

Hybrid Control of a Robotic Manufacturing System *

Xenofon D. Koutsoukos and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
e-mail: `xkoutsou,antsaklis.1@nd.edu`

Abstract

In this paper, a new approach for control of hybrid systems is introduced and illustrated using a robotic manufacturing system. Hybrid systems, which are used to model the physical process, and their controllers are viewed as system components of an intelligent control framework and they are modeled as set-dynamical systems. The central concept studied in hybrid system modeling is quasideterminism and it is used to address the problems that arise because of the nondeterministic nature of the discrete approximations of the continuous dynamics. Decision algorithms are derived based on supervisory control of discrete-event systems described by Petri nets. The notions of abstraction and multirate time scales play an important role in the design of decision algorithms to supervise the operation of the system.

1 Introduction

Hybrid control systems typically arise from the interaction of discrete planning algorithms and continuous processes, and as such, they frequently arise in the computer aided control of continuous processes in manufacturing, communication networks, and industrial process control, for example. The study of hybrid control systems is essential in designing sequential supervisory controllers for continuous systems, and it is central in designing intelligent control systems with a high degree of autonomy. An intelligent control architecture for high autonomy systems has been proposed in [3] (for intelligent control architectures see also [17], the contributions in [2] and the references therein). The architecture shown in Fig. 1 outlines an intelligent control framework which is used to describe how hybrid systems can be important components in the design of high autonomy systems. Hybrid control systems appear in the intelligent autonomous control system framework whenever one considers the execution level together with control functions performed in the higher coordination and management levels.

In this paper, a new approach for modeling and control of hybrid systems is presented. The main characteristic of the modeling approach is that hybrid systems are represented as set-dynamical systems. Set-dynamical systems can be viewed as dynamical systems with no algebraic structure and they are suitable in modeling both continuous and discrete components. Although the framework is very general, system theoretic notions as reachability, observability, and regulation

*The partial financial support of the National Science Foundation (ECS95-31485) and the Army Research Office (DAAG55-98-1-0199) is gratefully acknowledged.

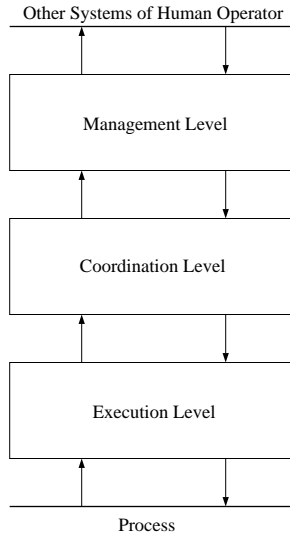


Figure 1: Hierarchical architecture for intelligent control

are still well-defined and can be translated directly into a hybrid framework. Another important characteristic driven by the advent of digital devices in modern control engineering is the use of a discrete-time framework. Complex engineering applications usually consist of system components (physical processes, digital microcontrollers, computer networks) that operate at different time scales. Continuous time models have been used to address fundamental issues such as the existence and uniqueness of solutions in the presence of discontinuities. The use of discrete-time models abstracts the theoretic difficulties and focuses on more practical issues. The need behind the modeling approach is the development of a mathematical framework that will enable the use of control ideas from either continuous or discrete-event systems. The class of systems we are interested in is the class of piecewise-linear systems [20]. Piecewise-linear systems represent an interesting class for engineering applications and they can be studied with existing mathematical tools [18, 19].

The design methodology presented in the paper is motivated mainly by the work in [21] (see also [4, 22]). More specifically, we present a method to abstract the continuous dynamics of hybrid systems by discrete-event models and we concentrate our analysis on the nondeterministic nature of the derived models. Central in our treatment is the concept of quasideterminism which is used to address the problems that arise due to the nondeterminism in the approximating discrete-event systems. Abstractions of dynamical systems have been used extensively in the hybrid system literature, see for example [14, 6, 15, 12]. Controller synthesis methods are usually based on automata models [23, 7]. Here, supervisor control methods based on Petri nets [10] are used mainly because of their computational advantage to enforce control objectives that can be described as convex constraints on the marking of the Petri net.

The remaining of the paper is organized as follows. In Section 2, a robotic manufacturing system is described that will be used to illustrate our approach for control of hybrid systems. In Section 3, the mathematical model for hybrid systems is presented and illustrated using the robotic manufacturing example. In Section 4, the notion of quasideterminism is presented and a method to obtain partitions of the state space that lead to quasideterministic systems is described. The controller synthesis approach is based on supervisory control of discrete-event systems using Petri

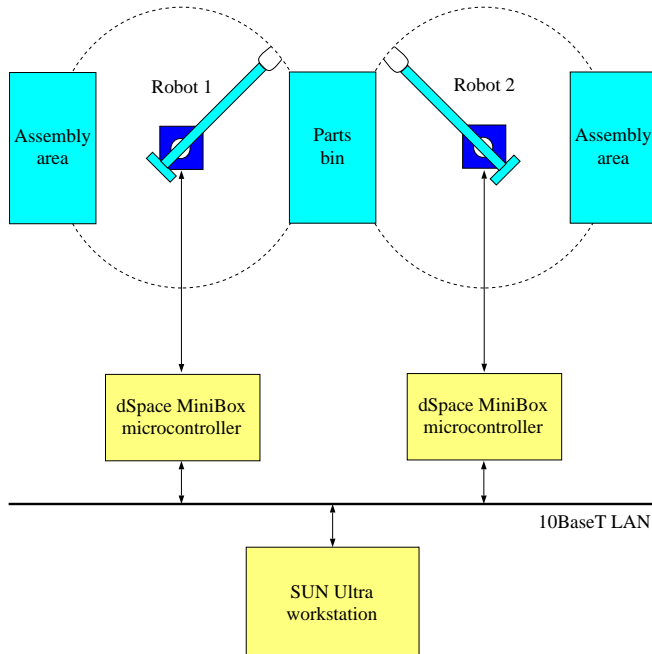


Figure 2: Robotic manufacturing system

nets and is presented in Section 5.

