Discrete Event Dynamic Systems: Theory and Applications, 7, 151–190 (1997)

A Survey of Petri Net Methods for Controlled Discrete Event Systems

L. E. HOLLOWAY holloway@engr.uky.edu
Center for Robotics and Manufacturing Systems and Department of Electrical Engineering, University of Kentucky, Lexington, KY 40506-0108, USA

B. H. KROGH krogh@ece.cmu.edu
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

A. GIUA giua@elettro1.unica.it
Dipartimento di Ingegneria Elettrica ed Elettronica, Università di Cagliari, Piazza d'Armi, 09123 Cagliari, Italy

Received January 27, 1995; Revised August 6, 1996; Accepted August 14, 1996

Abstract. This paper surveys recent research on the application of Petri net models to the analysis and synthesis of controllers for discrete event systems. Petri nets have been used extensively in applications such as automated manufacturing, and there exists a large body of tools for qualitative and quantitative analysis of Petri nets. The goal of Petri net research in discrete event systems is to exploit the structural properties of Petri net models in computationally efficient algorithms for computing controls. We present an overview of the various models and problems formulated in the literature focusing on two particular models, the controlled Petri nets and the labeled nets. We describe two basic approaches for controller synthesis, based on state feedback and event feedback. We also discuss two efficient techniques for the on-line computation of the control law, namely the linear integer programming approach which takes advantage of the linear structure of the Petri net state transition equation, and path-based algorithms which take advantage of the graphical structure of Petri net models. Extensions to timed models are briefly described. The paper concludes with a discussion of directions for future research.

Keywords: Petri nets, supervisory control, untimed models, logical control

1. Introduction

Models of discrete event systems (DESs) may be grouped into two main classes. *Untimed models* are those models in which order of states or events is relevant in the control specification and design. The specific time instants when state transitions and events occur are not considered. *Timed models* are intended for the study of properties explicitly dependent on inter-event timing. Petri nets are effective for modeling both untimed and timed DESs, particularly when there is a high degree of concurrency and synchronization. The purpose of this survey is to provide an overview of recent research on the application of Petri net models and methods to problems in the logical control of DESs focusing on untimed models. The theory and applications of timed Petri net models for simulation, performance evaluation, and system optimization are outside the scope of this survey (see Baccelli et al. (1992), Cohen et al. (1989) and references therein for recent research on Petri net models for timed DESs). For general background on the theory and applications of Petri nets, the

reader is referred to the survey papers Murata (1977), Zurawski and Zhou (1994) and the standard texts Peterson (1981), Reisig (1982).

Three main design approaches for the control of logical DES using Petri net models are discussed in the literature.

Controlled Behavior approach

In this approach, which is the most common when using Petri net models for manufacturing systems, the model describes the behavior of the closed loop system, i.e., the behavior of the plant and controller joined together. When the desired controlled behavior is obtained, it is necessary to extract the controller logic for implementation. This approach has some advantages when a declarative model, rather than procedural model, is used. Bottom-up or top-down design rules may be used to ensure that the final model enjoys properties of interests (liveness, boundedness, etc.). Examples of this approach are found in Zhou, DiCesare, Desrochers (1992), Jeng and DiCesare (1993), Zhou and DiCesare (1993), Suziki and Murara (1983).

Logic controller approach

The second approach focuses on the direct design and implementation of a controller for the plant. The objective is to define the input-output behavior for the controller to achieve the desired controlled behavior for the closed-loop system. Generally the controller receives commands from an external agent which it must translate into a sequence of operations to be performed by the plant. This approach leads naturally to the physical implementation of the control program, but simulation is required to validate the closed-loop behavior. Examples of this approach in which Petri nets are used to define the control logic include Bruno and Marchetto (1986), Valette (1983), Zhou, DiCesare and Rudolph (1992). David and Alla (1992, 1993) discuss the relationships between Petri nets and the programming language GRAFCET for specification of controller logic.

Control theoretic approach

This approach adopts the controller synthesis paradigm from control theory for continuous systems. Given a model of the plant dynamics and a specification for the desired closed-loop behavior, the objective is to synthesize a controller to achieve the specifications. In this approach there is a clear distinction between the plant and the controller and the information flow between the plant and controller is modeled explicitly. Different restrictions on the information flow give rise to problems of controllability, observability, decentralized control, etc. Examples of this approach to DESs are the classical Ramadge and Wonham (1989) approach that will be discussed in this paper, that of Lewis et al. (1993) based on the definition of task matrices, and that of Stiver and Antsaklis (1993) which extends the

representation power of the Ramadge and Wonham approach to hybrid systems. This paper focuses on the use of Petri nets in the control theoretic approach.

