Timed Petri nets in Hybrid Systems: Stability and Supervisory Control [*]

Xenofon Koutsoukos[†], Kevin He, Michael Lemmon and Panos Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

In this paper, timed Petri nets are used to model and control hybrid systems. Petri nets are used instead of finite automata primarily because of the advantages they offer in dealing with concurrency and complexity issues. A brief overview of existing results on hybrid systems that are based on Petri nets is first presented. A class of timed Petri nets named programmable timed Petri nets (PTPN) is then used to model hybrid systems. Using the PTPN, the stability and supervisory control of hybrid systems are addressed and efficient algorithms are introduced. In particular, we present sufficient conditions for the uniform ultimate boundness of hybrid systems composed of multiple linear time invariant plants which are switched between using a logical rule described by a Petri net. This paper also examines the supervisory control of a hybrid system in which the continuous state is transfered to a region of the state space in a way that respects safety specifications on the plant's discrete and continuous dynamics.

[*]This work was supported in part by the National Science Foundation grant ECS95-31485.

[†]The first author gratefully acknowledges the fellowship of the Center of Applied Mathematics at the University of Notre Dame for the academic year 1997-98.

1 Introduction

In hybrid systems the behavior of interest is governed by interacting continuous and discrete dynamic processes. There are several reasons for using hybrid models to represent dynamic behavior of interest. Reducing complexity was and still is an important reason for dealing with hybrid systems. For example, in order to avoid dealing directly with a set of nonlinear equations one may choose to work with sets of simpler equations (e.g. linear), and switch among these simpler models. This is a rather common approach in modeling physical phenomena. In control, switching among simpler dynamical systems has been used successfully in practice for many decades. Recent efforts in hybrid systems research along these lines typically concentrate on the analysis of the dynamic behaviors and aim to design controllers with guaranteed stability and performance. The advent of digital machines has made hybrid systems very common indeed. Whenever a digital device interacts with the continuous world, the behavior involves hybrid phenomena that need to be analyzed and understood. Whenever the behavior of a computer program depends on values of continuous variables within that program (e.g. continuous time clocks) one needs hybrid system methodologies to guarantee correctness of the program and the safe operation of the hybrid system; in fact the verification of such digital computer programs has been one of the main goals of several serious research efforts in hybrid systems literature. The investigation of hybrid systems is creating a new and fascinating discipline bridging control engineering, mathematics and computer science; further information on hybrid systems may be found in references (Grossman *et al.*, 1993; Antsaklis *et al.*, 1995; Alur *et al.*, 1996*a*; Antsaklis *et al.*, 1997; Antsaklis *et al.*, 1998; Morse, 1997; Antsaklis and Nerode, 1998).

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.

This paper considers systems that arise when computers are used to supervise or synchronize the actions of subsystems described by continuous dynamics (that involve continuous variables.) Examples of such systems arise in chemical process control, command and control networks, power distribution networks, as well as distributed manufacturing systems. The size and complexity of such systems often requires that the system use a number of distinct operational modes. Consequently, these systems can be viewed as *supervised* systems, in which a high-level discrete (-event) supervisor is used to coordinate the actions of various subsystems so that overall system safety is not compromised. By *safety*, we mean that pre-specified limits or tolerances on the subsystem states are not violated. These supervised systems can be viewed as a *hybrid* mixture of systems with continuous dynamics (continuous variables) supervised by a switching law generated by a (discrete-event) supervisor described by discrete dynamics (discrete variables). As such, our system is properly viewed as a *hybrid dynamical system*, since it mixes two distinctly different types of dynamical processes: continuous and discrete.

In recent years, a variety of models have been introduced for hybrid systems. These models generally describe the continuous part of the system by a set of ordinary differential equations and represent the discrete part of the system by a discrete-event system. The discrete-event model which has been most widely used in the past is the finite automaton. Finite automata provide a particularly convenient method for hybrid system modeling. In (Stiver *et al.*, 1996*c*), the use of finite automata allowed an extension of the logical supervisory control framework to hybrid systems. A timed automata structure known as the hybrid automata was developed in (Alur and Dill, 1994*a*; Alur *et al.*, 1996*b*) and permitted the extension of symbolic model checking methods to the verification of real time systems (Henzinger *et al.*, 1995).

In spite of this success, however, there are some significant limitations to using finite automata in the modeling, analysis and synthesis of hybrid control systems. The principle limitation concerns the complexity of such automata when used to design control supervisors, and particularly when used to model concurrent processes. Concurrent systems are systems in which several subsystems operating at the same time. The problem here is that the state space for a finite automaton representing the various discrete operational states that a network of systems can generate will grow in an exponential manner with the number of processes. In other words, many of the techniques developed in (Stiver *et al.*, 1996*c*) and (Henzinger *et al.*, 1995) may not scale well with problem size (Puri and Varaiya, 1994). This means that automata based methods for hybrid modeling have an intrinsic limitation when dealing with highly concurrent processes.

In order to deal with highly concurrent processes, therefore, it is necessary to use discrete-event system models which are better suited to model system concurrency. One such model is the ordinary Petri net (Reisig, 1985). Petri nets can be viewed as a generalization of the finite automaton. A finite automaton is generally represented as a finite directed graph consisting of vertices and arcs between these vertices. The current state of the automaton is represented by the *marking* of the directed graph. For finite automata the marking rules are relatively simple. The Petri net can also be viewed as a directed graph in which there are two different types of vertices; places and transitions. Unlike the automaton, marking rules for Petri nets are more complex and allow the system modeller to synchronize the actions of various parts within the system in a way which is not easily accomplished using a finite automaton. Petri nets provide an excellent tool for easily capturing the inherent concurrency of a complex system as well as providing the means of modeling conflict within the system. In general, a Petri net representation for a concurrent process will be more compact (fewer vertices) than its associated automaton representation and with the use of partial order semantics (McMillan, 1992) it is now possible to search the Petri net's state space in a much more efficient manner than is possible using automata models of the same system. Furthermore recent results in the supervisory control of discrete-event systems using ordinary Petri nets (Moody, 1997) have made it possible to design supervisors in an efficient and transparent manner; and this methodology is used in this paper.

In this paper, we use ordinary Petri net models of hybrid systems to study stability and supervision of such systems. It is shown that Petri nets represent a very powerful tool in the analysis and design of hybrid control systems. In particular, this paper presents two approaches that use a type of timed Petri nets, referred to as *programmable timed Petri nets* to address problems important in the safe supervision of hybrid systems. The remainder of the paper is organized as follows. In Section 2, a brief discussion of existing Petri net approaches to hybrid control systems reported in the literature is presented. In Section 3, programmable timed Petri nets (PTPN) are introduced.

PTPN are used to model and study hybrid control systems throughout the paper. Sections 4 and 5 we discuss in detail two Petri net approaches to hybrid control. The first examines stability of switched systems in the presence of disturbance. The second approach studies the supervisory control of hybrid systems. Related work to supervisory control of hybrid systems has appeared in (Lemmon and Bett, 1996; Koutsoukos and Antsaklis, 1997).

2 Petri Nets in Hybrid Control Systems

Petri nets have been used extensively as tools for the modeling, analysis and synthesis of discrete event systems. As it was discussed in the introduction, Petri net offer advantages over finite automata, particularly when the issues of model complexity and concurrency of processes are of concern. Such advantages are also present when Petri nets are used to model hybrid systems. Later in this paper, in Section 4 and in Section 5, Petri net models are used successfully in the study of hybrid control system stability and supervision. This section briefly overviews several results on hybrid systems that are modeled by Petri nets, with emphasis on results that are more closely related to the ones presented later in this paper.

Peleties and DeCarlo (Peleties and DeCarlo, 1994) presented a model based on the work in (Ramadge, 1990) on the periodicity of symbolic observations of piecewise smooth discrete-time systems. In their work, the continuous plant is approximated by a Petri net and a partition of the state space is defined in with each region of the partition corresponds to a place of the Petri net. A transition between two places exists depending on conditions derived from the continuous dynamics. The analysis is based on the construction of a suitable projection from the state space of the continuous dynamics to the space of the symbolic dynamics. A Petri net supervisor controls the behavior of the plant. The supervisor consists of two Petri nets, which communicate with each other and the Petri net representation of the plant through an event based interface. The first Petri net acts as a marker identifying which of the subsystems is currently activated and the other Petri net executes the actual control/supervision task by selecting the next system structure to be activated. The control objective is to drive the continuous state from between any two regions of the state space as these are defined by the partition. The proposed methodology requires the identification of properties which are invariant to the evolution of the continuous plant. Under the assumption that the regions formed by the partition of the state space satisfy these invariant properties, it is shown that any region is reachable from any region through some switching sequence and the advantages of carrying out the analysis using symbolic dynamics are examined.

In (Lunze *et al.*, 1997) a Petri net model is used as discrete event representation of the continuous variable system. The continuous-variable system is represented by a state-space model $\dot{x} = f(x(t), u(t)),\ y = g(x(t), u(t)),\ x(0) = x_0$. The interface between the continuous variable system and the supervisor consists of two parts, which are called the *quantiser* and the *injection.* The output events are generated by a quantiser, which can be represented by a partition of the state space or the output space. The qualitative description of the state defines a partition of the state space into regions which are called *qualitative states.* The injection associates a control input with each control event. The supervisor represents a mapping of the output event sequence into the input event sequence and therefore. The control aim is to reach a prescribed qualitative state.

The system with the quantiser and the injection is modeled using a class of Petri nets, namely finite state machines. Each place represents a qualitative state and each transition is associated with a Boolean expression with variables the available control inputs. The transition is enabled if the predecessor place is marked and the Boolean expression holds. A transition exists only if there exists an input which can force the continuous state to cross the hypersurface between two qualitative states. An algorithm for supervisory control is proposed where the Petri net structure is used to find a set of input sequences that drive the system to the target qualitative state. The Petri net capture the nondeterministic properties of the discrete-event approximation of the plant; the supervisory control algorithm uses information from the state space model to reduce the nondeterminism intrinsic in the model.

Several other approaches using Petri nets to model hybrid systems have also been reported in the literature. *Hybrid Petri nets* proposed in (Le Bail *et al.*, 1991) combine ordinary and continuous Petri nets. Pettersson and Lennartson (Pettersson and Lennartson, 1995) used Bond graphs to verify both discrete and continuous state specifications of systems described by hybrid Petri nets and compared hybrid Petri net and switched Bond graph modeling using a process example. *High-level hybrid Petri nets* proposed in (Giua and Usai, 1996) are characterized by the use of structured individual tokens (e.g. colors) in the discrete part of the net, and provide a rich modeling formalism which takes advantage of the modular structure of Petri nets. *High level Petri nets including a set of differential equations* were proposed in (Vibert *et al.*, 1997) to model batch processes taking into account fluctuations of continuous variables. *Hybrid flow nets*, an extension of the hybrid Petri nets have been proposed in (Flaus and Alla, 1997). These can be seen as a continuous nonlinear extension of Petri nets and are analyzed using a generalization of the theory of structural invariants. Demongodin and Koussoulas (Demongodin and Koussoulas, 1998) considered a new extension of Petri nets, called *differential Petri nets*. Through the introduction of the differential place, the differential transition, and suitable evolution rules, it is possible to model concurrently discrete-event processes and continuous-time dynamic processes, represented by systems of linear ordinary differential equations.

In the following section, a class of timed Petri nets named *programmable timed Petri nets* is used to model hybrid control systems. The main characteristic of the proposed modeling formalism is the introduction of a clock structure which consists of generalized local timers that evolve according to continuous-time vector dynamical equations. They can be seen as an extension of the approach taken in (Alur and Dill, 1994*b*) and provides a simple, but powerful way to annotate the Petri net graph with generalized timing constraints expressed by propositional logic formulae. In contrast with previous efforts to include continuous processes in the Petri net modeling framework (see references for hybrid nets above), the proposed model still consists of two different kind of nodes, discrete places and transitions, and it preserves the simple structure of ordinary Petri nets. The information for the continuous dynamics of a hybrid system is embedded in the logical propositions that label the different elements of the Petri net graph. We believe that this modeling approach extends the power of Petri net formalism which stems from the simplicity of its evolution rules.

3 Programmable Timed Petri Nets

This section introduces a hybrid system model in which timed Petri nets (Sifakis, 1977) generate the switching logic of the system. In particular, we introduce a *programmable timed Petri net* (PTPN) (Lemmon *et al.*, 1998) which is a timed Petri net whose places, transitions, and arcs are all labeled with formulae representing constraints and reset conditions on the rates and times generated by a set of continuous-time systems called *clocks*. The model can be seen as an extension of the Alur-Dill hybrid system model (Alur and Dill, 1994*a*; Alur *et al.*, 1996*b*).

An ordinary Petri net is a directed graph in which there are two types of nodes; places and transitions. Graphically, we represent the places by open circles and the transitions by bars. Petri nets are often characterized by the 4-tuple, (P, T, I, O) where P is the set of *places*, T is the set of *transitions*, $I \subset P \times T$ is a set of input arcs (from places to transitions), and $O \subset T \times P$ is a set of output arcs (from transitions to places). We denote the *preset* of a transition $t \in T$ as $\bullet t$ and define it as the set of places, $p \in P$ such that $(p, t) \in I$. In a dual manner, we introduce the postset of a transition $t \in T$ as $t\bullet$ and define it as the set of places, $p \in P$ such that $(t, p) \in O$. We define presets and postsets of places in a similar way.