2 Description of the robotic manufacturing system

A robotic manufacturing system (RMS) will be used to illustrate our approach for control of hybrid systems. The system shown in Fig. 2 consists of two robots whose task is to move components from a *parts bin* to an *assembly area*. The operation of the RMS requires that the task of fetching a workpiece and transporting it to the assembly area has to be performed periodically following the production needs of a manufacturing plant. It is assumed that the parts bin is shared by the robotic arms which cannot enter the fetching area simultaneously. Each robotic arm is driven by an armature-voltage-controlled DC servomotor (Quanser Servo SRV-02). Each servomotor is controlled by a local controller (dSpace MiniBox microcontroller) and the overall system is monitored and coordinated by a supervisor (implemented in a SUN Ultra workstation). The supervisor communicates with the local controllers via a standard computer network (10BaseT LAN). The system described above is an experimental setup in the control lab at the University of Notre Dame (<http://www.nd.edu/~lemmon/aro-durip>).

Problem statement Our objective is the design of decision and control algorithms that guarantee the safe and efficient operation of the RMS. Conventional control methods can be used in the execution level to address problems as tracking and disturbance rejection. The control objective at the coordination level is to supervise the actions of the robotic arms to ensure that they will not enter the parts bin at the same time. The parts bin represents the critical section of the system and it is described by $|\theta_i| \leq 45$ where θ_i is the angular position of the i th robotic arm. This problem

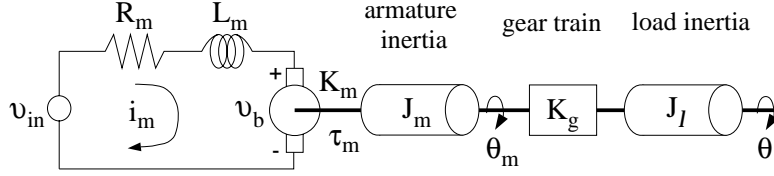


Figure 3: Armature-voltage-controlled DC servomotor

is simple enough to be described in the paper, but also rich enough to demonstrate our approach. The safety requirement at the coordination level can be addressed using numerous approaches based on logical models that can result in attractive solutions. However, there is an important need to investigate additional approaches that take into consideration the continuous dynamics. Such methodologies can be useful in applications with more complicated, stringent, and dynamic requirements. It should be expected that the consideration of the continuous dynamics may lead to inefficient solutions for the specific problem. The fact that the problem has artificially enhanced in order to demonstrate our approach must be considered for the evaluation of the method. In the following, we present the dynamics of the system as modeled at the different levels of abstraction of the hierarchical control architecture.

Physical Process The system of the servomotor and the lever arm is shown in Fig. 3. The dynamics are described by the simplified set of differential equations

$$v_{in} = R_m i_m + K_m K_g \frac{d\theta}{dt} \quad (1)$$

$$i_m = \frac{\tau_m}{K_m} = \left(\frac{K_g^2 J_m + J_l}{K_k K_g} \right) \frac{d^2\theta}{dt^2} \quad (2)$$

The transfer function derived using the motor parameters supplied by the manufacturer is

$$\frac{\theta(s)}{v_{in}(s)} = \frac{1}{s(0.0026s + 0.1081)} \quad (3)$$

The output of the system is the angular position of the robotic arm with respect to a fixed reference system. The parts bin corresponds to $\theta = 0$ and the assembly area to $\theta = 180$.

Execution Level The open loop position response of the servomotor is unstable due to the pole at the origin. The objective at this level is to design local controllers for each servomotor so that each task is performed in an acceptable manner. Another control objective for the specific system is to protect the servomotor for high frequency voltages that can eventually damage the gearbox or the brushes. Conventional control methods are applied to force each robotic arm to follow a reference trajectory with satisfactory performance. The control algorithms are implemented in the microcontrollers. The closed loop system consisting of the servomotor and its local controller is described by the sampled data system with sampling period $T = 0.001s$

$$x(k+1) = \bar{A}x(k) + \bar{B}r(k) \quad (4)$$

$$y(k) = \bar{C}x(k). \quad (5)$$

There are three available reference inputs with corresponding controllers associated with the commands of the supervisor. A command `goto_parts_bin` issued by the supervisor is translated to the reference signal $r_1 = 0$ representing the angular position when the arm is at the parts bin. The controller used for this task allows a fast response of the system and results in the following closed-loop system representation

$$\bar{A}_1 = \begin{bmatrix} 0.96853 & 0.00076411 \\ -62.5 & 0.53147 \end{bmatrix}, \bar{B}_1 = \begin{bmatrix} 8.1813 \cdot 10^{-5} \\ 0.1625 \end{bmatrix}, \bar{C}_1 = [384.62 \quad 0]. \quad (6)$$

Similarly, for the second task `goto_assembly_area`, the corresponding reference signal is $r_2 = 180$ and a more conservative controller is used to guarantee an overdamped response in order to protect sensitive workpieces. The closed-loop system representation is

$$\bar{A}_2 = \begin{bmatrix} 0.99874 & 0.00095029 \\ -2.5 & 0.90126 \end{bmatrix}, \bar{B}_2 = \begin{bmatrix} 3.2725 \cdot 10^{-6} \\ 0.0065 \end{bmatrix}, \bar{C}_2 = [384.62 \quad 0]. \quad (7)$$