The seminal research by Ramadge and Wonham on the existence and synthesis of controllers for DESs used controlled automata to model the plant (Ramadge and Wonham 1987a, 1987b). Controlled automata provide a general framework for establishing fundamental properties of DES control problems. They are not convenient or intuitive models for practical systems, however, because of the large number of states that have to be introduced to represent several interacting subsystems. Moreover, the lack of structure in controlled automata models limits the possibilities for developing computationally efficient algorithms for analysis and synthesis.

Petri nets have been proposed as an alternative modeling formalism for DES control to exploit purported advantages Petri nets offer over automata models. Petri net models are generally more compact and more powerful than automata models. Petri nets are already used in application areas such as automated manufacturing, and there exists a large body of tools for Petri net analysis and design. For control, Petri nets offer a structured model of DES dynamics that can be exploited in developing more efficient algorithms for controller synthesis.

In this paper we survey research on controller synthesis for plants modeled by Petri nets focusing on two main approaches. The *state feedback control* has been mainly studied by means of a particular model called *controlled Petri nets* (CtlPNs). The *event feedback control* has been mainly considered in a formal language setting and the corresponding models are called *labeled Petri nets*. The following section presents the basic untimed Petri net model, controlled Petri nets, and labeled nets. Section 3 describes state feedback controllers for CtlPNs and the general conditions that must be satisfied for a state feedback control policy to exist to prevent the CtlPN from reaching a given set of forbidden markings. More general results on modular synthesis and restricted observations of the marking are also described. In section 4, control of the event feedback behavior of labeled nets is considered. We summarize a general design technique based on net operators, and discuss some problems related to the existence and computability of controllers. We then present two general approaches for the on-line computation of state feedback policies, namely, techniques which rely on the linear-algebraic formulation of the net model (section 5), and techniques which rely on the graphical structure of Petri nets (section 6). Section 7 presents an overview of supervisory control to avoid deadlocks and ensure liveness. Recent extensions of logical controller synthesis methods for CtlPNs to timed Petri net models are described in section 8. The paper concludes with a discussion of several directions for further research in section 9.

2. Preliminaries

In section 2.1 we introduce basic definitions and notation for ordinary Petri nets. This basic Petri net model has often been enhanced and modified to serve various purposes (see, e.g. Jensen (1995)). In sections 2.2 and 2.3 we review two models that have been used for DES control, namely, *controlled Petri nets* and *labeled Petri nets*.

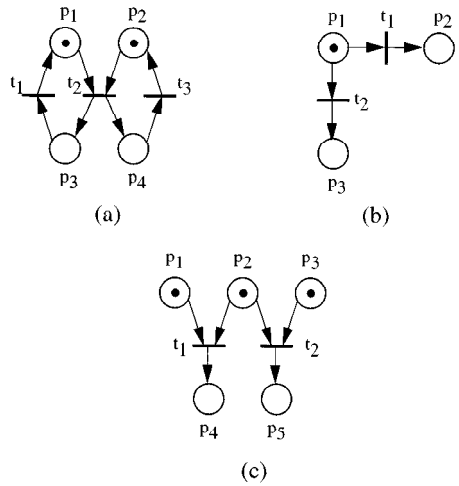

Figure 1. Examples of place/transition nets.

2.1. Ordinary Petri Nets

An *ordinary Petri net structure* is a triple, $N = (P, T, \mathcal{E})$, where:

- P is a finite set of *places*
- T is a finite set of *transitions*;
- $\mathcal{E} \subseteq (P \times T) \cup (T \times P)$ is the *incidence relation*, representing the set of directed arcs connecting places to transitions and vice versa.

It is assumed that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$. Graphically, places are represented by circles and transitions are represented by bars, as illustrated in the nets in Fig. 1.