The dynamics of ordinary Petri nets are characterized by the way in which the network marking evolves. The marking $\mu : P \rightarrow \mathbb{Z}^+$ is a mapping from the places onto non-negative integers. The marking $\mu(p)$ denotes the number of *tokens* in the place p (represented graphically by small filled circles). We say that the transition t is *enabled* if $\mu(p) > 0$ for all $p \in \bullet t$. An enabled transition may *fire*. We introduce a firing function $q : T \rightarrow \{0, 1\}$ such that $q(t) = 1$ if t is firing and is zero otherwise. If $\mu(p)$ and $\mu'(p)$ denote the marking of place p before and after the firing of enabled transition t, then

$$\mu'(p) = \begin{cases} \mu(p) + 1 & \text{if } p \in t \bullet \setminus \bullet\, t \\ \mu(p) - 1 & \text{if } p \in \bullet t \setminus t\bullet \\ \mu(p) & \text{otherwise} \end{cases} \tag{1}$$

In ordinary Petri nets, places and transitions represent abstractions of the system "states" and "actions". In practice, however the actions take a finite amount of time to complete (fire). It is therefore necessary to work with timed Petri nets (Sifakis, 1977). In a timed Petri net the firing vector and marking vectors become functions of a global time τ. We denote the timed firing vector as q_τ. It indicates which transitions are in the act of "firing" at time τ. The timed marking vector is denoted as μ_τ. Just as in ordinary Petri nets, we will say that a transition t is enabled at time τ if $\mu_\tau(p) > 0$ for all $p \in \bullet t$. An enabled transition is free to fire. For the timed Petri net, however, the firing of a transition occurs over a time interval $[\tau_0, \tau_f]$. The length of this interval is called the transition's *holding time*. A transition t starts to fire at time τ_0 is said to be *committed* and its firing function $q_{\tau_0}(t)$ is set to unity. During the time that the transition is committed, the network's marking vector is not changed. It is only when the firing is completed at time τ_f that the marking vector is changed according to equation 1 given above. At the time the transition t has completed firing, we also reset the firing function $q_{\tau_f}(t)$ to zero.

The duration of the firing interval (holding time) can be characterized in a variety of ways. Common approaches assume that the holding time is either a fixed constant or a random variable.

In some applications, there is a growing realization that these holding times can be treated as control variables. These times can be controlled by introducing "local" timers which cause transitions to fire when specified conditions *programmed* by the system designer are satisfied. This approach was used for concurrent state machines in (Alur and Dill, 1994*a*). Essentially, this approach characterizes the holding times by logical propositions defined over the times generated by a set of *local clocks.* Petri nets whose holding times are defined in this way will be referred to as *programmable timed Petri Nets* (PTPN).

Let $\mathcal{N} = (P, T, I, O)$ be an ordinary Petri net. We introduce a set, $\mathcal{X}$, of N *local clocks* where the ith clock $\mathcal{X}_i$ is denoted by the triple $(\dot{x}_i, x_{i0}, \tau_{i0})$. $x_{i0} \in \Re^n$ is a real vector representing the clock's offset. τ_{i0} is an initial time (measured with respect to the global clock) indicating when the local clock was started. $\dot{x}_i : \Re^n \rightarrow \Re^n$ is a Lipschitz continuous automorphism over $\Re^n$ characterizing the local clock's rate. Assume that the clock rate $\dot{x}_i$ is denoted by the automorphism f. The *local time* generated by the ith clock will be denoted as x_i which is a continuous differentiable function over $\Re^n$ that is the solution to the initial value problem,

$$\frac{dx_i}{d\tau} = f(x_i) \tag{2}$$

$$x_i(\tau_{i0}) = x_{i0} \tag{3}$$

for $\tau > \tau_{i0}$. We therefore see that the local timers are vector dynamical equations. The local time of the ith timer at global time τ is denoted as $x_i(\tau)$ and the timer's rate is denoted as $\dot{x}_i(\tau)$. We say that the *state* of the ith timer is the ordered pair $z_i(\tau) = (x_i(\tau), \dot{x}_i(\tau))$. The ensemble of all local clock states will simply be denoted as $z(\tau)$.

The interval $[\tau_0, \tau_f]$ over which a transition t will be firing is going to be characterized by formulae in a propositional logic whose atomic formulae are equations over the local times or clock rates of $\mathcal{X}$.

Definition 1 *An atomic formula, p, takes one of the following forms;*

1. *It can be a time constraint of the form $h(x_i) = 0$ or $h(x_i) < 0$ where $h : \Re^n \rightarrow \Re$ is a real valued function. This formula evaluates as "true" when the clock time x_i satisfies the equation.*
2. *The atomic formula p can be a rate constraint of the form $\dot{x}_i = f$ which means that the ith clock's rate $\dot{x}_i$ is equal to the vector field $f : \Re^n \rightarrow \Re^n$.*
3. *Finally, p can be a reset equation of the form $x_i(\tau) = \bar{x}_0$ which says that the ith clock's local time at global time τ is set to the vector $\bar{x}_0$.*

Definition 2 *We define a well-formed formula or wff as any expression generated by a finite number of applications of the following rules;*

1. *Any atomic formula is a wff,*

2. If p and q are wff's, then $p \wedge q$ is a wff.

3. If p is a wff, then $\tilde{}p$ is a wff

The set of all wffs formed in this manner will be denoted as $\mathcal{P}$.

The *syntax* for well formed formulas is defined with respect to an underlying Petri net structure of the form $\mathcal{N} = (P, T, I, O)$ and a set of local clocks $\mathcal{X}$. The local clock state z at time τ is said to *satisfy* a formula $p \in \mathcal{P}$ if p is "true" for the given clock state, $z(\tau)$. The satisfaction of p by $z(\tau)$ is denoted as $\mathcal{N} \models p[z(\tau)]$. The truth of the atomic formula is understood in the usual sense. We say that an atomic formla, $p \in \mathcal{P}$ is satisfied by $z(\tau)$ if and only if the evaluation of that formula is true. We say that $\mathcal{N} \models \tilde{}p[z(\tau)]$ if and only if $z(\tau)$ does not satisfy p. We say that $\mathcal{N} \models p \wedge q[z(\tau)]$ if and only if $\mathcal{N} \models p[z(\tau)]$ and $\mathcal{N} \models q[z(\tau)]$.

Consider an ordinary Petri net, $\mathcal{N} = (P, T, I, O)$ and a set of logical timers, $\mathcal{X}$. A *programmable timed Petri net* (PTPN) is denoted by the ordered tuple, $(\mathcal{N}, \mathcal{X}, \ell_P, \ell_T, \ell_I, \ell_O)$ where the functions $\ell_P : P \to \mathcal{P}$, $\ell_T : T \to \mathcal{P}$, $\ell_I : I \to \mathcal{P}$, and $\ell_O : O \to \mathcal{P}$ label the places, transitions, input arcs, and output arcs (respectively) of the Petri net $\mathcal{N}$ with wffs in $\mathcal{P}$.

4 Qualitative Analysis of Switched Dynamical Systems

This section examines the qualitative behaviour of switched dynamical systems. A switched system is a continuous-time system whose structure changes in a discontinuous manner as the system state evolves into switching sets. More formally, such systems are often represented by equations of the following form

$$\begin{aligned} \dot{x} &= f_{i(\tau)}(x(\tau), w(\tau)) && (4) \\ i(\tau) &= q(x(\tau), i(\tau^-)) && (5) \end{aligned}$$

where $x : \Re \to \Re^n$ and $i : \Re \to \mathbb{Z}^+$ denote the continuous and discrete states of the system, respectively. The signal $w : \Re \to \Re^m$ is an exogenous disturbance. We say that $w \in \mathcal{BL}_\infty$ if $\operatorname{ess\,sup}_\tau \|w(\tau)\| \leq 1$. The continuous dynamics are controlled by a finite collection of N control strategies

$$\mathcal{D} = \{f_1, f_2, \cdots f_N\} \qquad (6)$$

where $f_i : \Re^n \times \Re^m \to \Re^n$ for $i \in \{1, \ldots, N\}$ are locally Lipschitz continuous functions. The discrete state of the system is controlled by a *successor* function $q : \Re^n \times \mathbb{Z}^+ \to \mathbb{Z}^+$ which determines the next possible discrete state $i(\tau)$ at time τ given the current continuous state and the "previous" discrete state $i(\tau^-)$, where $i(\tau^-)$ denotes the left hand limit of i at time τ.

There are a variety of results providing sufficient conditions for the Lyapunov stability of such switched systems (assuming $w = 0$). In (Peleties and DeCarlo, 1991) a single positive definite functional is found which is a Lyapunov function for all subsystems of the switched system. Multiple Lyapunov function methods (Branicky, 1994; Hou *et al.*, 1996) have been developed which apply to a larger set of systems than the single Lyapunov function methods. In certain cases, where the switched system consists of linear time invariant subsystems, it has been suggested that multiple candidate Lyapunov-like functions can be determined by finding feasible points of a linear matrix inequality (LMI) (Pettersson and Lennartson, 1996) (Johansson and Rantzer, 1998). These last results are particularly important because they provide a computational method for checking the sufficient conditions for switched system stability provided in (Branicky, 1994).

The sufficient conditions presented in (Branicky, 1994; Hou *et al.*, 1996) and used in (Pettersson and Lennartson, 1996) to compute candidate Lyapunov functionals provide conditions for switched system stability, which may be very conservative, unless the structure of the switching law is explicitly accounted for. The purpose of this section is to show how such structural information can be extracted from Petri net models of switching logics and how such information is then used to formulate the LMIs whose feasibility provide a sufficient test for a switched system's qualitative behaviour. In particular, we examine a specific example from the power systems field in which we are interested in establishing uniform ultimate bounds on disturbed system behaviour.

4.1 PTPN Modeling of Switched Multi-agent Systems

The programmable timed Petri net provides a compact method of modeling switched systems consisting of several independently operating subsystems. Examples of such systems include networks of robots in a distributed manufacturing system, complex process control systems, distributed command and control systems. In this section we examine a distributed command and control system used to supervise the behaviour of a power distribution system.

Recall that a PTPN is a Petri net, $\mathcal{N}$, labeled with logical propositions defined over the times generated by a set $\mathcal{X}$ of local clocks. A PTPN can be used to model a switched dynamical system in the following manner. The network, $\mathcal{N}$, is used to represent the logical dependencies between mode switches in the successor function. The timers, $\mathcal{X}$, of the PTPN are the dynamical equations associated with the continuous time dynamics of the system. The labels ℓ_P, ℓ_T, ℓ_I, and ℓ_O are chosen to represent conditions on the continuous state for mode switches as well as describing the various switching behaviour within the network.

Let $\mathcal{D} = \{f_1, \ldots, f_N\}$ be a set of N Lipschitz continuous vector fields and let $\mathcal{G} = \{h_1, \ldots, h_M\}$ be a set of smooth hypersurfaces in $\Re^n$. The functions in $\mathcal{G}$ are sometimes referred to as the *guards* of the system. Consider a network $\mathcal{N} = (P, T, I, O)$ and a set of timers $\mathcal{X}$ where the ith timer has rate $\dot{x}_i$, initial time x_{i0}, and reset time τ_{i0}. We label the places, transitions, and arcs of the Petri net $\mathcal{N}$ with wffs defined over the timer states, z_i. In particular, these labels are defined as follows.

- Let $J(p)$ be a subset of $\{1, \ldots, N\}$ associated with place $p \in P$ representing those clocks associated with place p. $\ell_P(p)$ is a wff of the form,

$$\ell_P(p) = \bigwedge_{i \in J(p)} ((\dot{x}_i = f_j) \wedge (\tau_{i0} = \tau)) \tag{7}$$

  This formula is interpreted as follows. When place p is marked, then the timer states, z_i, for all $i \in J(p)$ are reset to satisfy $\ell_P(p)$. In particular, this means that the initial time, τ_{i0}, and the clock rate ,$\dot{x}_i$, are reset to the values specified in the equation. The label $\ell_P(p)$ is therefore used to represent switching of the system's vector field when events occur (i.e. transitions fire).

- $\ell_T(t)$ is chosen to be a tautology in this section. This need not always be the case (see following sections), but in the specific example given below, we will only reset or restrict system states at places and arcs.
- Let $J(p,t)$ be a subset of $\{1,\ldots,M\}$ denote a set of hypersurfaces in $\mathcal{G}$ associated with the input arc, (p,t). $\ell_I((p,t))$ is chosen to be a wff whose truth commits the transition t to firing provided this transition is already enabled. In particular, we confine our attention to wffs of the form
$$\ell_I((p,t)) = \bigwedge_{i\in J(p,t)} (h_i(x(\tau)) < 0) \tag{8}$$
This condition allows t to be committed to firing when the continuous state (at time τ) satisfies the listed set of inequalities with respect to the hypersurfaces in $\mathcal{G}$. We refer to $\ell_I((p,t))$ as the input guard equation.
- $\ell_O((t,p))$ is chosen as a wff whose truth completes the firing of the transition, t, assuming that transition t is enabled and committed. These conditions also take the same form as the input guard equation (8) labeling the network's input arcs.