The last command available to the supervisor is `stop`. It is assumed that no brake command is available and that the arm can stop only because of its natural damping. In this case the open loop system is associated with the reference input $r_3 = 0$ and we have

$$\bar{A}_3 = \begin{bmatrix} 1 & 0.0009795 \\ 0 & 0.95928 \end{bmatrix}, \bar{B}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \bar{C}_3 = [384.62 \quad 0]. \quad (8)$$

Coordination Level The control objective at this level is to coordinate the actions of the robotic arms to ensure that they will not enter their critical section (parts bin) simultaneously. The mathematical model used at the coordination level must combine the dynamics of the constituent subsystems and take into account the effect of the communication network. Very simple algorithms can be used to satisfy the safety requirement. However, an optimality criterion regarding the permissiveness of the supervisor can imposed in a natural way. Practically, it is desirable to characterize all the legal behaviors and design a supervisor that is maximally permissive (or minimally restrictive) [24]. Such a design will give the flexibility at the higher level to select safe supervisor strategies that take into consideration additional criteria concerning the energy consumption and the production needs for example. Our central idea is to model the RMS at the coordination level as a hybrid system. This paper presents some recent results for modeling, analysis, and synthesis of hybrid systems in the intelligent control framework described above. All the theoretical notions are presented and explained in terms of the robotic manufacturing system.

Management Level As mentioned earlier, the objectives at this level are concerned with the selection of supervisory strategies to optimize additional criteria. We believe that hybrid systems can be also used successfully at this level, however the study of relevant problems is out of the scope of the present paper.

3 Hybrid systems modeled by set-dynamical systems

3.1 Set-dynamical systems

Our viewpoint is that all processes in an intelligent control system, either discrete or continuous, can be presented as set-dynamical systems. The advantage of such a unified representation is that

it provides the tools for interconnecting heterogeneous systems via input-output maps, abstracting parts of the processes using equivalence relations, and reconciling the time scales of the processes using "timing maps". The definitions presented here are from [16]. The modeling formalism of set-dynamical systems is general enough since no structure is imposed on the sets and functions that define these systems. However, very important ideas like reachability, observability, and regulation can be still discussed in this general setting. For more details see [16].

A *set-dynamical system* (SDS) is denoted as

$$(X, U, Y; f, g) \tag{9}$$

where X is the state set of the system, U is the input set, Y is the output set, $f : X \times U \rightarrow X$ is the state transition function, and $g : X \times U \rightarrow Y$ is the output function. Note that no structure is imposed on any of the sets or functions included in the definition of a set-dynamical system. Therefore, a set-dynamical system can be seen as a dynamical system with no algebraic structure. It is often convenient to distinguish between the controlled and the uncontrolled inputs of an SDS. For this purpose, the input set can be written as $U = A \times D$ where A is the set of control actions and D is the set of disturbances. In the case when the measurements are different than the outputs, a measurement set M and a measurement function m can be also included in the system's description. The SDS is then denoted by

$$(X, A, D, Y, M; f, g, m) \tag{10}$$

where $f : X \times A \times D \rightarrow X$, $g : X \times A \times D \rightarrow Y$, and $m : X \times A \times D \rightarrow M$.

To define the timing characteristics of the system (10) we introduce a set of "times" called the *index set* equipped with an order relation. Following [16] given the *index set* J , we define *index functions* $\alpha : \mathbb{N} \rightarrow J$. An index function is said to be *admissible* if

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

The state $x \in X$ is associated with an index $j(n) \in Im\alpha$ meaning the state at time $j(n)$. Similarly, the input, output, and measurement signals of the systems can be associated with the index $j(n)$. Then the action of the system is described by

$$\begin{aligned} x(j(n+1)) &= f(x(j(n)), u(j(n))) \\ z(j(n)) &= g(x(j(n)), u(j(n))) \\ y(j(n)) &= m(x(j(n)), u(j(n))) \end{aligned} \tag{11}$$

Let J be an index set and $\alpha \in J^{\mathbb{N}}$ an index function. Then the pair (α, J) describes the timing characteristics of the system. The dynamic behavior of the system is understood with respect to the time advancement defined by the pair (α, J) . In the case the pair (α, J) has been selected, the cumbersome notation of equations (11) takes the usual form

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ z(k) &= g(x(k), u(k)) \\ y(k) &= m(x(k), u(k)) \end{aligned} \tag{12}$$

By the previous discussion, it is clear that we consider set-dynamical systems that evolve in discrete-time. By the admissibility requirements, it follows that (discrete) time cannot be prevented from diverging and that two state transitions cannot occur simultaneously. The utilization of index sets and index functions provides the tools to describe synchronous and asynchronous operation and to connect synchronous with asynchronous systems. Time interpretations of discrete-event systems have been treated in a similar manner in [13]. Index sets and functions will be used also to define the interplay between the time scales at different levels of abstraction.

The modeling formalism of set-dynamical systems is adequate to describe either continuous or discrete systems since no algebraic structure is imposed on the involved sets. If all the sets that appear in the definition of a SDS are continuous (discrete) then the SDS is characterized as continuous (discrete). An SDS is characterized as a *hybrid dynamical system* if either it contains both discrete and continuous sets or if some sets are defined as the Cartesian product of a continuous and a discrete set. Set-dynamical systems establish a unified framework for the system representations that arise in a hierarchical architecture for intelligent control. However, for the development of efficient analysis and synthesis methods the special characteristics of the representations and the algebraic structure of the individual systems should be exploited.

