An Approach to Hybrid Systems Control Applied to Clocks

Xenofon D. Koutsoukos and Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

1 Introduction

Hybrid systems incorporate both continuous components, described by differential or difference equations, and digital components governed by digital control programs. The key features that characterize these systems are the mixture of continuous and discrete variables, and the coupling of the *time-driven* and *event-driven* dynamics of the continuous and discrete-event parts respectively. Recently, attention has focused on a particular class of hybrid systems in which the continuous dynamics are governed by the differential equation $\dot{x}(t) = c$, where $c \in \Re^n$ [1], [2], [3], [4], [5]. In [3] hybrid systems with continuous dynamics described by first order integrators are used for control of batch processes.

In this paper, the discrete-event part of the hybrid plant is modeled using timed Petri nets. For DES control, Petri nets modeling formalism offers some advantages over finite automata, the exploitation of which seems also useful for hybrid systems control. The hybrid plant is controlled by a supervisor which determines a timed sequence of events to be followed by the discrete event part of the system. The explicit introduction of time in the discrete event part of the hybrid plant and in the supervisor is necessary because of the coupling of time-driven and event-driven dynamics. Petri nets are known to be effective for modeling both *untimed* and *timed* DESs and can capture the discrete behavior of the hybrid system and relate it to the evolution of time. Ordinary Petri nets have been used before for modeling of hybrid systems in [6].

The paper presents some preliminary work for hybrid systems with clocks and is organized as follows. In Section 2, the general model describing the hybrid plant is presented. In Section 3 the supervisor and its tasks are presented. Finally, Section 4 outlines the design procedure for the supervisor, illustrated by an example modeling resource contention.

2 Hybrid Plant

The hybrid plant consists of the discrete event part that models the physical modes of operation of the system, the continuous part which describes the dynamics of the system at each mode, and the interface consisting of an actuator and a generator [7] that governs the interaction between the two parts.

2.1 Discrete-Event Part

An *ordinary Petri net structure* [8], [9], [10] is a triple, $N = (P, T, \mathcal{E})$, where P is a finite set of places, T is a finite set of transitions, and $\mathcal{E} \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs connecting places to transitions and vice versa. A *marking* is a map μ from the set of places onto the nonnegative integers and is represented by a number of tokens at each place.

A *labeled Petri net* is a 5-tuple $G = (N, \Sigma, l, \mu_0, F)$ where $N = (P, T, \mathcal{E})$ is an ordinary Petri net, Σ is a finite set of events, $l : T \rightarrow \Sigma$ is a labeling function that assigns an event to each transition, μ_0 is an initial marking, and F is a finite set of final markings. The *L-type language* (also called *marked behavior*) represents all evolutions of the labeled net that reach a terminal state or states.

The introduction of time to Petri nets can be done by assigning time either to transitions or to places, resulting to *timed transition Petri nets* (TTPNs) or to *timed place Petri nets* (TPPNs). Both TPPNs and TTPNs have been studied in [11] to model manufacturing processes. In [12] a timed Petri net, called *programmable timed Petri net* has been proposed as a modeling formalism for hybrid systems with clocks.

In this preliminary work we consider TTPNs, we attach a deterministic time delay to each transition which can be controlled. Because of the time constraints in the operation of the system, lower and upper bounds may be applied to this time delay. It is the supervisor's task to compute a valid time delay for each transition so that the continuous specifications of the hybrid plant will be satisfied. The computation of the time delays will be performed by solving an optimization problem. In the case of clocks, the optimization problem becomes a linear programming problem. A key idea for the efficiency of the proposed approach is that only the periodic behaviors that satisfy discrete event specifications need be examined, reducing considerably the size of the optimization problem. The approach is explained in Section 4.

To model a hybrid plant we think of the transitions as jobs described by differential equations. Let τ_i denote the operation time associated with the transition t_i, τ_i corresponds to the time delay of the operation assigned to transition t_i. During this time interval the continuous state of the hybrid plant evolves according to the differential equation $\dot{x} = c_i$, $c_i \in \Re^n$. The first transition models the initialization of the system, or in systems theory terminology, the transient response. The processing time of this transition and its effect on the system may be studied independently. In the following we are interested in studying the hybrid system in the "steady state".

The markings and constraints on the markings of the Petri net represent the available resources and the logical constraints that have to be satisfied during the operation of the hybrid system. In the context of the hybrid plant we think of the labeled TTPN as an input-output system. The inputs come from the supervisor representing the time instants for mode changes and from the continuous part through the generator representing uncontrollable plant events. The output of the Petri net is applied through the actuator to the continuous part affecting the evolution of the continuous state. The state (marking) of the Petri net do not affect directly the continuous part of the system, the evolution in time of the discrete state represents the structural properties the hybrid system must posses during its operation.