Use the guidelines mentioned above, we can construct PTPN for switched systems characterized by a generalization of equations (4) and (5). The generalization we consider treats the discrete state i in equation (5) as a vector in $\{0,1\}^N$ rather than a nonnegative integer in $\mathbb{Z}^+$. Let $i \in \{0,1\}^N$ be represented by the vector
$$i = \left[\begin{array}{cccc} i_1 & i_2 & \cdots & i_N \end{array}\right]$$
where $i_j \in \{0,1\}$ for all $j = 1,\ldots,N$ is the jth element of the vector i. We can therefore generalize equations (4) and (5) as follows. Let the mapping f in equation (4) be written as $f = [f_1, f_2, \cdots, f_n]$ where $f_j : \Re^n \times \Re^m \times \{0,1\}^N \rightarrow \Re$ is a scalar function representing the rate of change for the jth continuous state. Also let the mapping q in equation (5) be written as $q = [q_1, q_2, \ldots, q_n]$ where $q_j : \Re^n \times \{0,1\}^N \rightarrow \{0,1\}$ is a scalar function representing change of jth discrete state. We assume f and q are both partial functions of the discrete vector $i \in \{0,1\}^N$ which means that f and q many not exist for all i. We represent the switched system by the equations
$$\begin{aligned} \dot{x}_j(\tau) &= f_j(x(\tau), w(\tau), i(\tau)),\ j = 1,\ldots,n \qquad (9)\\ i_k(\tau) &= q_k(x(\tau), i(\tau^-)),\ k = 1,\ldots,N \qquad (10)\end{aligned}$$
We say the model is *well-posed* if for all $i, i' \in \{0,1\}^N$ such that $f_j(x,w,i)$ and $f_j(x,w,i')$ exist, then $f_j(x,w,i) = f_j(x,w,i')$ whenever the lth components, i_l and i'_l, are marked (i.e. $i_l = i'_l = 1$). This condition ensures that the marking of the lth component of the discrete state i has a unique set of differential equations associated with it.

The original set of switched system equations (4) and (5) can be represented as a special case of equations (9) and (10) in the following manner. Recall that the discrete state i in equation (5)

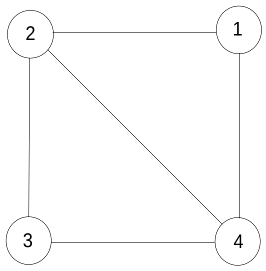

Figure 1: The example power system

is a nonnegative integer between 1 and N, inclusive. We simply associate the kth integer with a boolean vector of length N in which the kth element is 1 and all the rest of the elements are 0.

We can now associate a Petri net $\mathcal{N} = (P, T, I, O)$ with the switched system characterized by equation (9) and (10), by letting the set of places be $P = \{1, 2, \ldots, N\}$ and the set of transitions $T = \{(i,j) \in \{0,1\}^N \times \{0,1\}^N | q(x,i) = j \text{ exists } \}$. The input and output arcs are obtained by examining the transitions in T. The set of input arcs are characterized by the equation $I = \{(p,t) \in P \times T | t = (i,j), i_p = 1\}$ and the set of output arcs by $O = \{(t,p) \in T \times P | t = (i,j), j_p = 1\}$.

The following example illustrates the use of the PTPN in modeling multiagent systems. We consider the 4 node power system shown in figure 1. Each node in the figure represents a generator and the arcs denote the transmission lines between generators.

The continuous state of the ith generator is characterized by its rotor angle, θ_i, and the rotor angle's rate of change $\dot{\theta}_i$. Without loss of generality, we assume that node 4 is a reference node, so we can assume $\theta_4 = 0$ and $\dot{\theta}_4 = 0$ for all time. We therefore represent the continuous-state of the system as a vector in $\Re^6$ of the following form

$$x = \left[\begin{array}{cccccc} \theta_1 & \dot{\theta}_1 & \theta_2 & \dot{\theta}_2 & \theta_3 & \dot{\theta}_3 \end{array} \right]^T \tag{11}$$

The differential equation for the ith generator's angle, θ_i, is

$$\delta p_i = \frac{d^2\theta_i}{d\tau^2} + D_i \frac{d\theta_i}{d\tau} \tag{12}$$

where δp_i is the REAL power's variation about a specified operating level and D_i is a constant determined by the system's operating point and mass properties. From the power flow equation we know that

$$\delta p_i = \sum_j \left(B_{ij} \cos(\theta_i - \theta_j)\theta_i \right) + w_i \tag{13}$$

where B_{ij} is a constant based on the transmission line parameters and w_i is a bounded disturbance signal. From the preceding two equations, we obtain the following linearized system equations.

$$\begin{aligned} \dot{x} &= Ax + Bw && (14) \\ z &= Cx && (15) \end{aligned}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -B_{11} & -D_1 & -B_{12} & 0 & -B_{13} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -B_{21} & 0 & -B_{22} & -D_2 & -B_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -B_{31} & 0 & -B_{32} & 0 & -B_{33} & -D_3 \end{bmatrix} \tag{16}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{18}$$

In this example, the setpoint was chosen to be $x_{set} = [0.1, 0, 0.1, 0, 0.1, 0]^T$. The transmission line parameters were chosen so that $B_{ij} = 10$ for all i, j.

The control objective is to regulate the variation of the generator angle to less than 0.1 for any disturbance $w \in \mathcal{BL}_\infty$. A switching policy is used to help achieve this goal. It is assumed that each generator has two winding ratios to choose from, D_{i0} and D_{i1}, for $i = 1, 2$ and 3. It is assumed that $D_{i0} = 5$ and $D_{i1} = 10$. The ith generator (node) is in discrete state 0 if the first winding ratio is used (i.e. $D_i = D_{i0}$) and is in state (mode) 1 otherwise. There are two conditions which the generators need to respond to.

- The generator must respond to a *fault* which could be caused by a large transient current. If such a transient is detected in the neighborhood of node i then we need to increase the generator winding ratio to protect the generator. The supervisory strategy therefore switches the generator from mode 0 to 1. In this example it is assumed that such a fault is detected when $|\theta_i| > 0.05$.
- At times, however, the generator will need to adjust its output in order to track changing load conditions. If a request to change the load is generated and the node is in mode 1, then we will switch the generator to mode 0. In the context of a detected fault, the switch from mode 1 to mode 0 will be constrained to reset the operating mode 5 seconds after the fault was tripped.

The strategy outlined above can be applied to each generator in a decoupled manner. We can therefore construct a network, $\mathcal{N}$, to represent the logical states of the system. We generalize the discrete state i to a vector

$$i = \begin{bmatrix} i_1 & i_2 & \cdots & i_6 \end{bmatrix}$$

where the kth component represent the marking of the kth place. We let the set of places $P = \{1, 2, \ldots, 6\}$ represent three generators in two different modes in the following way. Let place $2i-1$ represent generator i in mode 0 and place $2i$ represent generator i in mode 1, $i = 1, 2, 3$. It is easy

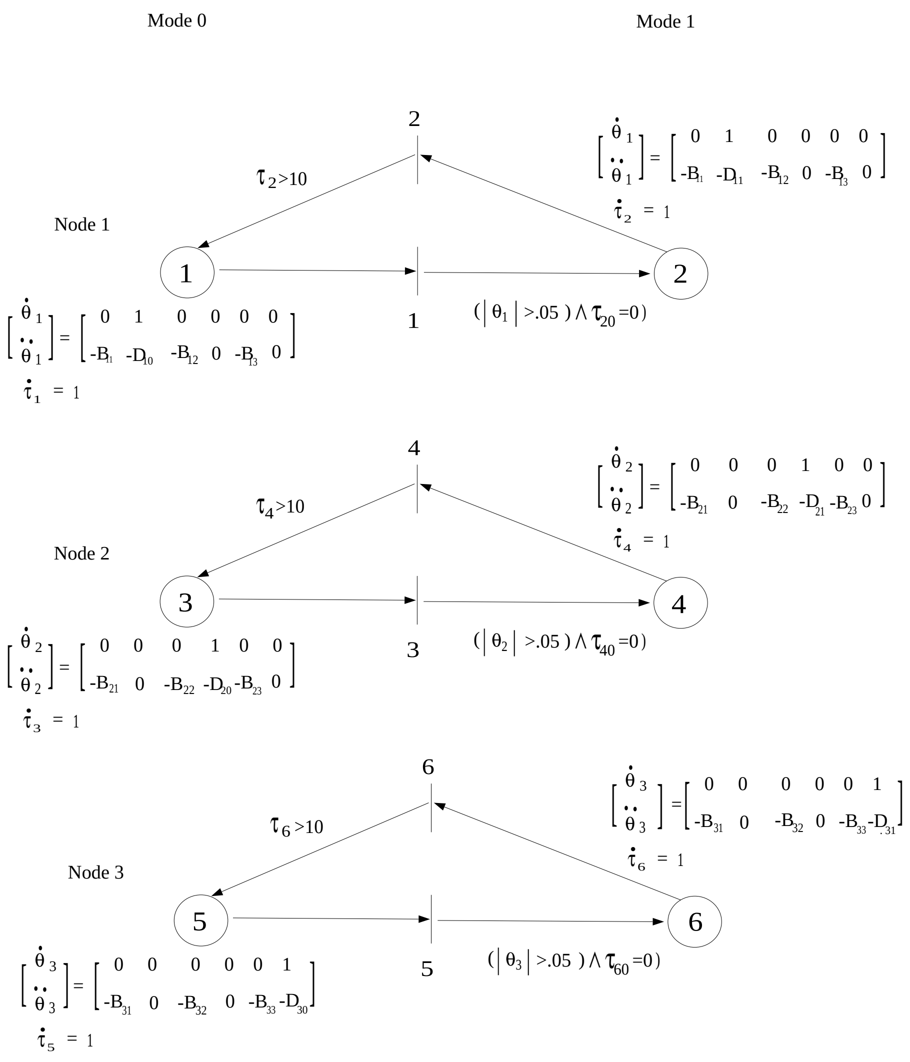

Figure 2: Original Petri net model of example power system

to show that the preceeding construction satisfies the "well-posed" condition. We can therefore associate $\left[\dot{\theta}_i, \ddot{\theta}_i\right]$, $(D_i = D_{i0})$ with place $2i - 1$ and $\left[\dot{\theta}_i, \ddot{\theta}_i\right]$, $(D_i = D_{i1})$ with place $2i, (i = 1, 2, 3)$ as timers. We also associate each place with a local time τ_i. Six transitions are then derived to represent the tripping of the fault alarm and the subsequent resetting of the generator.

The complete PTPN model of the original system is shown in figure 2. The conditions for tripping these alarms and resetting, i.e. $|\theta_i| > 0.05$ and $\tau_{2i} > 10$ appear as logical labels on the input arcs from transition $2i - 1$ to place $2i$ and transition $2i$ to place $2i - 1$ in the PTPN, respectively.

In practice, however, the simple strategy shown in figure 2 will not be able to meet the performance specification. This failure is due to the fact that the generators are coupled by the transmission lines shown in figure 1. We can readily verify the effect of generator coupling through

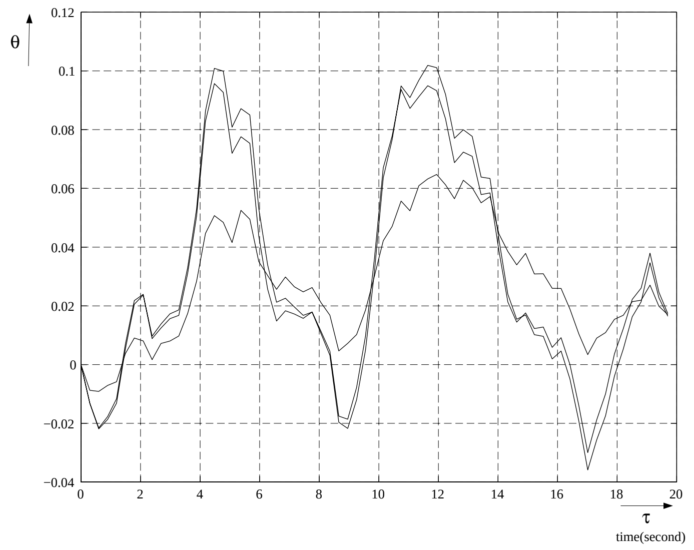

Figure 3: Simulation Results

simulation studies. Figure 3 shows a simulation result in which an impulsive input $\omega(\tau)$ satisfying $\|\omega(\tau)\| \leq 1$ occurs every 10 seconds within the system. The simulation shows that if two neighborhing generator nodes (i.e. nodes 1 and 2 or nodes 2 and 3) are both in mode 1, the rotor angle of the generators will exhibit large variations in excess of the performance requirement in the presence of the disturbance. To achieve the control objective, we therefore implement a supervisory control that prevents adjacent generators from being in mode 1 at the same time. The Petri net model of this controlled system is shown in figure 4. This supervision introduces a control place (also called a monitor) to ensure that adjacent generators enter mode 1 in a mutually exclusive manner. It is the stability of this supervised system that will be studied in the following sections.