3.2 Hybrid systems

The class of systems that we are interested is the class of *piecewise-linear systems* [18, 20]. 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 includes linear systems, finite state machines, and their interconnections [20].

An interesting case that arises very often in intelligent control is when the plant is described as a hybrid system composed of multiple linear time invariant subsystems which can be switched between using a logical decision rule. In this case, it is assumed that the controlled inputs or the available control actions are the *control modes* of the hybrid system that determine which subsystem is active at each time instant. A hybrid system satisfying the previous assumptions can be modeled by the SDS

$$(X, A, D, Y, M; F, \pi, \pi_F) \quad (13)$$

where $X = \mathbb{R}^n$ is the state set, A is a finite set containing the control modes and is viewed as the input set of the hybrid system, D is the disturbance set which is assumed to be a polytope in \mathbb{R}^m , and Y and M are the output and measurement set respectively. The state transition function $F : X \times A \times D \rightarrow X$ is defined as follows. For fixed control action $a \in A$, the dynamics are described by a discrete-time linear time invariant system

$$x(k+1) = Ax(k) + Bd(k) \quad (14)$$

The control mode which is considered here to be the input of the hybrid system can change at every time instant k meaning that the system can switch from a subsystem to another at $t = kT$ for every $k \in J$, where T is the sampling period of the system.

The mappings π and π_F represent the output and the measurement function respectively. The desired behavior of the hybrid system will be expressed with respect to the output signals and therefore, the output function can be selected from the control specifications. The measurement

function on the other hand must be selected so that the measurement signals provide sufficient information to facilitate control and decision algorithms. The mappings π and π_F are defined as projections from the state set X to the power set of X . The output and measurements sets Y and M can be defined as the images of the mappings π and π_F respectively. The output and measurement functions define partitions of the state set. Because of their importance in our study of hybrid systems, they will be discussed at length in the remaining of the paper.

3.3 Control specifications and the primary partition

The mapping π can be viewed as an “analog-to-digital” map defined as a projection of the state set $X = \mathbb{R}^n$ to a finite set of regions that cover the set X . Because we like to remain in the class of piecewise-linear systems, it is required that these regions are defined by linear equalities and inequalities and therefore, can be described by piecewise-linear sets in the state space. It is assumed that the partition defined by the generator is appropriate for extraction of important information for the system and it will be called the *primary partition*. Often the control specifications in hybrid systems are expressed as the membership of the continuous state in safe regions in the state space. The primary partition describes exactly these regions. The assumption that these regions are described by linear equalities and inequalities is of great practical importance. Such regions arise in many applications by considering the crucial thresholds for the behavior of the system in the state space.

First, some basic notions from algebraic system theory [16] that are necessary for the definition of the output and measurement mappings will be described. Let $E(X)$ be the set of all equivalence relations on X . We can define the poset $(E(X), \leq)$ where the partial order relation \leq on $E(X)$ defined as

$$E_1 \leq E_2 \text{ if } x_1 E_1 x_2 \Rightarrow x_1 E_2 x_2.$$

$E(X)$ is bounded below by the equality relation and above by the equivalence relation corresponding to X^2 . A lattice structure can also be developed on the set of all equivalence relations on X .

Next, consider the state set X of a SDS and define the mapping $\pi : X \rightarrow \mathbb{P}(X)$ 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

$$x_1 E_\pi x_2 \text{ iff } \pi(x_1) = \pi(x_2) \tag{15}$$

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

$$\pi : X \rightarrow X/E_\pi \tag{16}$$

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*.

The generator map is denoted by $\pi : X \rightarrow X/E_\pi$ where $X = \mathbb{R}^n$ is the state space of the hybrid system and X/E_π is the quotient space of X by the equivalence relation E_π . The generator is defined by a set of hyperplanes in \mathbb{R}^n . First, consider the collection

$$\{h_i\}_{i=1,2,\dots,d}, \quad h_i : \mathbb{R}^n \rightarrow \mathbb{R} \tag{17}$$

of real-valued functions of the form (affine)

$$h_i(x) = a_i^T x - b_i, \quad i = 1, 2, \dots, d \tag{18}$$

where $a_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$. Let H_i denote the kernel of $h_i(x)$,

$$H_i = \ker(h_i) = \{x \in \mathbb{R}^n : h_i(x) = a_i^T x - b_i = 0\} \quad (19)$$

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

$$\hat{h}_i(x) = \begin{cases} -1 & \text{if } h_i(x) < 0 \\ 0 & \text{if } h_i(x) = 0 \\ 1 & \text{if } h_i(x) > 0 \end{cases} \quad (20)$$

Then, the generator is defined by

$$\pi(x) = [\hat{h}_1(x), \dots, \hat{h}_d(x)]^T \quad (21)$$

Therefore, the output $y = \pi(x)$ can be viewed as a column vector consisting of 0s, 1s, and -1s describing when the continuous state belongs to an intersection of open halfspaces in the state space or when the state lies in the intersection of the hyperplanes H_i . Two continuous states x_1 and x_2 are equivalent if and only if $\pi(x_1) = \pi(x_2)$. Observe that although the generator has been defined as $\pi : \mathbb{R}^n \rightarrow \{-1, 0, 1\}^d$ there is a bijection between $\{-1, 0, 1\}^d$ and the quotient set X/E_π (they are eventually the same set). Since any other symbols could be used in (20) to identify the region the continuous state lies in, the image of the generator will be denoted as X/E_π . Therefore, the output of the hybrid system $y(k) = \pi(x(k)) \in X/E_\pi$ represents the equivalence class of the state $x(k)$. The measurement function π_F defines a new partition that will be called the *final partition* of the system and can be also represented as an equivalence relation on X . The design of a feedback controller depends directly on the available measurements. In Section 4, a desirable property of the final partition called quasideterminism is studied. Quasideterminism leads to a constructive way to design the final partition of the hybrid system.