A net is said to be *pure* if it has no self-loops, i.e., if for $p \in P$, $t \in T$, $[(p, t) \in \mathcal{E} \Rightarrow (t, p) \notin \mathcal{E}]$. If a net is pure the incidence relation can be represented by a single matrix $E\colon P \times T \to \{0, 1, -1\}$, called the *incidence matrix* of the net, defined as

$$E(p, t) = \begin{cases} 1 & \text{if } (t, p) \in \mathcal{E} \\ -1 & \text{if } (p, t) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}.$$

The *preset* and *postset* of a transition t are defined respectively as ${}^{(p)}t = \{p \mid (p, t) \in \mathcal{E}\}$, and $t^{(p)} = \{p \mid (t, p) \in \mathcal{E}\}$. The *preset* and *postset* of a place p are respectively ${}^{(t)}p = \{t \mid (t, p) \in \mathcal{E}\}$, and $p^{(t)} = \{t \mid (p, t) \in \mathcal{E}\}$.

A *marking* is a vector $m\colon P \to \mathbb{N}$ that assigns to each place of a Petri net a non-negative integer number of tokens, represented by black dots as in Fig. 1, where $m(p)$ denotes the number of tokens assigned by marking m to place p. The set of all markings defined on a

net $N = (P, T, \mathcal{E})$ is $\mathcal{M} = \mathbb{N}^{|P|}$. A *net system* $\langle N, m_0 \rangle$ is a net N with an initial marking m_0.

A set of transitions $\tau \subseteq T$ is *enabled* by a marking m if

$$\forall p \in P, m(p) \geq |p^{(t)} \cap \tau|, \tag{1}$$

that is, for each place $p \in P$, $m(p)$ is greater than the number of transitions in τ for which p is an input place.

There are two different assumptions commonly made regarding the number of transitions that can fire at a given instant. Under the *concurrency assumption* more than one transition can fire at any instant. Thus, if a set of transitions $\tau \subseteq T$ is enabled at marking m, then τ may fire yielding a new marking

$$m'(p) = m(p) + |{}^{(t)}p \cap \tau| - |p^{(t)} \cap \tau|. \tag{2}$$

In words, firing an enabled set of transitions $\tau \subseteq T$ causes one token to be removed from each place $p \in {}^{(p)}t$, and one token to be added to each $p \in t^{(p)}$, for each $t \in \tau$. We write $m\ [\tau\rangle$ to denote that τ may fire at m, and $m\ [\tau\rangle\ m'$ to denote that τ may fire, resulting in m'.

In much of the Petri net literature it is assumed that only a single transition can fire at any instant. We refer to this case as the *no concurrency (NC) assumption*. Under the NC assumption, the firing equation (2) holds with τ a singleton set.

A *firing sequence* from a marking m_0 is a (possibly empty) sequence of transition sets $\sigma = \tau_1 \ldots \tau_k$ such that $m_0\ [\tau_1\rangle\ m_1\ [\tau_2\rangle\ m_2 \cdots\ [\tau_k\rangle\ m_k$. We also write $m_0\ [\sigma\rangle$ to denote that we may fire the sequence σ at m_0, and $m_0\ [\sigma\rangle\ m_k$ to denote that the firing of σ yields m_k. Under the NC assumption, each τ_i is a singleton set, and σ is a sequence of transitions.

A marking m is *reachable* in $\langle N, m_0 \rangle$ if there exists a firing sequence σ such that $m_0\ [\sigma\rangle\ m$. Given a net system $\langle N, m_0 \rangle$, the set of reachable markings (also called the *reachability set* of the net) is denoted $R(N, m_0)$. A transition $t \in T$ is *live* if for any marking $m \in R(N, m_0)$ there always exists a marking $m' \in R(N, m)$ such that t is enabled by m'; a net system is live if all of the transitions are live. A transition $t \in T$ is said to be in *deadlock* at a marking $m \in R(N, m_0)$ if it cannot be enabled by any marking in $R(N, m)$. A net system is in deadlock at a marking if all of the transitions are in deadlock. In applications, deadlock in a Petri net model often represents the classical circular wait condition: each activity in a set of activities is holding a resource needed by one of the other activities the set, so none of the activities can proceed (Coffman et al. 1971). A related notion is *livelock*, which describes situations where it is possible to fire some set of transitions indefinitely without enabling some other set of transitions (Sifikas 1980). In section 7 we consider control policies for deadlock avoidance and liveness. The issue of livelock has not been addressed so far in the literature on Petri net methods for controlled DESs.

A *marked graph* or *event graph* is a Petri net such that each place has exactly one input arc and one output arc, i.e., $|{}^{(t)}p| = |p^{(t)}| = 1$. Marked graph structures can model synchronization of concurrent processes: tokens in places which share an output transition must progress synchronously. However, marked graph structures cannot represent choice in the plant model: there is only a single event that can remove a token from a given place,

namely, the firing of the unique output transition for the place. Thus, enabled transitions are *persistent* for marked graphs, which means that once a transition is enabled, it remains enabled only if it fires.