4.2 Uniform Ultimate Bounds of Switched Systems

The supervised controller represented in figure 4 attempts to ensure acceptable system behaviour in the presence of bounded faults. The determination of this supervisor was based on "simulation" methods which may not be able to check for all possible failures that can occur in the system. We therefore need to develop a more systematic method of identifying potential system faults that lead to violation of the performance specification without having to resort to exhaustive simulation. This subsection studies the qualitative behaviour of switched systems modeled as programmable timed Petri nets in which exhaustive simulations are not needed. In particular, we use recent results from (Bett and Lemmon, 1997) to compute uniform ultimate bounds for switched systems whose subsystems are linear time invariant (LTI) and whose switching regions form conic sectors in the continuous state space of the system. As noted above there has been considerable interest in studying the Lyapunov stability of such switched systems. In many applications, however, the ability to obtain uniform ultimate bounds on system behaviour may be more crucial. System switching occurs when state trajectories cross specified boundaries (guard conditions) in the state

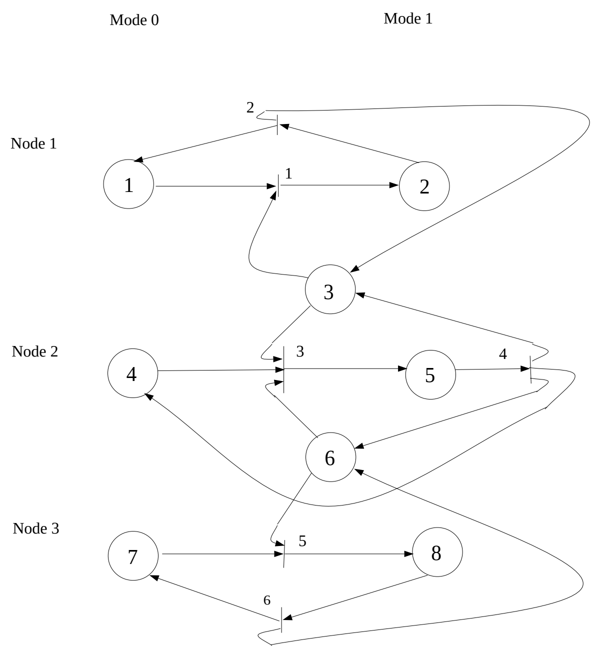

Figure 4: The controlled Petri net model of the example power system

space. From this standpoint, therefore, we are very concerned with being able to bound the *amplitude* of the state trajectory (as measured by the $\mathcal{L}_\infty$ signal norm) in order to control the system's switching behaviour.

Definition 3 *Consider a disturbed system $\dot{x} = f(x, w)$ where $w \in \mathcal{BL}_\infty$. We say that x_0 is an equilibrium point of the undisturbed system if $f(x_0, 0) = 0$. We say that the disturbed system is uniformly ultimately bounded if and only if for all $\epsilon > 0$ there exists a time $T(\epsilon) > 0$ and $\delta > 0$ such that if $x(t_0) < \epsilon$, then $x(T) < \delta$ for all $t > T(\epsilon)$.*

Computing the uniform ultimate bound δ and the dwell time $T(\epsilon)$ for switched systems is well understood. The key result here is a switching lemma stated in (Bett and Lemmon, 1997). Before stating this result we need to introduce some terminology. Consider a linear time invariant system of the form

$$\dot{x} = Ax + Bw \tag{19}$$

$$z = Cx + Dw \tag{20}$$

Let α and β be real non-negative constants. We say that a matrix $P \in \text{FeasRic}(A, B, \alpha, \beta)$, read as Feasible Riccati, if P satisfies the following Riccati inequality,

$$A'P + PA + (\alpha + \beta)P + \frac{1}{\alpha}PBB'P \leq 0 \tag{21}$$

Theorem 1 *(Bett and Lemmon, 1997). Consider two LTI systems $\Sigma_1 = (A_1, B_1, C_1, D_1)$ and $\Sigma_2 = (A_2, B_2, C_2, D_2)$ and consider any finite constants $r \in (0, 1]$ and $\gamma > 0$. Suppose there exists* *positive constants α, β, and ρ and positive definite matrices P_1 and P_2 such that*

$$\begin{aligned} rP_2 &\leq P_1 && (22) \\ \gamma^2 P_1 &\geq C_1' C_1 && (23) \\ \gamma^2 P_2 &\geq C_2' C_2 && (24) \\ P_1 &\in \mathrm{FeasRic}(A_1, B_1, 2\beta + \frac{\alpha}{r}, \alpha) && (25) \\ P_2 &\in \mathrm{FeasRic}(A_2, B_2, \rho, \rho) && (26) \end{aligned}$$

Consider a time $t_s > 0$ and let w, x, and z, be the input, state, and output of the dynamical system which evolves according to system Σ_1 for $0 < t < t_s$ and which evolves according to Σ_2 for $t > t_s$. If

$$t_s > -\frac{1}{2\beta} \log r = T_d \tag{27}$$

then $\|z\|_\infty \leq \gamma$ for all $t > T_d$.

The preceding theorem provides conditions that the LTI subsystems must satisfy in order to ensure that switching transients do not violate the amplitude constraint $\|z\|_\infty < 1$. These results, therefore provide a convenient way of ensuring uniform ultimate boundedness of the switched system. Note that these conditions can also be used to formulate linear matrix inequalities similar in structure to those used by (Pettersson and Lennartson, 1996) and (Johansson and Rantzer, 1998) to ensure Lyapunov stability of switched systems. The obvious implication here is that we should be able to use the preceding theorem to construct linear matrix inequalities whose feasibility is sufficient for the switched system trajectory to possess a uniform ultimate bound.

Unfoldings (Engelfriet, 1991) can be used to study the stability and performance of a switched system represented by a PTPN. A sufficient condition (He and Lemmon, 1997) for the Lyapunov stability and ultimate bounded behaviour of a switched LTI system is that a set of LMIs associated with the fundamental cycles of the system's reachability graph be feasible. The LMIs constructed by these methods are precisely the ones presented in Theorem 1. These fundamental cycles can be found by constructing the reachability graph, an inefficient approach, or they can be systematically constructed from the network unfolding. In the following example, we illustrate how the use of unfoldings can reduce the complexity of searching for fundamental cycles in the PTPN's reachability graph.

We first summarize some basic results related to unfoldings. (see (Engelfriet, 1991) or (Esparza *et al.*, 1996) for more precise definitions and detailed results). Let $\mathcal{N} = (P, T, I, O)$ be a Petri net. Let μ and μ' be two markings of $\mathcal{N}$. We denote $\mu \xrightarrow{t} \mu'$ if μ and μ' represent the markings before and after the firing of enabled transition $t \in T$. A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is an *occurrence sequence* if there exist markings $\mu_1, \mu_2, \ldots, \mu_n$ such that

$$\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \ldots \mu_{n-1} \xrightarrow{t_n} \mu_n$$

μ_n is the marking reached by the occurrence of σ, also denoted by $\mu_0 \xrightarrow{\sigma} \mu_n$. The *reachability graph* of network $\mathcal{N}$ is a labelled graph having the set of reachable markings of $\mathcal{N}$ as nodes and the relations $\xrightarrow{\sigma}$ between markings as edges.

A *node* x is an element of $P \cup T$. A node x_1 precedes node x_2 if there exist an occurrence sequence such that x_2 is reachable from x_1. Given a Petri net $\mathcal{N} = (P, T, I, O)$, we say that two nodes x_1, $x_2 \in P \cup T$ are in *conflict*, denoted by $x_1 \# x_2$, if there exist distinct transitions t_1, $t_2 \in T$ such that $\bullet t_1 \cap \bullet t_2 \neq \emptyset$ and t_i precedes x_i for $i = 1, 2$. We say that, a node x is in *self-conflict* if $x \# x$. This means that there is a node y preceding x such that x can be reached by more than one distinct occurrence sequence from y.

We denote by $Min(\mathcal{N})$ the set $\{p \in P | \bullet p = \emptyset\}$. An *occurrence net* is a finitary(the number of places preceding any $t \in T$ is finite) acyclic net $\mathcal{N} = (P, T, I, O)$ with the initial marking μ_0 such that

- for every $p \in P, | \bullet p| \leq 1$,
- no transition $t \in T$ is in self-conflict, and
- $\mu_0 = Min(\mathcal{N})$.

Let $\mathcal{N}_1 = (P_1, T_1, I_1, O_1)$ and $\mathcal{N}_2 = (P_2, T_2, I_2, O_2)$ be two nets with initial markings μ_{01} and μ_{02}. A *homomorphism* from $\mathcal{N}_1$ to $\mathcal{N}_2$ is a mapping $h : P_1 \cup T_1 \rightarrow P_2 \cup T_2$ such that:

- $h(P_1) \subseteq P_2$ and $h(T_1) \subseteq T_2$ and
- for every $t \in T_1$, the restriction of h to $\bullet t$ is a bijection between $\bullet t$(in $\mathcal{N}_1$) and $\bullet h(t)$ (in $\mathcal{N}_2$), and similarly for $t\bullet$ and $h(t)\bullet$.
- the restriction of h to μ_{01} is a bijection between μ_{01} and μ_{02}.

In other words, a homomorphism is a mapping that preserves the arcs between nodes and preset and postset of transitions.

A *branching process* of network $\mathcal{N}$ with the initial marking μ_0 is a pair $(\mathcal{N}', h)$ such that $\mathcal{N}' = (P', T', I', O')$ is an occurrence net and h is a net homomorphism mapping $\mathcal{N}'$ to $\mathcal{N}$ such that for every $t_1, t_2 \in T'$, if $\bullet t_1 = \bullet t_2$ and $h(t_1) = h(t_2)$ then $t_1 = t_2$. Two branching processes $\beta_1 = (\mathcal{N}_1, h_1)$ and $\beta_2 = (\mathcal{N}_2, h_2)$ of a network are *isomorphic* if there is a bijective homomorphism h from $\mathcal{N}_1$ to $\mathcal{N}_2$ such that $h_2 \circ h = h_1$. Intuitively, two isomorphic branching processes differ only in the names of places and transitions. Furthermore, we say that $(\mathcal{N}_1, h_1)$ contains $(\mathcal{N}_2, h_2)$ if $\mathcal{N}_2 \subseteq \mathcal{N}_1$ and the restriction of h_1 to nodes in $\mathcal{N}_2$ is identical to h_1. We say a branching process of $\mathcal{N}$ is *maximal* if it contains all other branching processes of $\mathcal{N}$.

An *unfolding* is the maximal branching process up to isomorphism associated with a Petri net $\mathcal{N}$. In general, the unfolding of network $\mathcal{N}$ is infinite in size. It is possible, however, to construct finite prefixes of a maximal branching process which enumerate the reachable markings of $\mathcal{N}$ in a computationally efficient manner. This ideas was first discussed in (McMillan, 1992) as a possible solution to state explosion problems and later improved upon in (Esparza *et al.*, 1996).

Consider the occurrence net of a Petri net unfolding. A *configuration* C of this net is a set of transitions satisfying the following conditions,

- $t \in C$ implies if t' precedes t then $t' \in C$, and
- all transitions in C are conflict free.

An occurrence net may have several different configurations. Two configurations which can be marked at the same time are said to be "concurrent". Concurrency can be viewed as an equivalence relation over the set of all configurations of an occurrence net. In particular, this means that the set of configurations can be partitioned into equivalence classes.

The configuration, to some extent, represents a fundamental run of a process. The actual occurrence sequence generated by a network is obtained by interleaving the runs of these configurations. So, configurations provide a very convenient way of decomposing the behavior of a Petri net into simpler structures which make the analysis of the Petri net less complex from a computational standpoint. It is this characteristic of configurations and unfoldings which was successfully exploited in (McMillan, 1992) to address state explosion problems in the verification of asynchronous digital circuits.

Let's return to the example considered above. In this example, we will use network unfoldings to reduce the complexity associated with testing the uniform ultimate boundedness of the switched system. The unfolding of the controlled Petri-net is shown in figure 6. Three configurations are identified in this unfolding. These configurations are represented by the following three sets of transitions; (t_1, t_2), (t_3, t_4), and (t_5, t_6). We label these configurations as c_1, c_2 and c_3, respectively. Concurrency of these configurations induces two equivalence classes. Configuration c_2 forms one of the equivalence classes and configurations c_1 and c_3 form the other equivalence class. We have been able to develop a systematic algorithm (Lemmon *et al.*, 1998) for constructing the fundamental cycles. The basic idea of this algorithm is that, the sequence of firing of all the transitions in a configuration forms a fundamental cycle in the reachability graph.

In this particular example, the two configurational equivalence classes form 5 fundamental cycles represented by the sequence of firing of transitions t_1, t_2, t_5, and t_6. These fundamental cycles are $t_1 - t_2$, $t_5 - t_6$, $t_3 - t_4$, $t_1 - t_5 - t_6$, and $t_5 - t_1 - t_2$. For each fundamental cycle an LMI is constructed. The feasibility of the resulting LMI is easily checked using existing commercial software. In this example, the LMI is feasible thereby showing that the system satisfies the bounded amplitude objective, i.e. $\|z\|_\infty < 0.1$. Simulation results (fig. 5) shows that $\|\theta_i(\tau)\| \leq 0.6$, which clearly validates the correctness of this approach.