3.4 Robotic manufacturing system

In this section, the RMS is modeled by a hybrid system. In order to solve the coordination problem, the mathematical model of the system must include the dynamics of the two servomotors and take into consideration the effects of the computer network. The angular position and velocity of each robotic arm are measured and are used for feedback by the local controller. The same information is transmitted to the workstation in order to be used by the supervisor. The local controllers and the workstation synchronize their internal clocks by exchanging messages. An upper bound of the time delay of a round trip message from the workstation to the local controller and back is $T_d = 0.01s$. Therefore, it is natural to consider that the model of the physical process seen by the supervisor is evolving at a time rate $1/T_d$. Two different time scales are used for the control of the system. The time scale density at the coordination level is lower than that at the execution level in accordance with the characteristics of the hierarchical control architecture. The timing characteristics of the hybrid system are described by the pair (α, J) where $\alpha(n) = nT_d$. Note that, although in this example α describes just the decimation (downsampling) of the discrete-time model at the execution level, the notion of index functions is more general and can be used to describe nonuniform or multirate sampling, even synchronous or asynchronous operation.

Practically, when the message carrying the angular position of the arm reaches the workstation, the actual position will have changed. In order to take into account the stochastic nature of the time delays in the network, a disturbance term must be included in the model. By the assumption for the upper bound of the time delay of a message, the disturbance can be described as a bounded additive term in the state space representation of each subsystem.

Following the previous discussion, the RMS is described by the hybrid system

$$(X, A, D, Y, M; F, \pi, \pi_F) \quad (22)$$

$X = \mathbb{R}^4$ is the state set with $x = [x_1, x_2, x_3, x_4]^T$ where x_1, x_2 and x_3, x_4 are the states of robot 1 and robot 2 respectively.

$A = \{\text{goto_parts_bin}, \text{goto_assembly_area}, \text{stop}\}_{robot1} \times \{\text{goto_parts_bin}, \text{goto_assembly_area}, \text{stop}\}_{robot2}$ is the set of the control actions available to the supervisor. For simplicity, we will represent this set by $A = \{a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}, a_{31}, a_{32}, a_{33}\}$ where a_{ij} corresponds to the i th task for robot 1 and the j th task for robot 2 (in the above order).

$D \subset \mathbb{R}^4$ is a bounded polytope described by $|d_i| \leq 0.1$, $i = 1, 2, 3, 4$.

$F : X \times A \times D \rightarrow X$ is the state transition function. For fixed control action a_{ij} we have that

$$x(k+1) = \begin{bmatrix} A_i & 0 \\ 0 & A_j \end{bmatrix} x(k) + \begin{bmatrix} B_i & 0 \\ 0 & B_j \end{bmatrix} \begin{bmatrix} r_i \\ r_j \end{bmatrix} + d(k) \quad (23)$$

The system parameters are determined by decimation (downsampling) of the state space representations (6), (7), and (8) and are given by

$$A_1 = \begin{bmatrix} 0.2204 & 0.00057373 \\ -46.9285 & -0.10777 \end{bmatrix}, B_1 = \begin{bmatrix} 0.002027 \\ 0.12201 \end{bmatrix}, r_1 = 0 \quad (24)$$

$$A_2 = \begin{bmatrix} 0.90593 & 0.0059892 \\ -15.756 & 0.29154 \end{bmatrix}, B_2 = \begin{bmatrix} 0.00024458 \\ 0.040966 \end{bmatrix}, r_2 = 180 \quad (25)$$

$$A_3 = \begin{bmatrix} 1 & 0.0081816 \\ 0 & 0.65983 \end{bmatrix}, B_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, r_3 = 0 \quad (26)$$

$Y = X/E_\pi$ is the quotient space induced by the generator π . As it was described earlier, the mapping π is determined by the control specification, which for the RMS can be expressed using the inequalities

$$-45 \leq \theta_1 \leq 45 \text{ and } -45 \leq \theta_2 \leq 45 \quad (27)$$

Translating the specification from the output space to the state space we get the unsafe region described by the inequalities

$$-0.1170 \leq x_1 \leq 0.1170 \text{ and } -0.1170 \leq x_3 \leq 0.1170 \quad (28)$$

Following the discussion of the previous section, we define the affine functions

$$h_1(x) = x_1 + 0.1170, h_2(x) = x_1 - 0.1170, h_3(x) = x_3 + 0.1170, \text{ and } h_4(x) = x_3 - 0.1170 \quad (29)$$

The generator $\pi : X \rightarrow X/E_\pi$ is then defined in a straightforward manner using the equations (18 - 21).