For completeness, we introduce also a final transition corresponding either to the shutdown or to the reset of the system. During the firing of this transition the continuous state may exhibit discontinuous jumps. Intutively, the processing time associated with the initial transition will depend on whether it followed a shutdown or a reset.

2.2 Continuous Part

The continuous state of the hybrid plant evolves in $X \in \Re^n$ and is described by first order integrators

$$\dot{x}(t) = c_i, \quad T_k \leq \tau_i \leq T_{k+1}$$

where $c_i \in W \subset \Re^n$ a finite set of control vectors and T denotes the global time.

Definition [13] A set C is said to be a *finitely generated cone* if it has the form

$$C = \left\{ x : \; x = \sum_{j=1}^{r} \tau_j c_j, \; \tau_j \geq 0, \; c_j \in \Re^n, \; j = 1, \ldots, r \right\}$$

Assumption In the following, we assume that the finitely generated cone by the set W coincides with the continuous state space X. This assumption guarantees that continuous specifications such as state targeting can be satisfied.

2.3 Interface

In [7] an interface was used to provide means for communication between the continuous plant and the DES controller. In this approach a similar interface is necessary in the hybrid plant to convert continuous time signals to events and vice versa. The interface consists of two subsystems, the generator γ_p and the actuator α. The generator γ_p issues plant events depending on the continuous state. In particular, a set of hypersurfaces is specified and whenever the plant state crosses a hypersurface a plant event is issued. From the view of the discrete event part, the plant events will be considered as uncontrollable events.

A generator γ_c will be used also to provide means of communication between the continuous part of the hybrid plant and the supervisor. The generators γ_p and γ_c are in general different. For example, consider the situation of high temperature in a chemical process. The lack of the plant resources to keep the temperature at a high level may be modeled with an uncontrollable plant event. If the system is not allowed to switch to a control vector that will further increase the temperature, the supervisor must be also informed, but the event will not necessarily be the same as the plant event.

Finally, the actuator α converts the timed sequence of events generated by the supervisor to a piecewise constant function of time applied to the continuous part. We assume that the actuator and the generators respond instantaneously to their inputs.

3 Supervisor

The supervisor has two main tasks. First, to determine sequence of events that satisfy specifications imposed on the discrete-event part of the hybrid plant and second, to compute the inter-event durations to satisfy specifications on the continuous part of the plant. In untimed Petri nets one can prohibit controlled transitions from firing, but cannot force the firing of a transition at a particular instant. In a timed Petri nets controlled transitions are forced to fire, this can be accomplished by considering the firing vectors to be functions of a global time T.

In DES control using Petri nets, modeling and control are interrelated in the sence that the control policy is applied not by a separate structure, but by modifying the plant appropriately, constructing a controlled Petri net [8]. Here, we consider the discrete event part of the hybrid plant and the supervisor to be a controlled timed Petri net. The dynamics of this structure are represented by a timed sequence of transitions or events which exhibits periodic behavior. The operation times for each mode are computed by an other part of the supervisor, which is responsible for solving a linear programming problem based on specifications and information from the plant.

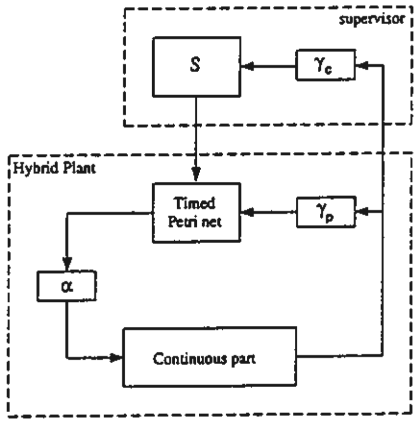

Figure 1: The hybrid plant and the supervisor

4 Supervisor Design

The procedure for the design of the supervisor is outlined and a simple example is presented. First, applying DES control methods we will satisfy the discrete specifications of the hybrid plant. The discrete specifications may include state specifications described by constraints on the marking of the Petri net of the form $L\mu_p \leq b$, sequential specifications expressed by an L-type language, and specifications to ensure properties as liveness, boundedness, and deadlock avoidance of the Petri net. In [14], [15] feedback control based on place invariants is used for supervisory control of Petri nets.