It is useful to examine the reachability tree for this example. Constructing the reachability graph requires a total number of 10 nodes to be created and 33 paths to be traced. The unfolding of the network, however, only requires 3 configurations to be identified and 3 paths to be traced. This observation demonstrates that unfolding provides a more efficient method for finding fundamental cycles in PTPN than direct construction of the reachability graph. This empirical finding supports the claims made in (McMillan, 1992) where it was asserted that the computational complexity of constructing the reachability graph is exponential in the number of places and transitions. In contrast, the unfolding generally has a polynomial complexity. This difference is illustrated by comparing the reachability graph for this problem (fig. 7), with the relatively simple occurrence net for this problem (fig. 6). In this figure, the node labels use octal representation of the network marking vector.

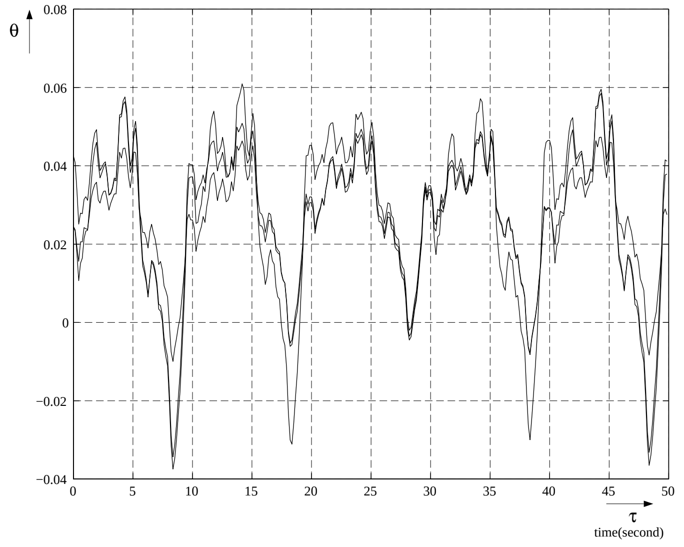

Figure 5: Controlled System Simulation Results

5 Supervision of hybrid systems

In this section, algorithms for supervisory control are presented. Our goal is to determine the switching policy of a hybrid system to drive the continuous state of the system to a prescribed region of the state space. Initially, all the information about the continuous dynamics is disregarded. Logical constraints on the switching policy (for example mutual exlusion constraints) are expressed as specifications on the discrete state of the hybrid system. A DES control method, namely supervisory control of Petri nets based on place invariants (Moody, 1997) is applied to satisfy these discrete specifications. Next, the continuous dynamics are considered and an algorithm based on the notion of a common flow region is used to determine the exact mode switching between the subsystems and the length of time each subsystem will be active.

This section is organized as follows. First, the hybrid plant is modeled by a PTPN. Then, the supervisor is introduced. In particular, a DES control methodology based on the place invariants of the Petri net is briefly discussed. Next, an algorithm based on the notion of a common flow region is presented to satisfy the continuous specifications. Then, this algorithm is applied to affine systems. Finally, we study the special case when the continuous dynamics are described by first order integrators.

5.1 Hybrid Plant

In this section, we consider that the plant is a hybrid dynamical system modeled by the PTPN $(\mathcal{N}, \mathcal{X}, \ell_P, \ell_T, \ell_I, \ell_O)$. $\mathcal{X}$ is a set of continuous-time vector dynamical equations of the form (4)

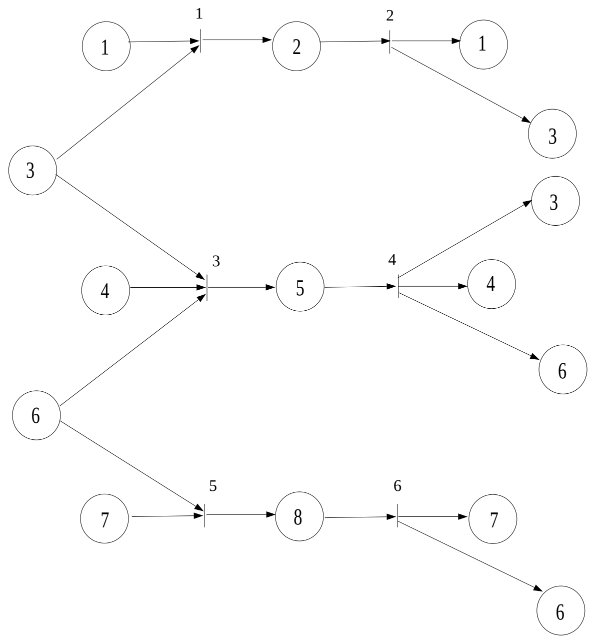

Figure 6: Occurence Network

where the disturbance $w(\tau)$ is assumed to be zero, namely

$$\dot{x}(\tau) = f_i(x(\tau)), \quad i \in \{1, \dots, N\} \tag{28}$$

Equation (28) describes the continuous dynamics of the hybrid system. The network, $\mathcal{N}$, is used to represent the logical dependencies between mode switches in the successor function described by equation (5). The continuous dynamics are controlled using a finite collection of N modes or subsystems

$$\mathcal{D} = \{f_1, f_2, \dots, f_N\}$$

that satisfy the same assumptions as in Section 4. Each f_i corresponds to a control policy.

To represent the logical dependencies between the control policies, we associate with each transition of the net a differential equation of the form (28). This assignment is defined by the labeling function $\ell_T(t) : T \to \mathcal{P}$ which is chosen to be an atomic rate formula of the form $\dot{x} = f_i(x)$. Note that is possible for different transitions to have the same labels. The graph of the Petri net $\mathcal{N}$ describes all the possible mode switches that can occur in the hybrid plant. This is accomplished by defining $\ell_P(p)$ to be tautologies for all places $p \in P$. The input and output guard equations $\ell_I((p,t))$ and $\ell_O((t,p))$ have the same form as equation (8). The set of hypersurfaces $\{h_i\}$ will be determined by the control algorithm to ensure desirable behaviour of the hybrid system.

We introduce now some additional notation that will be useful in formulating the control algorithms later in the section. The firing times of transition t are described by $\sigma^t(n)$, $n \in \mathbb{Z}^+$, where $\sigma^t(k) \in \Re$ represents the duration of the k^{th} firing of transition t. During the time interval $\sigma^t(k)$ the tokens of the input places of transition t do not change. These tokens are put into the output places of t upon the completion of firing of the transition, according to the enabling condition of

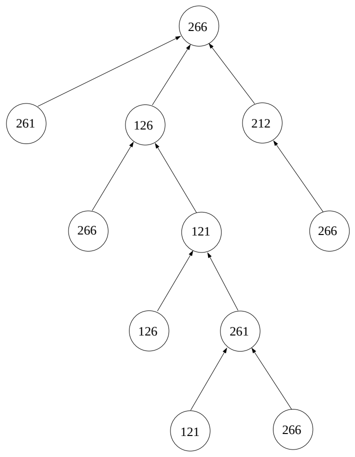

Figure 7: Reachability Graph

the untimed Petri net. We assume that at each time instant exactly one transition is firing. In addition, we assume that $0 < \Delta \leq \sigma^t(n) < \infty$, for some $\Delta \in \Re$, for all firings n and transitions t. We may easily incorporate in our model instantaneous transitions, but these correspond to jumps in the continuous state and will not be considered here. The assumption $0 < \Delta \leq \sigma^t(n)$ eliminates the possibility of infinitely many switchings in a finite time interval.

The control algorithm for the mode selection problem will be based on structural information associated with the places of the Petri net. Routing policies for timed Petri nets are used usually for resolution of conflicts and were introduced in (Baccelli *et al.*, 1992). In our case, we define a mapping $\nu^p(n) : \mathbb{Z}^+ \rightarrow T$ for each place $p \in P$, where $\nu^p(k)$ identifies the particular transition $t \in p\bullet$ to which the kth token to enter place p is to be routed. Note that more than one transition is enabled but only one is allowed to actually fire. If the kth token is routed to $t \in p\bullet$, then the transition t wins the token, which after a firing time of $\sigma^t(k)$ is routed to $t\bullet$, the output places of the transition.

Next, a firing event is defined as the pair (t, τ) which denotes that the transition t starts firing at time τ. Consider the sequence of firing events

$$s = (t_{i_0}, \tau_0), (t_{i_1}, \tau_1), \ldots\ , \quad i_j \in \{1, \ldots, N\}, \text{ for } j = 0, 1, 2, \ldots \tag{29}$$

where j denotes the ordering of the transitions that fire. For example $s = (t_1, \tau_0), (t_3, \tau_1), \ldots$ denotes that t_1 fires at τ_0, next t_3 fires at τ_1 and so on. The firing time intervals are defined by the equation

$$\sigma^{t_i}(k) = \tau_{k+1} - \tau_k \tag{30}$$

At the kth firing of the network, the transition t_i starts firing (at time τ_k) for $\sigma^{t_i}(k)$ time units (until τ_{k+1}). The continuous state of the system during this interval evolves according to

$$\dot{x}(\tau) = f_i(x(\tau)), \text{ for } \tau_k \leq \tau < \tau_{k+1}. \tag{31}$$

The *event projection* and the *timed projection* of the sequence s are defined as

$$\pi_1(s) \ = \ i_0, i_1, i_2, \ldots \tag{32}$$

$$\pi_2(s) \quad = \quad \sigma^{t_{i0}}(k_0), \sigma^{t_{i1}}(k_1), \ldots \tag{33}$$

These are used later in this section.

5.2 Supervisor

The supervisor has two main tasks. The first task is to allow only sequences of events that satisfy specifications imposed on the discrete-event part of the hybrid plant. In particular, consider the net $\mathcal{N}$ of the hybrid system. The objective here is to restrict the possible mode switches of the systems to satisfy additional logical constraints (for example mutual exclusion constraints) that have not been taken into consideration in the modeling phase of the hybrid plant. This can be accomplished without any information about the continuous dynamics. The differential equations of the continuous subsystems associated with the transitions are used to label these transitions. The second task of the supervisor is to enforce firing times that satisfy specifications on the continuous state of the plant. In untimed Petri nets one can prohibit transitions from firing, but cannot force the firing of a transition at a particular instant. In a timed Petri net controlled transitions are forced to fire, as this can be accomplished by considering the firing vectors to be functions of the global time τ. We will show that for a special class of problems, we can first determine the routing policy and then the firing times that will not violate certain conditions imposed on the continuous dynamics. These conditions will be expressed as well formed formulas labeling the input and output arcs of the Petri net.

5.2.1 Supervisory Control of Petri Nets Based on Place Invariants

The first step is to satisfy the discrete specifications of the hybrid plant by applying DES control methods. We assume that the discrete specifications are described by linear inequalities on the marking vector of the Petri net. A methodology for DES control based on Petri net place invariants has been proposed in (Yamalidou *et al.*, 1996; Moody, 1997). A feedback controller based on place invariants is implemented by adding control places and arcs to existing transitions in the Petri net structure. Although the method was developed for ordinary Petri nets, the introduction of time delays associated to each transition will not affect the controlled behavior of the Petri net with respect to the discrete specifications. The supervisor is used to enforce a set of linear constraints on the discrete state of the hybrid plant. These constraints can decribe a broad variety of problems including forbidden state problems, mutual exclusion problems, a class of logical predicates on plant behavior (Yamalidou and Kantor, 1991), conditions involving the concurrence of events, and the modeling of shared resources.

The system to be controlled is the untimed Petri net $\mathcal{N} = (P, T, I, O)$, which is called the plant net. We assume that the plant net has n places and m transitions and its incidence matrix is D_p. The controller net is a Petri net with incidence matrix D_c made up of the transitions of the plant net and a separate set of places. The controlled net is the Petri net with incidence matrix D made up of both the plant and the controller net. The control objective is to enforce the discrete state to satisfy constraints of the form

$$L\mu_p \leq b \tag{34}$$

where μ_p is the marking vector of the plant net, L is an $n_c \times n$ integer matrix, b is an $n_c \times 1$ integer vector, and n_c is the number of 1-dimensional constraints of the type $\sum_{i=1}^{n_p} l_i \mu_i \leq \beta$. Note that inequality (34) is considered componentwise.

This inequality constraint can be tranformed to the following equality by introducing nonnegative slack variables,

$$L\mu_p + \mu_c = b \tag{35}$$

where μ_c is an n_c integer vector which represents the marking of the places of an external Petri net controller. The structure of the controller net will be computed by observing that the introduction of the slack variables forces a set of place invariants on the controlled system. A place invariant is defined by an integer vector x that satisfies

$$x^T \mu = x^T \mu_0 \tag{36}$$

where μ_0 in the initial marking. The place invariants of a net are elements of the kernel of the net's incidence matrix, and they can be computed by finding integer solutions to

$$x^T D = 0 \tag{37}$$

where D is an $n \times m$ incidence matrix. The matrix D_c contains the arcs that connect the controller places to the transitions of the plant net. The incidence matrix D of the closed loop system is given by

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix} \tag{38}$$

and the marking vector μ and the initial marking μ_0 are given by

$$\mu = \begin{bmatrix} \mu_p \\ \mu_c \end{bmatrix} \quad \mu_0 = \begin{bmatrix} \mu_{p0} \\ \mu_{c0} \end{bmatrix} \tag{39}$$

Note that equation (35) is in the form of (36), thus the invariants defined by equation (35) on the system (38),(39) must satisfy equation (37), that is

$$\begin{aligned} X^T D = [L, I] \begin{bmatrix} D_p \\ D_c \end{bmatrix} &= 0 && (40) \\ LD_p + D_c &= 0 && (41) \end{aligned}$$

If D_c is chosen as the solution of equation (41), then the rows of $[L, I]$ are elements of the kernel of the net's incidence matrix. Therefore, they represent place invariants of the closed loop systems and equation (35) is satisfied. Since $\mu(p) > 0$ for all $p \in P$, inequality (34) holds. The above analysis leads to the following proposition presented in (Moody, 1997; Moody and Antsaklis, 1997).