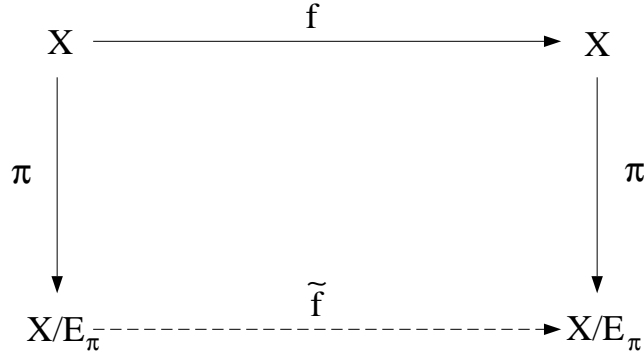


Figure 4: Quotient spaces and induced systems

4 The generator of the final partition

Intuitively, the generator π is used to coarsen the state space of the system. The question that arises is if the system can follow this abstraction. Consider the diagram shown in Fig. 4. The important question that arises is concerned with the existence of a mapping $\tilde{f} : X/E_\pi \rightarrow X/E_\pi$ that makes the diagram commute. It is shown in [16] that \tilde{f} exists if and only if

$$x_1 E_\pi x_2 \Rightarrow (\pi \circ f)(x_1) = (\pi \circ f)(x_2) \quad (30)$$

and moreover, if (30) 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 . Generators that satisfy the above property can be determined based on the natural invariants of the continuous dynamics and result in deterministic discrete-event systems [22]. In this paper, we concentrate on generators described by linear equalities and inequalities in order to exploit the characteristics of piecewise-linear systems. Consider two states $x_1, x_2 \in X$, $x_1 \neq x_2$ such that $\pi(x_1) = \pi(x_2) = z \in X/E_\pi$. The states x_1 and x_2 may be driven under the mapping f to different equivalence classes of the quotient space X/E_π . 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 $\tilde{f} : X/E_\pi \rightarrow X/E_\pi$ can be viewed as a nondeterministic system.

Suppose that at time k (considering the timing characteristics defined by the pair (α, J)), $\pi(x(k)) = z(k) \in X/E_\pi$. If it is agreed that the granularity of the generator π is appropriate for the extraction of important information for the plant, then it is desirable to uniquely determine the next state up to its membership on an equivalence class $z(k+1) \in X/E_\pi$. This can be accomplished by considering a finer partition than the partition defined by the generator π to obtain better estimates for the continuous state. 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 poset $(E(X), \leq)$ satisfy the condition $E_{\pi'} \leq E_\pi$. In the case when the estimates of the state at time k provide sufficient information to uniquely determine the membership of the state of the induced system at time $k+1$ on an equivalence class of E_π , the system is said to be *quasideterministic*.

To determine a finer partition that results in a quasideterministic system, we define the operator $pre_f : \mathbb{P}(X) \rightarrow \mathbb{P}(X)$ by

$$pre_f(P) = \{x \in \mathbb{R}^n : \neg \exists u \in U, f(x, u) = Ax + Bu \in P\} \quad (31)$$

where U is a polytope in the input space. The set $pre_f(P)$ represents all the states x for which there does not exist any $u \in U$ such that $f(x, u) \in P$. If the set P is polyhedral, then the set $pre_f(P)$ is also a polyhedral set in the state space. The proof of the above statements is based on the fact that equation (31) contains only piecewise-linear sets and piecewise-linear mappings, and it is omitted due to length limitations. For more details see [19].

Suppose that the polyhedral set P is bounded by the hyperplanes H_i , $i = 1, \dots, d$, then the hyperplanes that bound $pre_f(P)$ can be computed by the following algorithm.

Algorithm for the construction of the final partition

INPUT: $f(x, u) = Ax + Bu$, $H_i = \ker(h_i) = \{x \in \mathbb{R}^n : h_i(x) = a_i^T x - b_i = 0\}$, $i = 1, \dots, d$
for $i = 1 \dots, d$

$$\underline{u} = \underset{u \in U}{\operatorname{argmin}} (-a_i^T B u);$$

$$a_i'^T = a_i^T A;$$

$$b_i' = b_i - a_i^T B \underline{u};$$

end

OUTPUT: $H_i' = \ker(h_i') = \{x \in \mathbb{R}^n : h_i'(x) = a_i'^T x - b_i' = 0\}$, $i = 1, \dots, d$

In order to explain the previous algorithm, assume first that the input u is fixed. Then it is easy to show that

$$H_i' = pre_f(H_i) = \ker(h_i') = \{x \in \mathbb{R}^n : h_i'(x) = a_i'^T x - b_i' = 0\} \quad (32)$$

where $a_i'^T = a_i^T A$ and $b_i' = b_i - a_i^T B u$. From equation (32) it follows that the hyperplanes $H_i' = pre_f(H_i)$ for arbitrary inputs are parallel. Selecting the input as $\underline{u} = \underset{u \in U}{\operatorname{argmin}} (-a_i^T B u)$ corresponds to the worst case in view of the effect of the input $u \in U$. The above procedure is repeated for every hyperplane that bounds the set P to compute the hyperplanes H_i' , $i = 1, \dots, d$. Since P is a polyhedral, it is defined as the intersection of a finite number of halfspaces. Because of the linearity of the dynamics, $pre_f(P)$ will be defined as the intersection of halfspaces bounded by the hyperplanes H_i' , $i = 1, \dots, d$.

Consider now the hybrid system $(X, A, D, Y, M; F, \pi, \pi_F)$. The above algorithm is repeated for every control action and every hyperplane of the generator π of the primary partition. The generator of the final partition π_F is defined (using equations (18) - (21)) by the original hyperplanes and all the computed ones by the above algorithm. The only requirement for the algorithm that determines the final partition is that $a_i \notin \ker(A^T)$. This condition guarantees that all the hyperplanes have dimension $n - 1$. Repetitive applications of the above procedure may be necessary. For example if the granularity of the final partition π_F is not sufficient to solve the safety problem, the method can be applied to the hyperplanes that define π_F to get an even finer partition. In the next section, a Petri net based method to design supervisor for the safety problem given the final partition is presented. The conditions for the supervisor synthesis depend on controllability properties of the discrete-event model inherited from the final partition. These properties can be translated to conditions that the final partition (measurement function) must satisfy in order to solve the safety problem.

