figure 29: Temperature control system simulation 5: Control input.

[25] M. Lemmon. On the existence of solutions to controlled hybrid automata. In N. Lynch and B. Krogh, editors, *Hybrid Systems—Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 229–242. Springer-Verlag, 2000.

[26] J. Lunze, B. Nixdorf, and J. Schroder. Deterministic discrete-event representations of linear continuous-variable systems. *Automatica*, 35(3):396–406, 1999.

[27] N. Lynch, R. Segala, F. Vaandrager, and H.Weinberg. Hybrid I/O automata. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer, 1996.

[28] A. Morse. Supervisory control of families of linear set-point controllers-Part 1: Exact matching. *IEEE Transactions on Automatic Control*, 41:1413–1431, 1996.

[29] A. Morse, editor. *Control using logic-based switching*, volume 222 of *Lecture Notes in Control and Information Sciences*. Springer, 1997.

[30] T. Motzkin. *The theory of linear inequalities*. Rand Corp., Santa Monica, CA, 1952.

[31] S. Nash and A. Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[32] A. Nerode and W. Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 317–356. Springer-Verlag, 1993.

[33] A. Nerode and R. Shore. *Logic for Applications*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.

[34] J. Ostroff. *Temporal Logic for Real-Time Systems*. Research Studies Press, 1989.

[35] G. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, 45(6):1144–1160, 2000.

[36] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):568–573, 1998.

[37] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–89, January 1989.

[38] M. Sain. *Introduction to Algebraic System Theory*. Academic Press, 1981.

[39] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.

[40] E. Sontag. Remarks on piecewise-linear algebra. *Pacific Journal of Mathematics*, 92(1):183–210, 1982.

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

[42] J. Stiver, P. Antsaklis, and M. Lemmon. Digital control from a hybrid perspective. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 4241–4246, Lake Buena Vista, FL, December 1994.

[43] J. Stiver, P. Antsaklis, and M. Lemmon. A logical DES approach to the design of hybrid control systems. *Mathl. Comput. Modelling*, 23(11/12):55–76, 1996.

[44] J. Stiver, X. Koutsoukos, and P. Antsaklis. An invariant based approach to the design of hybrid control systems. Technical Report ISIS-2000-001, ISIS Group at the University of Notre Dame, February 2000. To appear in the International Journal of Robust and Nonlinear Control.

[45] C. Tomlin, J. Lygeros, and S. Sastry. Synthesizing controllers for nonlinear hybrid systems. In T. Henzinger and S. Sastry, editors, *HSCC 98: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1386, pages 360–373. Springer-Verlag, 1998.

[46] J. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 36(3):259–294, 1991.

[47] H. Williams. Fourier's method of linear programming and its dual. *American Mathematical Monthly*, 93:681–695, 1986.