Proposition 1 *The Petri net controller with incidence matrix D_c and initial marking μ_{c0}, which enforces the constraints $L\mu_p \leq b$ when included in the closed loop system (38) with marking (39) is defined by*

$$D_c = -LD_p \tag{42}$$

with initial marking

$$\mu_{c0} = b - L\mu_{p0} \tag{43}$$

assuming that the transitions with arcs from D_c are controllable, observable, and that $\mu_{c0} \geq 0$.

This proposition leads to a controller that enforces the linear constraints $L\mu \leq b$ under the assumption that the controller will enable or inhibit controllable and observable transitions. These results have been extended for handling uncontrollable and unobservable transitions in (Moody and Antsaklis, 1996). In the hybrid systems case, we have associated transitions to coninuous subsystems described by differential equations. It is assumed that the supervisor can force and observe the firing of the transitions. This is accomplished by imposing conditions described by well-formed formulas on the input and output arcs of the transitions, as described in the next section. This work is an effort to incorporate the well-established discrete event control methods of Petri nets into a hybrid systems framework. Linear constraints on the discrete state represent only a small, yet very useful class of state specifications. The supervisor that restricts the behaviour of the controlled Petri net to satisfy such constraints is computed very efficiently and is introduced in the PTPN in a straighforward manner. However, other control policies that enforce more general discrete state specifications can be used. For example, more general supervisor policies have been reported in (Holloway *et al.*, 1997) for solving forbidden markings problems and in (Sreenivas, 1997) to enforce liveness. Actually, this is one of the main advantages of the proposed model. By preserving the simple structure of ordinary Petri nets in the modeling of hybrid systems, it is possible to incorporate various supervisory policies developed for discrete-event systems.

5.2.2 Hybrid Strategy based on Equilibria

In the nonlinear control literature, switching has been used to expand the domain of attraction of operation points in control systems (McClamroch *et al.*, 1997; Guckenheimer, 1995). In the hybrid systems case, we assume that for each control strategy there exists a unique equilibrium point for the resulting continuous subsystem f_i, $i = 1, \dots, N$. Each equilibrium has a domain of attraction associated with it. The idea is to switch at discrete time instants from one mode (subsystem f_i) to another in a way that the system gradually progresses from one equilibrium to another towards the final equilibrium.

This idea can be formalized using an *invariant based approach* for hybrid systems proposed in (Stiver *et al.*, 1995; Stiver *et al.*, 1996*a*). To describe this approach, certain results will first be introduced. A *common flow region* for a given target region, is defined as a set of states which can be driven to the target region with the same control policy. Common flow regions are bounded by invariant manifolds and an exit boundary, so that the state trajectory can leave the common flow region only through the exit boundary.

Definition 4 *For the continuous part of the hybrid plant, the set B is a common flow region for a given region R if*

$$\forall x(\tau_0) \in B, \ \exists \tau_1, \tau_2, \ \tau_0 < \tau_1 < \tau_2$$

such that

$$x(\tau) \in B,\ \tau \leq \tau_1$$

and

$$x(\tau) \in R,\ \tau_1 < \tau < \tau_2$$

subject to

$$\dot{x}(\tau) = f_i(x(\tau))$$

The following two propositions are presented in (Stiver *et al.*, 1995) and give sufficient conditions for the hypersurfaces bounding B and R to ensure that all state trajectories in B will reach R (B, R are regions of the state space $X \subset \Re^n$).

Proposition 2 *Given the following:*

1. *A flow generated by a smooth vector field, f*
2. *A target region, $R \subset X$*
3. *A set of smooth hypersurfaces, $h_i,\ i \in I_B \subset 2^I$ (power set of I)*
4. *A smooth hypersurface (exit boundary), h_e*

such that $B = \{\xi \in X :\ h_i(\xi) < 0,\ h_e(\xi) > 0,\ \forall i \in I_B\} \neq \emptyset$. For all $\xi \in B$ there is a finite time, τ, such that $x(0) = \xi,\ x(\tau) \in R$, if the following conditions are satisfied:

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0,\ \forall i \in I_B$
2. $\exists \epsilon > 0, \nabla_\xi h_e(\xi) \cdot f(\xi) < -\epsilon,\ \forall \xi \in B$
3. $B \cap Null(h_e) \subset R$

The following proposition uses in addition, a cap boundary (bounding hypersurface) in order to obtain a common flow region which is bounded.

Proposition 3 *Given the following:*

1. *A flow generated by a smooth vector field, f*
2. *A target region, $R \subset X$*
3. *A set of smooth hypersurfaces, $h_i,\ i \in I_B \subset 2^I$*
4. *A smooth hypersurface (exit boundary), h_e*

5. A smooth hypersurface (cap boundary), h_c

such that $B = \{\xi \in X : \ h_i(\xi) < 0, \ h_e(\xi) > 0, \ h_c(\xi) < 0, \ \forall i \in I_B\} \neq \emptyset$ and $\bar{B}$ (closure of B) is compact. For all $\xi \in B$ there is a finite time, τ, such that $x(0) = \xi, \ x(\tau) \in R$, if the following conditions are satisfied:

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \ \forall i \in I_B$
2. $\nabla_\xi h_c(\xi) \cdot f(\xi) < 0, \ \forall \xi \in B \cap Null(h_c)$
3. $B \cap Null(h_e) \subset R$
4. *There are no limit sets in $\bar{B}$*

Remark The set of smooth hypersurfaces, $\{h_i, \ i \in I_B \subset 2^I\}$ is a set of smooth functionals $\{h_i : \ \Re^n \to \Re, \ i \in I_B\}$, defined on the state space of the plant. Each functional must satisfy the condition

$$\nabla_x h_i(\xi) \neq 0, \ \forall \xi \in Null(h_i) \tag{44}$$

which ensures that the null space of the functional $Null(h_i) = \{\xi \in \Re^n : \ h_i(\xi) = 0\}$ forms an $n-1$ dimensional manifold separating the state space.

Each of the two propositions above gives sufficient conditions for a set of hypersurfaces to form a common flow region. These hypersurfaces can be either invariant under the vector field of the given control policy or cap boundaries for the given vector field.

The set of all invariant hypersurfaces can be found in terms of $n-1$ functionally independent mappings which form the basis for the desired set of functionals, $\{h_i\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(x)} = \frac{dx_2}{f_2(x)} = \dots \frac{dx_n}{f_n(x)} \tag{45}$$

where $f_j(x)$ is the jth element of $f(x)$ ($f(x)$ is used rather than $f_i(x)$ to avoid subscript confusion).

Remark The hypersurfaces $\{h_i, \ i \in I_B\}$ must be invariant under the vector field f of the given control policy. This can be achieved by choosing them to be integral manifolds of an $n-1$ dimensional distribution which is invariant under f. An $n-1$ dimensional distribution, $\Delta(x)$, is invariant under f if it satisfies

$$[f(x), \Delta(x)] \subset \Delta(x) \tag{46}$$

where $[f(x), \Delta(x)]$ indicates the Lie bracket. Of the invariant distributions, those that have integral manifolds as we require, are exactly those which are involutive (according to Frobenius). This means that

$$\delta_1(x), \delta_2(x) \in \Delta(x) \Rightarrow [\delta_1(x), \delta_2(x)] \in \Delta(x)$$

Therefore by identifying the involutive distributions which are invariant under the vector field f, we have identified a set of candidate hypersurfaces. For details about the relationships between vector fields and invariant distributions, see (Isidori, 1996).

We will now describe a method to determine appropriate cap boundaries and common flow regions. This method is based on Lyapunov functions. Consider the hypersurface $h_c(x)$ that forms a cap boundary for the common flow region B. Assume that there exists an appropriate Lyapunov function $V(x)$ for the vector fields f such that

$$\begin{gathered} V(x) > 0,\ \forall x \in B \\ V(x) \to \infty \text{ as } \|x\| \to \infty \\ \dot{V}(x) < 0,\ \forall x \in B. \end{gathered} \tag{47}$$

Then $\Omega_c = \{x \in \Re^n |\ V(x) \leq c\}$ is bounded and the hypersurface $h_c(x) = V(x) - c$ is a cap boundary candidate. The constant parameter c can be selected appropriately so that $h_c(x)$ bounds the common flow region B.

Based on these results, appropriate cap boundaries can be determined efficiently using Lyapunov theory. Furthermore, the design based on Lyapunov functions will exhibit desirable robustness properties. Note that the task to determine suitable invariant hypersurfaces is very difficult in general. The next proposition gives sufficient conditions for the state to progress from one equilibrium point to another.

Proposition 4 *Let $f_{i_1}, f_{i_2} \in \mathcal{D}$ satisfy the following assumptions*

- *Each f_i is globally Lipschitz and admits an isolated equilibrium point $\bar{x}_i$, and $\bar{x}_i$ is asymptotically stable w.r.t. f_i.*
- *For each f_i there exists an appropriate Lyapunov function $V_i :\ \Re^n \to \Re$ and $\Omega_i = \{x \in \Re^n |\ V_i(x) \leq c_i\}$ such that*

$$\begin{gathered} V(x) > 0,\ \forall x \in \Omega_i \\ V(x) \to \infty \text{ as } \|x\| \to \infty \\ \dot{V}(x) < 0,\ \forall x \in \Omega_i \end{gathered} \tag{48}$$

In addition, assume that $\Omega_{i_1} \cap \Omega_{i_2} \neq \emptyset$ and $\bar{x}_{i_1} \in R' = int(\Omega_{i_1} \cap \Omega_{i_2})$, then for every $x_0 \in \Omega_{i_1}$ there exists a switching sequence

$$s(x_0, \tau_0) = (i_1, \sigma^{t_{i_1}}(k_0)), (i_2, \sigma^{t_{i_2}}(k_1))$$

which drives the state to a region R of the equilibrium point $\bar{x}_{i_2}$.

Proof Let $\Omega = \Omega_{i_1} \setminus \Omega_{i_2}$ and define the hypersurface $h_c(x) = \partial\Omega \cap \partial\Omega_{i_1}$ and $h_e(x) = \partial\Omega \cap \partial\Omega_{i_2}$. Since $\bar{x}_{i_1} \in R'$ is an asymptotically equilibrium point for f_{i_1}, from Proposition 3 , Ω is a common flow region for $R' = int(\Omega_{i_1} \cap \Omega_{i_2})$. Let $\Omega' = \Omega_{i_2} \setminus R$ and define the hypersurface $h_c(x) = \partial\Omega_{i_2}$ and $h_e(x) = \partial R$, then Proposition 3 holds and Ω' is a common flow region for the target region R.

Remark The conditions of Proposition 4 are stronger than the condition of Proposition 3 but they provide a systematic way to check the existence of the cap boundary $h_c(x)$ and of the corresponding common flow region via a search of a suitable Lyapunov function.

In the following, a specific switching sequence will be determined. First notice that only sequences of vector fields that correspond to control policies that satisfy the discrete specifications have to be considered. The control policies that satisfy the discrete specifications are exactly those that are accepted by the controlled Petri net which consists of the plant and the supervisor designed using the methodology based on place invariants (see above). The switching sequences can be determined by identifying the periodic behavior of the controlled Petri net using for example the methods described in the previous section based on the unfoldings of Petri nets.

The underlying Petri net structure, which generates the switching policy offers two important advantages. First, it makes possible to efficiently design the supervisor that satisfy specifications that frequently appear in complex systems such as generalized mutual exlusion constraints. Second, it reduces considerably the search for common flow regions, since only desirable switching strategies generated by the controlled Petri net have to be examined.

The following corollary gives sufficient conditions for a switching sequence generated by the controlled Petri net to drive the continuous state x_0 to a target region of the state space. It is assumed that the initial conditions belong to the region of attraction Ω_{i_0} of the first control policy and that the state progresses towards $\bar{x}_{i_m} \in \Omega_{i_m}$ by allowing switchings to occur on the intersection $\Omega_{i_j} \cap \Omega_{i_{j+1}}$ of consecutive invariant manifolds. In the case when all the pairs of subsystems satisfy Proposition 4, the set $\Omega_{i_j} \cap \Omega_{i_{j+1}}$ will be nonempty and the proof is straightforward.

Corollary 1 *Suppose there exists a switching sequence with event projection $\pi_1(s) = i_0, i_1, \ldots, i_m$ accepted by the controlled Petri net such that every pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 4 . Given a target region R such that $\bar{x}_{i_m} \in int(R)$, there exists switching policy to drive the continuous state from any initial condition $x_0 \in \Omega_{i_0}$ to the region R in finite time. The firing time intervals $\sigma^t(n)$ will be chosen so that the switchings occur while $x \in int(\Omega_{i_j} \cap \Omega_{i_{j+1}})$.*

Remark The condition that every pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 4 can be relaxed by allowing intermediate transitions which will keep the continuous state in the domain of attraction of the equilibrium $\bar{x}_{i_{j+1}}$ of the control strategy $f_{i_{j+1}}$.