5 Controller synthesis approach based on Petri nets

In this section, the hybrid system $(X, A, D, Y, M; F, \pi, \pi_F)$ is approximated by a Petri net $N = (P, T, F)$. The places $p_i \in P$ correspond to the equivalence classes of the equivalence relation E_{π_F} induced by the generator of the final partition. For simplicity, it is assumed that initially each place corresponds to an open polyhedral set (defined as the finite intersection of open halfspaces). The boundaries of these polyhedral sets, which are also equivalence classes of E_F are associated arbitrarily with the open halfspaces as follows. Each boundary is associated with exactly one place so that the places correspond to disjoint regions that cover the state space. The above assumption is made to simplify our modeling approach. Note that problems associated with the boundaries of regions that partition the state space and the continuity of the analog-to-digital maps defined from these partitions have been studied at length in [11, 5]. The consideration of these results is essential for the study of hybrid systems and can be incorporated in the models described in this work. However, the use of these ideas here would obscure our modeling approach and will be omitted.

In the following, the polyhedral region associated with the place p_i will be denoted by P_i while the polyhedral regions defined by the primary partition will be denoted by Q_j . Observe that from the construction of the final partition it follows that for every P_i there exists exactly one region Q_j such that $P_i \subset Q_j$. In addition, each region of the primary partition can be written as the union of regions of the final partition. Therefore, each region Q_j of the primary partition corresponds to a set of places $S_j \subset P$. Next, consider two places $p_i, p_{i'} \in P$. A transition t with input place p_i and output place $p_{i'}$ exists if and only if there exists control action $a \in A$ such that $P_{i'} \subset Q_j$ and $pre_{F(a)}(Q_j) \cap P_i \neq \emptyset$. The interpretation of this statement is that there exists a state $x \in P_i$ that may be driven to the region Q_j of the primary partition under the state transition of the hybrid system. The transition t is labeled with the control action $a \in A$. Recall that a transition is said to be *controllable* if it can be disabled by an external control action. The marking of the Petri net represents membership of the continuous state in an equivalence class of the final partition, i.e. if p_i is marked then $x \in P_i$. Note that a class of Petri nets named programmable timed Petri nets has been used in [8] to model hybrid systems. The main idea is to use labeling functions over a propositional logic formulae to represent the polyhedral regions associated to each place. Here, we will use ordinary Petri nets keeping in mind the correspondence between the places and the polyhedral regions of the final partition. Note that the derived Petri net is equivalent to a finite state machine. Petri net representations are used because of the computational advantages they offer for the design of supervisors that enforce convex constraints on the marking.

Let Q_j representing an unsafe polyhedral region in the state space X . The objective of the supervisor is to prevent the state from entering the region Q_j . Let S the set of places corresponding to Q_j . Then the previous condition can be translated to the linear constraint $\mu(S) < 1$ in the marking of the approximating Petri net model. By the construction of the Petri net model, if we can design a supervisor that enforces the previous constraints, then the safe operation of the original hybrid system is also guaranteed. Linear constraints on the marking vector can be enforced by *monitor* or *controller* places. These places represent control places that are connected to existing transitions of the Petri net model. A methodology for DES control based on Petri net place invariants has been developed in [10, 9]. The method is used to enforce linear constraints on the marking vector even in the presence of uncontrollable and unobservable transitions. The Petri net supervisor disables all the transitions that can lead to markings that violate the constraint. For

more details for supervisory control of Petri nets see [10, 9].

5.1 Petri net supervisor for the robotic manufacturing system

The design of the Petri net supervisor is illustrated using the robotic manufacturing system. First, the final partition is designed. The primary partition consists of 4 hyperplanes. Since there are 9 available control actions, we finally obtain $40 = 4 \times 9 + 4$ hyperplanes in the state space \mathbb{R}^4 . Each region of the partition is represented by a place in the Petri net representation of the system. The number of places and transition is of order 10^2 . The critical section of the system is represented by a set S of places. A Petri net supervisor consisting of one control place and several arcs connecting this place to existing transitions can enforce the constraint $\mu(S) < 1$ and ensure the safe operation of the system. Some important remarks for the evaluation of the method are in order. First, it should be clear that the Petri net model of the system is equivalent to a finite state machine since exactly one place is marked at each time instant and therefore each place can be associated with a state. The important observation here is that by the construction of the Petri net, the forbidden state problem of this example can be expressed as a convex constraint on the marking. For this type of problems, the methods presented in [10] offer computational advantages over automata based methods.

It is well-known that for certain systems the expressiveness of Petri nets can lead to more compact representations. In the following, we will exploit specific characteristics of the problem to reduce the size of the Petri net. Observe that the dynamics of the robotic arms are decoupled and moreover, they are described by the same state space representation resulting in identical partitions. Therefore, we can consider the final partition to the state space of the one subsystem. In Fig. 5, we show the hyperplanes of the final partition that are generated for the hyperplane H_2 of the primary partition described by $h_2(x) = x_2 - 0.1170$. The number of regions can be reduced drastically by taking into count the saturation constraints of the system. For example in Fig. 5, it is assumed that $|x_2| \leq 30$ which corresponds to a saturation constraint of the angular velocity of each servomotor. Note that saturation constraints are often described by linear inequalities and therefore, they can be included in our modeling framework without any changes. Given the state space partition shown in Fig. 5 for the first robotic arm, the Petri net shown in Fig. 6 can be constructed. The control objective is to keep the places $S = \{p_5, p_6, p_7\}$ marked with only one token and can be expressed as $\mu(S) \leq 1$. Assume that one servomotor is already in the critical section and consider the place p_3 which corresponds to the polyhedral region P_3 in Fig. 5. All the transitions that lead to the unsafe regions can be disabled by selecting the control action `stop`. Therefore the Petri net supervisor will disable the transitions t_5, t_6 , and t_7 . More details for the supervisory control of Petri nets can be found in [10].