The next step is to identify all the periodic behaviors the controlled Petri net can generate. The union of all the minimum cycles must be computed by expanding the coverability tree of the Petri net. The approach presented here does not intend to derive new results for DES control using Petri nets, but to use the already existing ones. Thus, we may restrict ourselves to classes of Petri nets, for which these control problems can be solved. Also we assume that all the legal periodic behaviors can be identified and further, structural properties as liveness, boundedness, and deadlock avoidance can be studied. For an extended survey on Petri net method for DES control look at [8].

Assume now that the controlled Petri net issues a timed sequence of events that satisfies the discrete specifications of the form $\sigma(T) = t_{init}pt_f$ where p consists of one or more minimum different cycles. The behavior of the continuous part of the hybrid plant will be studied for this particular sequence by assinging a operation time to each transition as it was explained in Section 2. A reasonable continuous specification is to drive the continuous state to a region in the state space represented by a convex set $\mathcal{T}$. The supervisor will compute the operation time of each transition by solving a linear programming problem formulated as follows

$$min\ a^T\tau$$

$$subject\ to : \begin{cases} x_f \in \mathcal{T} \\ l \leq \tau \leq u \end{cases}$$

where τ is the vector of operation times to be determined, a is a weighting vector, x_f is the response of the continuous part at the time instant when $\sigma(T)$ is completed, and l, u are vectors representing lower and upper bounds on the processing time of each transition.

Assume now that an additional continuous specification is expressed as a convex constraint $\mathcal{P}$ that the continuous state must satisfy during the operation of the system. If the two continuous specifications cannot be satisfied simultaneously with one cycle of events, additional cycles must be considered increasing the size

X.D. Koutsoukos and P.J. Antsaklis, "An Approach to Hybrid Systems Control Applied to Clocks," Proceedings o f H ybrid S ystems V , pp. 137-144, University of Notre Dame, Notre Dame, IN, September 11-13, 1997.

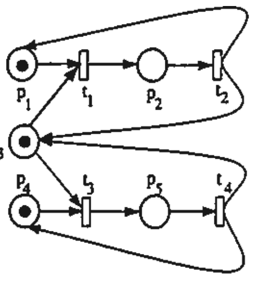

Figure 2: Timed transition Petri net describing the discrete part of the hybrid plant.

of the linear programming problem. When it becomes inefficient to solve a large linear programming problem an alternative method can be considered. Select a sequence of points $x_0, x_1, \ldots, x_m$ in the continuous state space with the following properties: (i) $x_i \in \mathcal{P}$, $i = 1, \ldots, m-1$, $x_m \in \mathcal{T}$ and (ii) we can drive x_i to x_{i+1} with one cycle of events. Such a sequence of points which exhibits "optimal" properties can constructed using an A^* algorithm. In [16] it is shown how to specify a "heuristic function" so that A^* exhibits desirable computational properties. Future work will study if the application of such an algorithm is more efficient than solving a large linear programming problem.

4.1 Example: Hybrid System Describing Resource Contention

Consider the case of two different processes that each needs the same resource to carry out their operations. This is a conflict situation which stems from the resource contention. More specifically, assume that each process consists of two different operations which are described by first order integrators. Transition t_2 denotes the end of the first process (see Fig. 2) and must follow t_1; similarly for t_4 and t_3. The system can be described by the hybrid model in Fig. 1. Assume that the state of the resource is described by the continuous variable x_2 and that with proper selection of the axes, when $x_2 < 0$ the first process consisting of the operations t_1 and t_2 can use the available resources; and when $x_2 > 0$ the second process consisting of t_3 and t_4 is allowed to use the same resources. The continuous part of the hybrid plant consists of a set of first order integrators

$$\dot{x} = c_i \in \Re^2, \quad i = 1,2,3,4.$$

$$C = [c_1, c_2, c_3, c_4] = \begin{bmatrix} 0.5 & -1 & -1 & 1 \\ 1 & 1 & -0.4 & 1 \end{bmatrix}$$

The discrete event part of the plant is described by the Petri net shown in Fig. 2. The policy for the resolution of the conflict will be based on the continuous data. We consider a partition of the state space $\Re^2$ formed by the equation $x_2 = 0$. We require for safety reasons that while the first process is active, x_2 cannot be positive, i.e. the second process cannot use the resources and while the second process is active x_2 cannot be negative. The control objective is to drive the state from a known initial condition

$$x_0 \in \mathcal{A}, \quad \mathcal{A} = \left\{ x \in \Re^2 : \begin{bmatrix} -1.1 \\ -1.1 \end{bmatrix} \leq x \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

to the convex region $\mathcal{B}$ of the state space (see Fig. 3) where

$$\mathcal{B} = \left\{ x \in \Re^2 : \begin{bmatrix} 1 \\ 1 \end{bmatrix} \leq x \leq \begin{bmatrix} 1.1 \\ 1.1 \end{bmatrix} \right\}$$