The supervisor is implemented by assigning well-formed formulas to the input and output arcs of the controlled Petri net. Let $\{h_i\},\ i = 1, \ldots, n$ be the set of hypersurfaces that bound a region M of the state space X. We can use the following well-formed formula

$$\ell = p_1 \wedge p_2 \wedge \ldots \wedge p_n \tag{49}$$

to describe that $x \in M$, where p_i is a constraint of the form $h_i(x) < 0$. Consider a pair of vector fields $(f_{i_j}, f_{i_{j+1}})$ that satisfy Proposition 4 and let B_{i_j}, R_{i_j} and $B_{i_{j+1}}, R_{i_{j+1}}$ be the corresponding common flow and target regions. From Proposition 4 we have that the target region R_{i_j} coincides with the common flow region $B_{i_{j+1}}$. The switching algorithm is described by the following labeling functions, where $p \in P$ is the place to connect the output arc of t_{i_j} to the input arc of $t_{i_{j+1}}$

- $\ell_P(p)$ is chosen to be a tautology.
- $\ell_T(t_{i_j}), \ell_T(t_{i_{j+1}})$ are chosen to be the atomic rate formulas $\dot{x} = f_{i_j}(x)$ and $\dot{x} = f_{i_{j+1}}(x)$ respectively.

- $\ell_O((t_{i_j}, p))$ is chosen to be a wff of the form (49) representing that $x \in R_{i_j}$.
- $\ell_I((p, t_{i_{j+1}}))$ is chosen to be a wff of the form (49) representing that $x \in B_{i_{j+1}}$

Assuming that transition t_{i_j} is firing, the next transition to fire, $t_{i_{j+1}}$ is determined by the routing policy $\nu^p(n)$ so that the pair $(f_{i_j}, f_{i_{j+1}})$ satisfies Proposition 4. Transition $t_{i_{j+1}}$ will fire only when the firing time intervals $\sigma^t(n)$ lead to true values of the logic formulas $\ell_O((t_{i_j}, p))$ and $\ell_I((p, t_{i_{j+1}}))$. For the initialization of the hybrid system we assume that $\ell_I((p, t_{i_0}))$ is a tautology.

Remark In the case when the Petri net is live and the event projection generated by the controlled Petri net $\pi_1(s)$ is an infinite sequence that satisfies Corollary 1, the hybrid system exhibits a periodic behavior in the sense that the continuous state is visiting periodically neighborhoods of the equilibria.

5.2.3 Affine Systems

A class of systems that satisfy the conditions for supervisory control design of the previous section is the affine systems. They represent physical systems that are described by linear ordinary differential equations with one additional assumption, namely that the input is allowed to take a finite number of prespecified constant values. Let the continuous dynamics be described by

$$\dot{x} = Ax + c_i, \quad i \in \{1, \dots, N\} \tag{50}$$

where $c_i \in W \subset \Re^n$ a finite set of control vectors and the matrix $A \in \Re^{n \times n}$ is Hurwitz. If $f_i(x) = Ax + c_i$, then $\bar{x}_i = -A^{-1}c_i$ is a globally asymptotically stable equilibrium point for $\dot{x} = f_i(x)$. In view of the global asymptotic stability of each equilibrium point (A is Hurwitz), it is clear that Proposition 4 holds for every pair of control inputs. The values for the control input can be selected so that the continuous state can be driven to prescribed regions of the state space. In the following, an example is given to illustrate the approach.

Example: Hybrid System Describing Resource Contention

Consider the case of two different processes that use 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 ordinary differential equations and the switching policy is represented by the Petri net in figure 8. This situation arises frequently in physical systems when different processes share the same resources. We will use this Petri net to describe the switching policy for two examples that follow. The example below is a temperature control system where the continuous dynamics are described by an affine system.

The incidence matrix of the plant net is

$$D_p = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{51}$$

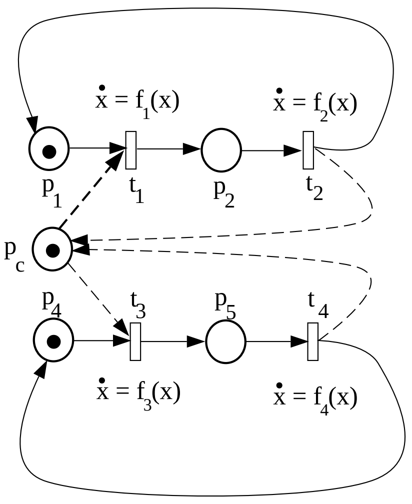

Figure 8: The controlled Petri net of the resource contention example

and the initial condition of the marking vector $\mu_{p_0} = [1, 0, 1, 0]^T$. We consider the mutual exclusion constraint $L\mu_p \leq b$, where $L = [0, 1, 0, 1]$ and $b = 1$. Using Proposition 1 the closed loop system has the incidence matrix

$$D = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \tag{52}$$

and initial condition $\mu_0 = [1, 0, 1, 0, 1]$. The last row of the incidence matrix D represents the Petri net supervisor.

The controlled Petri net is shown in figure 8 where the supevisor is implemented through a place p_c connected to already existing transition (dashed lines).

Example: Temperature Control System

Let a typical temperature control system be described by the electrical circuit shown in figure 9. Here, an electrical analog of the 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 control objective is to control the temperature at a point at the system by applying the heat input at a different point. The temperature control example is used in (Friedland, 1996) to illustrate PID control design. Here, we assume that only discrete levels are available for the current (heat) input u.

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 is allowed to take finitely many discrete values. Consider that the numerical values of the

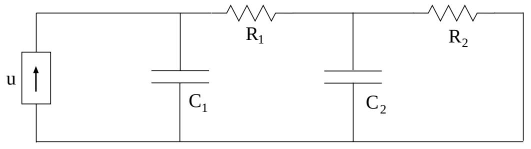

Figure 9: The temperature control system

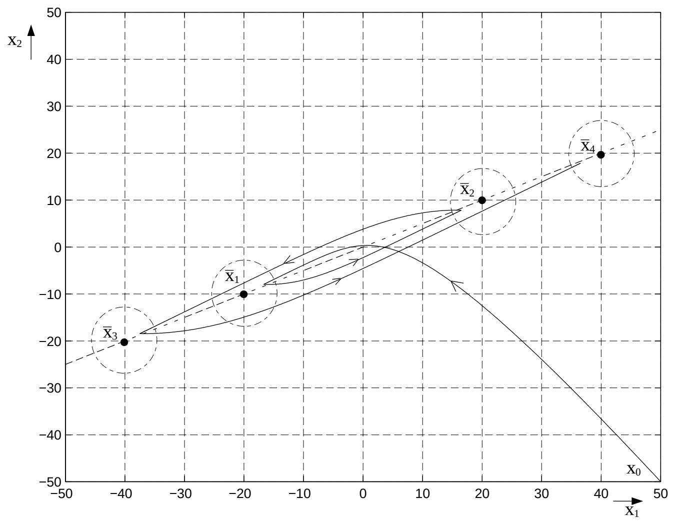

Figure 10: Periodic behavior of the hybrid system

electrical elements are

$$R_1 = R_2 = 1$$
$$C_1 = C_2 = 1$$

Then (using Kirchhoff's laws) the system is described by the state-space equation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \tag{53}$$

Assume that $q_i,\ i = 1, \ldots, N$ are the available discrete levels for the input that correspond to the available control policies. It is easily verified that the matrix A is Hurwitz and therefore $\bar{x}_i = -A^{-1}c_i$ where

$$A = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix}, \quad c_i = \begin{bmatrix} q_i \\ 0 \end{bmatrix}$$

is a globally asymptotically stable equilibrium point for the system $\dot{x} = Ax + c_i$.

We assume that the discrete levels of the heat input are $q_1 = -10, q_2 = 10, q_3 = -20$, and $q_4 = 20$ and that the switching policy is described by the Petri net in figure 8 representing resource contention, where transition t_i corresponds to the control input c_i. In this case, the supervisor control algorithm will determine the mode switches and the firing times so that the continuous state of the hybrid system visits periodically neihborhoods of the equilibria (figure 10). We consider a ball $B_i(\bar{x}_i, r)$ of radius r centered at $\bar{x}_i,\ i = 1, 2, 3, 4$ and we label the input and output arcs of the transitions with the wffs $\ell_O((t_i, p)) = \ell_I((p, t_{i+1})) = x \in B_i(\bar{x}_i, r)$. Then the supervisor allows switchings to occur only when the above logical propositions are true.

5.3 Supervisory Control of First Order Integrators

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 $c \in \Re^n$ (Alur *et al.*, 1993; Tittus, 1995; Stiver *et al.*, 1996*b*; Lennartson *et al.*, 1996). In (Tittus, 1995) hybrid systems with continuous dynamics described by first order integrators are used in the control of batch processes. In the case when the continuous dynamics are described by first order integrators, the previous algorithm cannot be applied since the continuous subsystems do not admit any isolated equilibria. In the following, we present an algorithm based on a similar idea to the case of multiple equilibria. There exists a family of invariant sets for each continuous subsystem. The switching from a subsystem to another occurs at discrete time instants in a way that the state gradually progresses towards the target region. We determine a sequence of events that drive the state to the precribed target region by solving a linear programming problem as in (Lennartson *et al.*, 1996).

5.3.1 Hybrid Plant

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

$$\dot{x}(\tau) = c_i, \ \ i \in \{1, \dots, N\} \tag{54}$$

where $c_i \in W \subset \Re^n$ a finite set of control vectors.

The hybrid plant is described by the PTPN $(\mathcal{N}, \mathcal{X}, \ell_P, \ell_T, \ell_I, \ell_O)$. $\mathcal{X}$ is the set of the differential equations (54), the network $\mathcal{N}$ and the functions labeling the places and transitions of the Petri net are the same as in the previous subsection. The input and output guard functions $\ell_I((p, t))$ and $\ell_0((t, p))$ are equality cnstraints on the independent time variable τ and completely determine the firing times of each transition. Before deriving the supervisor, the following definitions are in order.

Definition 5 *A set $\mathcal{C}$ is said to be a* finitely generated cone *if it has the form*

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

Assumption We assume in the following 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.

Definition 6 *A set $M \subset X$ is said to be* invariant *with respect to the system $\dot{x} = f(x)$ if $x(\tau_0) \in M$ implies $x(\tau) \in M$, $\forall \tau \in \Re$.*

The system $\dot{x} = c$ admits a family of invariant sets described by the equation $x = c\tau + x_0$, which represents a family of parallel lines parametrized in τ. The path from the initial state to the target region will be found by solving a linear programming problem to determine the time interval the state of the system evolves in each particular invariant set.

5.3.2 Supervisor

The procedure for the design of the supervisor is similar to that of the previous section. First, applying DES control methods we construct a controlled Petri net which satisfies the discrete specifications. Assume now that the control objective is to drive the state to the targer region $R \subset X$. The task of the supervisor is to select a control policy accepted by the Petri net, and to decide for how long it should be active.

Proposition 5 *Consider the switching sequence $s(\tau_0, x_0) = \{(i, \sigma^{t_i}(k))\}_{i=1,\dots,N\ ,\ k=0,1,\dots,m}$, where the first firing starts at time τ_0, such that*

- *The event projection $\pi_1(s) = i_0, i_1, i_2, \dots, i_m$ consists of the transitions which form all the fundamental cycles.*
- *The timed projection $\pi_2(s) = \sigma^{t_{i_0}}(k_0), \sigma^{t_{i_1}}(k_1), \dots, \sigma^{t_{i_m}}(k_m)$ satisfies the following conditions $\sum_k \sigma^{t_i}(k) = \tilde{\tau}_i$ and $0 < \Delta \leq \sigma^t(n) < \infty$ for all transitions and firings; $\tilde{\tau} = [\tilde{\tau}_1, \dots, \tilde{\tau}_N]^T$ is the solution of the following linear programming problem*

$$min\ a^T \tilde{\tau}$$

$$subject\ to: \left\{ \begin{array}{l} x_f = x_0 + \sum_{i=1}^N \tilde{\tau}_i c_i \in R \\ \tilde{\tau}_i \geq \Delta,\ i = 1, \dots, N \end{array} \right.$$

where $\tilde{\tau}$ 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 the switching sequence $s(\tau_0, x_0)$ is completed.

Then, the continuous state $x \in \Re^n$ is driven to the target region in finite time $\tau = \sum_{i=1}^N \tilde{\tau}_i$.

Proof By integrating the state equation (54) all the reachable states x_f from initial state x_0 are given by

$$x_f = x_0 + \sum_{i=1}^{N} \tilde{\tau}_i c_i \tag{55}$$

where $\tilde{\tau}_i$ represents the total time the corresponding control policy c_i is active. Although there is no unique switching policy to satisfy the convex constrain $x_f \in R$, the solution of the linear programming problem is unique and gives the control law that drives the state to the target region in minimum time.

Additionally, the necessary number of switches can be minimized by considering one firing of each transition. Additional safety conditions expressed as convex constraints can be incorporated in the linear programming problem. The solution of the above linear programming problem determines the input and output guard functions. The supervisor control algorithms is described by labeling the places, transitions, input and output arcs of the controlled Petri net as follows

- $\ell_P(p)$ is chosen to be a tautology.
- $\ell_T(t_i)$ is chosen to be the atomic rate formulas $\dot{x} = c_i$.
- $\ell_I((p, t_i))$ and $\ell_O((t_i, p))$ are chosen to be time constraints implement by a local timer so that $\sum_k \sigma^{t_i}(k) = \tilde{\tau}_i$.