Additional approximations can lead to existing solutions that have been derived using logical models. Assume for example that only the angular position can be used for feedback by the supervisor and the angular velocity is viewed as a disturbance with a known upper bound. In this case the behavior of each servomotor can be described as a first order linear differential equation (inclusion) and the overall system can be modeled by a hybrid automaton [1]. The methodology presented in the paper can be still used and it leads to the design of very simple partitions and a very efficient methodology for controller synthesis.

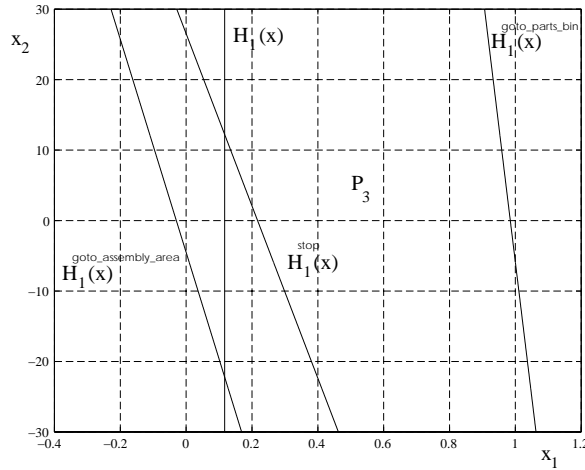


Figure 5: Final partition

6 Conclusions

A new approach for control of hybrid systems was illustrated via a robotic manufacturing system. This work was motivated by previous results concerning the approximation of hybrid systems by discrete-event systems. The class of systems studied is the class of piecewise-linear systems and is sufficiently general to describe interesting engineering applications. The discussion in the paper is mainly focuses on the study of the nondeterministic nature of the discrete-event approximations. The notion of quasideterminism is proposed as an alternative goal in the case when it is difficult to generate a deterministic approximation. Petri nets were selected for the modeling of the abstracting discrete-event system because of the computational advantages they offer for supervisory design methods that enforce convex constraints.

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] P. Antsaklis and K. Passino, editors. *An Introduction to Intelligent and Autonomous Control*. Kluwer Academic Publishers, 1993.
- [3] P. Antsaklis and K. Passino. Introduction to intelligent control systems with high degrees of autonomy. In P. Antsaklis and K. Passino, editors, *An Introduction to Intelligent and Autonomous Control*, pages 1–26. Kluwer, 1993.

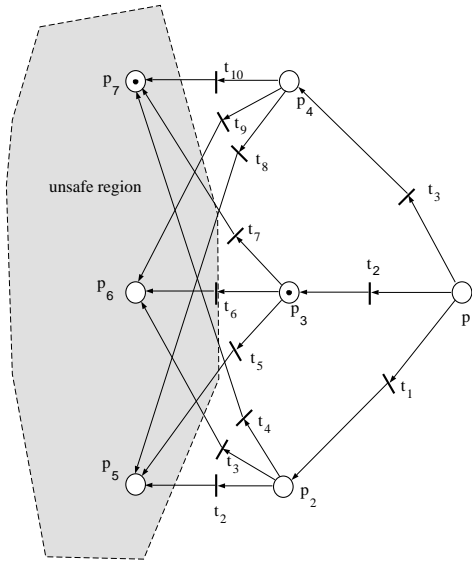


Figure 6: Petri net model

- [4] P. Antsaklis, J. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 366–392. Springer-Verlag, 1993.
- [5] M. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, MIT, 1995.
- [6] 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.
- [7] M. Heymann, F. Lin, and G. Meyer. Synthesis and viability of minimally interventive legal controllers for hybrid systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 8(2):105–135, 1998.
- [8] X. Koutsoukos, K. He, M. Lemmon, and P. Antsaklis. Timed Petri nets in hybrid systems: Stability and supervisory control. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 8(2):137–173, 1998.
- [9] J. Moody. *Petri Net Supervisors for Discrete Event Systems*. PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1997.
- [10] J. Moody and P. Antsaklis. *Supervisory Control of Discrete Event Systems using Petri Nets*. Kluwer Academic Publishers, 1998.
- [11] 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.

- [12] G. J. Pappas and S. Sastry. Towards continuous abstractions of dynamical and control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 329–341. Springer, 1997.
- [13] K. Passino. *Analysis and synthesis of discrete event regulator systems*. PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1989.
- [14] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In P. Antsaklis, W. Kohn, A. Nerode, and S. Shastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 359–369. Springer, 1995.
- [15] J. Raisch and S. O’Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):568–573, 1998.
- [16] M. Sain. *Introduction to Algebraic System Theory*. Academic Press, 1981.
- [17] G. Saridis. Architecture for intelligent controls. In M. Gupta and N. Sinha, editors, *Intelligent Control Systems: Theory and Applications*, pages 127–148. IEEE Press, 1996.
- [18] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.
- [19] E. Sontag. Remarks on piecewise-linear algebra. *Pacific Journal of Mathematics*, 92(1):183–210, 1982.
- [20] 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.
- [21] J. Stiver. *Analysis and design of hybrid control systems*. PhD thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, 1995.
- [22] J. Stiver, P. Antsaklis, and M. Lemmon. Interface and controller design for hybrid control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 462–492. Springer, 1995.
- [23] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4607–4612, San Diego, CA, December 1997.
- [24] W. Wonham and P. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. of Control and Optimization*, 25(3):637–659, May 1987.