A *state graph* is a Petri net such that each transition has exactly one input arc and one output arc, i.e. $|{}^{(p)}t| = |t^{(p)}| = 1$. A state graph structure with a single token is analogous to a finite-state automaton: each place corresponds to a state of an automaton and the token location indicates the current state. Since places can have multiple output transitions in a state graph structure, these structures can represent choice in the plant dynamics. Transitions are not necessarily persistent for state graphs. State graphs with multiple tokens may represent a restricted kind of concurrency, since more than one transition may be enabled at a given marking. A state graph is unable to represent synchronization of concurrent processes, however, since each transition is enabled by the marking of at most one place. Section 6 examines a family of control synthesis methods which exploit state graph and marked graph structures to compute controls.

Figure 1 shows three nets: (a) a marked graph; (b) a state graph; and (c) an ordinary Petri net which is neither a marked graph nor a state machine.

It is sometimes useful to write the firing equation (2) of a net as a linear matrix-vector equation. Let marking m be reachable from marking m_0 by firing a sequence $\sigma = \tau_1 \ldots \tau_k$. Then the following *state transition equation* is satisfied:

$$m = m_0 + E \cdot \overline{\sigma}, \tag{3}$$

where $\overline{\sigma}$: $T \rightarrow \mathbb{N}$ is a vector of non-negative integers, called the *firing count vector* defined as:

$$\overline{\sigma}(t) := \sum_{i=1}^{k} |\{t\} \bigcap \tau_i| \,.$$

That is, $\overline{\sigma}(t)$ represents the number of times transition t appears in σ. The set of markings such that there exists a vector $\overline{\sigma}$ satisfying the state transition equation (3) is called the *potentially reachable set* and is denoted $PR(N, m_0)$. Note that in general $PR(N, m_0) \supseteq R(N, m_0)$. However, for *acyclic* (also called *loop free*) nets, i.e., nets where no directed path forms a cycle, $PR(N, m_0) = R(N, m_0)$ (Ichikawa and Hiraishi 1988, Lemma 4).

The state equation (3) in matrix-vector form resembles the standard state transition equation for discrete-time linear systems, with the marking vector as the state vector, and the firing vector as input vector. Letting m_{k+1} denote the marking after the kth transition firing and letting $\overline{\sigma}_k$ denote the kth firing vector, (3) becomes

$$m_{k+1} = m_k + E \cdot \overline{\sigma}_k, \tag{4}$$

which is strongly reminiscent of $x_{k+1} = A \cdot x_k + B \cdot u_k$ from linear systems theory. This suggests that existing results from linear system theory can be readily applied to the special case of Petri net dynamics. This analogy was been explored by some authors (see, for example, Murata (1989)), but there is a complication in the Petri net dynamics which limits its usefulness: only nonnegative markings are allowed. Thus, when viewed as linear systems, Petri nets have a state constraint which imposes a state-dependent constraint on

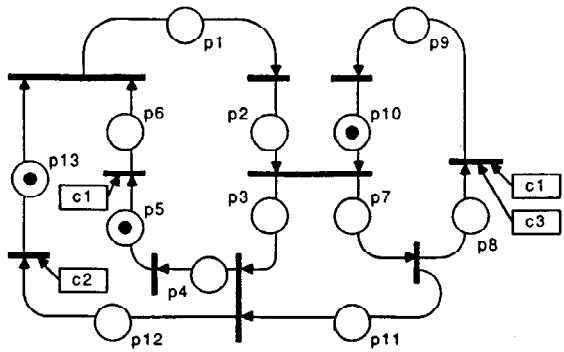

Figure 2. A controlled Petri net.

the set of admissible inputs (i.e., enabled transitions) at any instant. Nevertheless, there is some value in viewing the Petri net dynamics from a linear algebraic perspective and this approach, know as *structural analysis*, has been developed extensively in the literature (Memmi and Roucairol 1980, Best 1987, Johnson and Murara 1985, Sifakis 1978, Colom and Silva 1991). The linear algebraic approach to DES control is described in section 5.

2.2. Controlled Petri Nets

Controlled Petri nets (CtlPNs) are a class of Petri nets with external enabling conditions called *control places* which allow an external controller to influence the progression of tokens in the net. CtlPNs were first introduced by Krogh (1987) and Ichikawa and Hiraishi (1988).