Example In this example, we consider a hybrid system described by the Petri net in figure 8. We assume now that the continuous part of the hybrid plant consists of a set of first order integrators

$$\dot{x} = c_i \in \Re^2, \quad i = 1, 2, 3, 4. \tag{56}$$

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

and we associate the differential equation $\dot{x} = c_i$ with the transition t_i of the Petri net in figure 8. The control objective is to drive the state from the initial condition $x_0 = [1, -1]^T$ to the convex region R of the state space where

$$R = \left\{ x \in \Re^2 : \begin{bmatrix} 1 \\ 1 \end{bmatrix} \leq x \leq \begin{bmatrix} 1.1 \\ 1.1 \end{bmatrix} \right\} \tag{57}$$

According to Proposition 5 we formulate the following linear programming problem

$$min\ (\tilde{\tau}_1 + \tilde{\tau}_2 + \tilde{\tau}_3 + \tilde{\tau}_4)$$

$$subject\ to : \begin{cases} x_f = x_0 + \tilde{\tau}_1 c_1 + \tilde{\tau}_2 c_2 + \tilde{\tau}_3 c_3 + \tilde{\tau}_4 c_4 \in R \\ \tilde{\tau}_i \geq \Delta \end{cases}$$

where $\Delta = 0.1$. The solution of the linear programming problem gives

$$\tilde{\tau}_1 = 0.6585, \quad \tilde{\tau}_2 = 0.7554; \quad \tilde{\tau}_3 = 0.6261, \quad \tilde{\tau}_4 = 0.1$$

and we can drive the state from x_0 to R with one firing of each transition by setting $\sigma^{t_i} = \tilde{\tau}_i,\ i = 1, 2, 3, 4$. The trajectory of the continuous state is shown in figure 11.

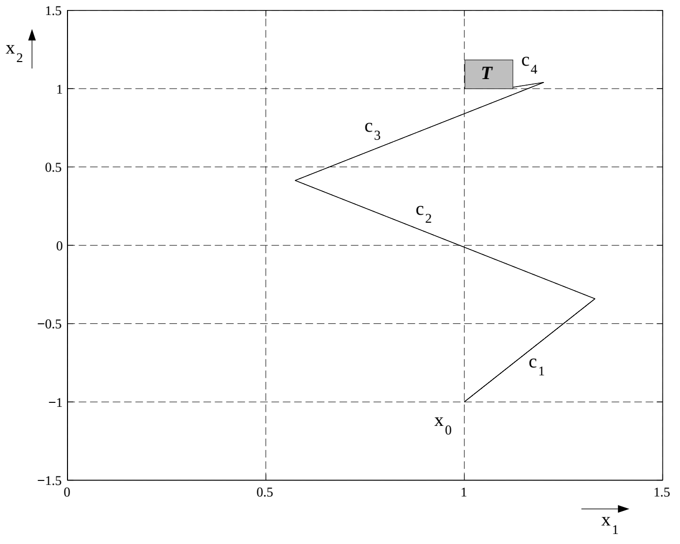

Figure 11: The trajectory of the continuous state.

6 Conclusions

In this paper, uniform ultimate boundness and supervisory control of hybrid systems were addressed using a class of timed Petri nets named programmable timed Petri nets. New methodologies were introduced and efficient algorithms were derived to address these issues. Using Petri nets to model hybrid systems offers advantages over finite automata when concurrency and complexity issues are of concern. Sufficient conditions for uniform ultimate boundness and supervisory control were presented. For the case when the plant is a collection of linear systems with switching logic generated by a programmable timed Petri net, efficient algorithms for stability testing and supervisory control synthesis were developed.

References

Alur, R. and D. Dill (1994*a*). The theory of timed automata. *Theoretical and Computer Science* **126**, 183–235.

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

Alur, R., Henzinger, T.A. and Sontag, E.D., Eds.) (1996*a*). *Hybrid Systems III, Verification and Control*. Vol. 1066 of *Lecture Notes in Computer Science*. Springer.

Alur, R., T. Henzinger and P-H Ho (1996*b*). Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering* **22**, 181–201.

Alur, Rajeev and David Dill (1994*b*). The theory of timed automata. *Theoretical and Computer Science* **126**, 183–235.

Antsaklis, P., Kohn, W., Nerode, A. and Shastry, S., Eds.) (1995). *Hybrid Systems II*. Vol. 999 of *Lecture Notes in Computer Science*. Springer.

Antsaklis, P., Kohn, W., Nerode, A. and Shastry, S., Eds.) (1997). *Hybrid Systems IV*. Vol. 1273 of *Lecture Notes in Computer Science*. Springer.

Antsaklis, P., Kohn, W., Nerode, A. and Shastry, S., Eds.) (1998). *Hybrid Systems V*. Lecture Notes in Computer Science. Springer. In progress.

Antsaklis, P.J. and Nerode, A., Eds.) (1998). *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*.

Baccelli, F., G. Cohen and B. Gaujal (1992). Recursive equations and basic properties of timed Petri nets. *DEDS: Theory and Applications* **1**(4), 415–439.

Bett, C.J. and M.D. Lemmon (1997). Bounded amplitude control using multiple linear agents. Technical Report ISIS-97-004. ISIS Group at the University of Notre Dame. Submitted to Automatica.

Branicky, Michael (1994). Stability of switched and hybrid systems. In: *Proceedings of the 33th Conference on Decision and Control*. Lake Buena Vista, Florida. pp. 3498–3503.

Demongodin, I. and N.T. Koussoulas (1998). Differential Petri nets: Representing continuous systems in a discrete-event world. *Transactions on Automatic Control, Special Issue on Hybrid Systems*.

Engelfriet, J. (1991). Branching processes of Petri nets. *Acta Informatica* **28**, 575–591.

Esparza, J., S. Romer and W. Vogler (1996). An improvement of McMillan's unfolding algorithm. In: *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems* (T. Margaria and B. Steffen, Eds.). Vol. 1055 of *Lecture Notes in Computer Science*. Springer-Verlag.

Flaus, J-M. and H. Alla (1997). Structural analysis of hybrid systems modelled by hybrid flow nets. In: *Proceedings of the European Control Conference 97*. Brussels, Belgium.

Friedland, Bernard (1996). *Advanced Control System Design*. Prentice-Hall.

Giua, A. and E. Usai (1996). High-level hybrid Petri nets: A definition. In: *Proceeding of the 35th Conference on Decision and Control*. Kobe, Japan.

Grossman, R.L., Nerode, A., Ravn, A.P. and Rischel, H., Eds.) (1993). *Hybrid Systems*. Vol. 736 of *Lecture Notes in Computer Science*. Springer-Verlag.

Guckenheimer, John (1995). A robust hybrid stabilization strategy for equilibria. *IEEE Transactions on Automatic Control* **40**(2), 321–326.

He, K.X. and M.D. Lemmon (1997). Lyapunov stability of ontinuous valued systems under the supervision of discrete event transition systems. Technical Report ISIS-97-010. ISIS Group at the University of Notre Dame. Submitted to Hybrid Systems: Control and Computation, Workshop at Berkeley, April 1998.

Henzinger, T.A., P-H. Ho and H. Wong-Toi (1995). A user guide to HyTech. In: *First Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS95*. Vol. 1019 of *Lecture Notes in Computer Science*. Springer-Verlag. pp. 41–71.

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

Hou, L., A.N.Michel and H.Ye (1996). Stability analysis of switched systems. In: *Proceedings of the 35th Conference on Decision and Control*. Kobe, Japan.

Isidori, A. (1996). *Nonlinear Control Systems*. 2nd ed.. Springer-Verlag.

Johansson, M. and A. Rantzer (1998). Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems.*

Koutsoukos, X.D. and P.J. Antsaklis (1997). An approach to hybrid systems control applied to clocks. In: *Fifth International Hybrid Systems Workshop*. Notre Dame, USA.

Le Bail, J., H. Alla and R. David (1991). Hybrid Petri nets. In: *Proceedings 1st European Control Conference*. Grenoble, France.

Lemmon, M., K. He and C.J. Bett (1998). Modeling hybrid control systems using programmable Petri nets. In: *3rd International Conference ADMP'98, Automation of Mixed Processes: Dynamic Hybrid Systems*. Reims, France.

Lemmon, M.D. and C.J. Bett (1996). Safe implementations of supervisory commands. Technical Report ISIS-96-006. ISIS Group at the University of Notre Dame. To appear in International Journal of Control.

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

Lunze, J., B. Nixdorf and H. Richter (1997). Hybrid modelling of continuous-variable systems with application to supervisory control. In: *Proceedings of the European Control Conference 97*. Brussels, Belgium.

McClamroch, N.H., C. Rui, I. Kolmanovsky and M. Reyhanoglu (1997). Hybrid closed loop systems: A nonlinear control perspective. In: *the 36th IEEE Conference on Decision and Control.*

McMillan, K. (1992). Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: *Computer Aided Verification, Fourth International Workshop, CAV'92* (B.V. Bochmann and D.K. Probst, Eds.). Vol. 663 of *Lecture Notes in Computer Science*. Springer-Verlag. pp. 164–177.

Moody, J.O. (1997). Petri Net Supervisors for Discrete Event Systems. PhD thesis. Department of Electrical Engineering, University of Notre Dame. Notre Dame, IN.

Moody, J.O. and P.J. Antsaklis (1996). Supervisory control of Petri nets with uncontrollable/unobservable transitions. In: *Proceedings of the 35th Conference on Decision and Control.* Kobe, Japan. pp. 4433–4438.

Moody, J.O. and P.J. Antsaklis (1997). Supervisory control using computationally efficient linear techniques: A tutorial introduction. In: *Proceedings of the 5th International IEEE Mediterranean Conference on Control and Systems.* Paphos, Cyprus.

Morse, A.Stephen, Ed.) (1997). *Control using logic-based switching.* Vol. 222 of *Lecture Notes in Control and Information Sciences.* Springer.

Peleties, P. and R. DeCarlo (1991). Asymptotic stability of m-switched systems using Lyapunov-like functions. In: *Proceedings of the American Control Conference.* pp. 1679–1684.

Peleties, P. and R. DeCarlo (1994). Analysis of hybrid systems using symbolic dynamics and Petri nets. *Automatica* **30**(9), 1421–1427.

Pettersson, S. and B. Lennartson (1995). Hybrid modelling focused on hybrid Petri nets. In: *Proceedings 2nd European Workshop on Real-time and Hybrid systems.* Grenoble, France. pp. 303–309.

Pettersson, S. and B. Lennartson (1996). Stability and robustness of hybrid systems. In: *Proceedings of the 35th Conference on Decision and Control.* Kobe, Japan.

Puri, Anuj and Pravin Varaiya (1994). Decidability of hybrid systems with rectangular differential inclusions. In: *CAV'94: Computer Aided Verification* (D.L. Dill, Ed.). Vol. 818 of *Lecture Notes in Computer Science.* Springer-Verlag. pp. 81–94.

Ramadge, P.J. (1990). On the periodicity of symbolic observations of piecewise smooth discrete-time systems. *IEEE Transactions on Automatic Control* **35**(7), 807–813.

Reisig, W. (1985). *Petri Nets.* Springer-Verlag.

Sifakis, J. (1977). Use of petri nets for performance evaluation. In: *Measuring, modelling and evaluating computer systems.* North Holand.

Sreenivas, R.S. (1997). On the existence of supervisor policies that enforce liveness in discrete-event dynamic systems modeled by controlled petri nets. *IEEE Transactions on Automatic Control* **42**(7), 928–945.

Stiver, J.A., P.J. Antsaklis and M.D. Lemmon (1995). Interface and controller design for hybrid control systems. In: *Hybrid Systems II* (Panos Antsaklis, Wolf Kohn, Anil Nerode and Shankar Sastry, Eds.). Vol. 999 of *Lecture Notes in Computer Science.* Springer. pp. 462–492.

Stiver, J.A., P.J. Antsaklis and M.D. Lemmon (1996*a*). An invariant based approach to the design of hybrid control systems. In: *IFAC 13th Triennial World Congress.* Vol. J. San Francisco, CA. pp. 467–472.

Stiver, J.A., P.J. Antsaklis and M.D. Lemmon (1996*b*). An invariant based approach to the design of hybrid control systems containing clocks. In: *Hybrid Systems III, Verification and Control* (Rajeev Alur, Thomas A. Henzinger and Eduardo D. Sontag, Eds.). Vol. 1066 of *Lecture Notes in Computer Science*. Springer. pp. 464–474.

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

Tittus, Michael (1995). Control Synthesis for Batch Processes. PhD thesis. Control Engineering Lab., Chalmers University of Technology. Göteborg, Sweden.

Vibert, D., C. Valentin-Roubinet and E. Neil (1997). A modelling method to take into account fluctuations of continuous variables in a class of hybrid systems. In: *Proceedings of the European Control Conference 97*. Brussels, Belgium.

Yamalidou, K. and J.C. Kantor (1991). Modeling and optimal control of discrete-event chemical processes using Petri nets. *Computers in Chemical Engineering* **15**(7), 503–519.

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