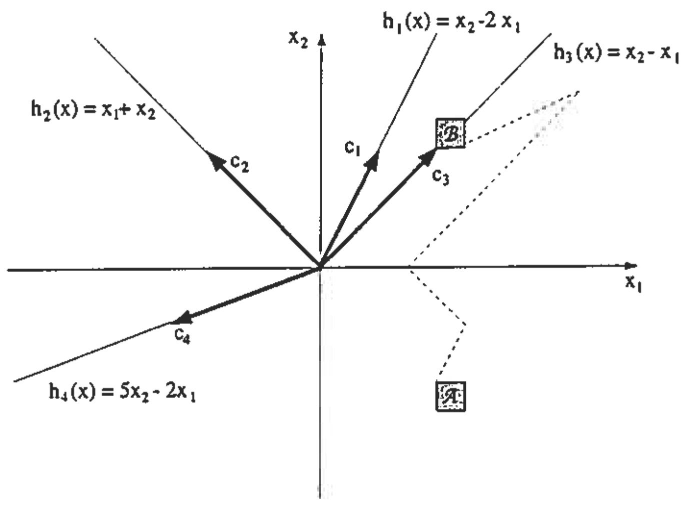

Figure 3: The continuous state space of the system

It is clear from the description of the system that whenever the continuous state crosses the line $h(x) = x_2 = 0$ a plant event is triggered. Thus, the interface γ_p which connects the discrete and the continuous part of the hybrid plant is defined by the line $h(x) = 0$. The interface γ_c which connects the hybrid plant with the supervisor is more complicated. First, the supervisor needs information about the plant events to determine its routing policy to satisfy the safety requirements. Here, we assume that the information about the operating conditions of the plant is captured in the continuous variable x_2. Based on this information the supervisor determines the routing policy to exclude any violation of the safety requirements (e.g. while $x_2 > 0$ the operations t_1 and t_2 cannot be active). The interface γ_c also triggers events representing the completion of the control task, which means that γ_c contains descriptions of the convex regions $\mathcal{A}$ and $\mathcal{B}$. We assume also that as soon as a control objective is satisfied, the supervisor has access to the continuous state, thus the supervisor is informed about the succesful completion of a control objective via an identifier for the convex region the state lies in and the exact value of the state. The actuator α is monitoring the operation of the timed Petri net and for each transition t_i it issues the corresponding control action c_i. For this example, we assume that the discrete event part described by the Petri net of Fig. 2 satisfies the discrete specifications of the system. The first step in the design of the supervisor is to determine the coverability tree of the controlled Petri net.

The incidence matrix for the the Petri net of Fig. 2 is

$$D = D^+ - D^- = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

and the initial marking is $\mu = [1,0,1,1,0]'$. Following the algorithm presented in [11] the coverability tree is shown in Fig. 4.

From the coverability tree is clear that the minimum cycles are t_1t_2 and t_3t_4. Combinations of the minimum cycles like $t_1t_2t_3t_4$ are valid only if they satisfy the safety criterion for the continuous state x_2. This safety constraint can be easily implemented by disabling the transition t_1 (t_3) when $x_2 > 0$ ($x_2 < 0$). Consider the case where because of failure of the control law, the safety constraint is violated, for example x_2 becomes positive while t_1 is active. This situation can be modelled as a failure of an unreliable machine [11] and can

$$\begin{tikzcd}
& (1\,0\,1\,1\,0) \arrow[dl, "t_1"'] \arrow[dr, "t_3"] & \\
(0\,1\,0\,1\,0) \arrow[d, "t_2"'] & & (1\,0\,0\,0\,1) \arrow[d, "t_4"] \\
\begin{array}{c}(1\,0\,1\,1\,0)\\ \text{old}\end{array} & & \begin{array}{c}(1\,0\,1\,1\,0)\\ \text{old}\end{array}
\end{tikzcd}$$

Figure 4: The coverability tree

be caused in our case by exogenous disturbances.

Since we cannot apply the same control policy in the two different regions of the state space, we will satisfy the control objective for the continuous state, that is drive the state from $x_0 \in \mathcal{A}$ to region $\mathcal{B}$, in two steps. In the first step, we will drive the state from $x_0 \in \mathcal{A}$ to the switching surface $h(x) = x_2 = 0$ and then to the convex region $\mathcal{B}$. Some observations are now in order. First, not all points of the line $h(x) = x_2 = 0$ can be driven from the switching surface to $\mathcal{B}$; furthermore, we must insure that x_2 will not change sign while t_1 is active.

Define the matrices $C_{1,2} = [c1, c2]$, $C_{3,4} = [c_3, c_4]$. The finitely generated cones by the matrices $C_{1,2}$ and $C_{3,4}$ are

$$\mathcal{C}_{1,2} = \{x \in \Re^2 : x = \tau_1 c_1 + \tau_2 c2, \ \tau_1, \tau_2 \geq 0\}$$

and

$$\mathcal{C}_{3,4} = \{x \in \Re^2 : x = \tau_3 c_3 + \tau_4 c4, \ \tau_3, \tau_4 \geq 0\}$$

Let $x_0 = [1,1]' \in \mathcal{A}$ be the initial condition and $\pi_1 = \{x \in \Re^2 : \ h(x) = x_2 < 0\}$, $\pi_2 = \{x \in \Re^2 : \ h(x) = x_2 > 0\}$ be the two disjoint regions formed by the partition of the state space. The set of all reachable states from x_0 while the first process (cycle $t_1 t_2$) is active is $\mathcal{R} = (x_0 + \mathcal{C}_{1,2}) \cap \pi_1$ and define $\mathcal{M} = \mathcal{R} \cap \partial \pi_1$. From simple geometry, we have $\mathcal{M} = \{x \in \Re^2 : \ 0 \leq x_1 \leq 1.5 \wedge x_2 = 0\}$. Observe that for any $x \in \mathcal{M}$, $\mathcal{B} \subset x + \mathcal{C}_{34}$. Both steps can be expressed using similar linear programming problem formulations.

Step 1: Drive the state from $x_0 \in \mathcal{A}$ to $\mathcal{M}$

$$min\ \tau_1 + \tau_2$$

$$subject\ to : \{ \ x_m = \tau_1 c_1 + \tau_2 c_2 + x_0 \in \mathcal{M}$$

Step 2: Drive the state from $x_m \in \mathcal{M}$ to $\mathcal{B}$

$$min\ \tau_3 + \tau_4$$

$$subject\ to : \{ \ x_f = \tau_3 c_3 + \tau_4 c_4 + x_m \in \mathcal{M}$$

The solution of the linear programming problem gives

$$\tau_1 = 0.5, \ \ \tau_2 = 0.5; \ \ \tau_3 = 1.4333, \ \ \tau_4 = 1.0833$$

Thus, the timed sequence of the controller is $(t_1, \tau_1)(t_2, \tau_2)(t_3, \tau_3, (t_4, \tau_4)$ and the trajectory of the continuous state is shown in Fig. 3.

References

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

[2] Yonit Kesten, Zohar Manna, and Amir Pnueli. Verifying clocked transition systems. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 13–40. Springer, 1996.

[3] Michael Tittus. *Control Synthesis for Batch Processes.* PhD thesis, Control Engineering Lab., Chalmers University of Technology, Göteborg, Sweden, 1995.

[4] J.A. Stiver, P.J. Antsaklis, and M.D. Lemmon. An invariant based approach to the design of hybrid control systems containing clocks. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III, Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 464–474. Springer, 1996.

[5] B. Lennartson, M. Tittus, B. Egardt, and S. Petterson. Hybrid systems in process control. *Control Systems Magazine*, 16(5):45–56, October 1996.

[6] Philippos Peleties and Raymond DeCarlo. Analysis of hybrid systems using symbolic dynamics and Petri nets. *Automatica*, 30(9):1421–1427, 1994.

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

[8] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Journal of Discrete Event Dynamic Systems*, 7(2):151–190, April 1997.

[9] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of IEEE*, 77(4):541–580, 1989.

[10] J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[11] A.A. Desrochers and P.Y. Al-Jaar. *Applications of Petri Nets in Manufacturing Systems.* IEEE Press, 1995.

[12] Michael Lemmon, Kevin He, and Christopher J. Bett. Modeling hybrid control systems using programmable Petri nets. In *3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*, Reims, France, March 1998. Submitted.

[13] D.P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1995.

[14] J.O. Moody and P.J. Antsaklis. Supervisory control of Petri nets with uncontrollable/unobservable transitions. In *Proceedings of the 35th Conference on Decision and Control*, pages 4433–4438, Kobe, Japan, December 1996.

[15] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1):15–28, 1996.

[16] K.M. Passino and P.J. Antsaklis. A metric space approach to the specification of the heuristic function for the A^* algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):159–166, 1994.