Formally, a CtlPN is a triple, $N^c = (N, C, \mathcal{B})$ where is $N = (P, T, \mathcal{E})$ is an ordinary Petri net structure and

– C is a finite set of *control places*, disjoint from P, T,

– $\mathcal{B} \subseteq (C \times T)$ is a set of directed arcs connecting control places to transitions.

In the CtlPN context, the elements of P are referred to as *state places*.

For a transition $t \in T$ we denote the set of input control places as ${}^{(c)}t := \{c \mid (c, t) \in \mathcal{B}\}$, and for a control place $c \in C$ we denote the set of output transitions as $c^{(t)} := \{t \mid (c, t) \in \mathcal{B}\}$. A transition t is said to be a *controlled transition* if its set of control inputs ${}^{(c)}t$ is nonempty. The set of all controlled transitions is denoted by T_c. Figure 2 illustrates a controlled Petri net, where circles represent state places, bars represent transitions, and squares indicate control places.

As with ordinary Petri nets, the state of a CtlPN is given by its *marking*, which is the distribution of *tokens* in the state places. A set of transitions $\tau \subseteq T$ is *state enabled* under a marking if equation (1) is satisfied.

A *control* for a CtlPN is a function $u\colon C \longrightarrow \{0,1\}$ associating a binary value to each control place. The set of all such controls is denoted by $\mathcal{U}$. A set of transitions $\tau \subseteq T$ is said to be *control enabled* if for all $t \in \tau$, $u(c) = 1$ for all $c \in {}^{(c)}t$. A control $u \in \mathcal{U}$ is said to be *as permissive as* control $u' \in \mathcal{U}$ if $u(c) \geq u'(c)$ for all $c \in C$. Control u is said to be *more permissive than* control u' if u is as permissive as u' and $u(c) > u'(c)$ for some $c \in C$. The most permissive control is $u_{one} := 1$, and the least permissive control is $u_{zero} := 0$.

CtlPNs are generally used under the concurrency assumption. Letting $\mathcal{T}_e(m,u) \subseteq 2^T$ denote the collection of sets of transitions that are both state enabled by $m \in \mathcal{M}$ and control enabled by $u \in \mathcal{U}$, any set of transitions $\tau \in \mathcal{T}_e(m,u)$ can *fire*, thereby changing the marking of the net to the marking m' as defined by equation (2). *Note that firing a set of transitions in a CtlPN has no effect on the control.*

Given a marking $m \in \mathcal{M}$ and control $u \in \mathcal{U}$, the set of *immediately reachable markings*, $R_1(m,u)$, is given by

$$R_1(m,u) = \{m\} \bigcup \{m' \in \mathcal{M} \mid m' \text{ is given by (2) for some } \tau \in \mathcal{T}_e(m,u)\}. \tag{5}$$

The set of *reachable markings* under an arbitrary number of transition firings from a given marking $m \in \mathcal{M}$ with a constant control $u \in \mathcal{U}$, denoted $R_\infty(m,u)$, is defined by

1. $m \in R_\infty(m,u)$;
2. if $m' \in R_\infty(m,u)$, then $R_1(m',u) \subseteq R_\infty(m,u)$; and
3. all $m' \in R_\infty(m,u)$ are defined by 1 and 2.

We conclude this section by noting that the controlled Petri net model defined by Ichikawa and Hiraishi differs from the CtlPNs defined above. In the model proposed by Ichikawa and Hiraishi (1988), the tokens in the control places (the *external input places* in their model) are consumed by the firing of controlled transitions. Thus, their control places are similar to state places, and their control-place markings can be non-binary. They also define the state transition equation (2) for only maximal sets (with respect to set containment) of enabled transitions, and they consider only *decision-free* Petri nets, which are Petri nets for which the maximal set of enabled transitions is unique for any reachable marking. In this context Ichikawa and Hiraishi consider the problems of loading up the control places initially and sequentially to achieve: (i) a given firing sequence; (ii) a given firing count (i.e., a given firing vector); and (iii) a given final marking. In all of these problems, the control policies are open-loop; feedback from the Petri net is not used to define a closed-loop system. In this paper we focus exclusively on the synthesis of *feedback* control policies.

2.3. Labeled Petri Nets

In a labeled Petri net the firing of a transition corresponds to an *event* in the usual DES terminology and the set of all admissible event sequences (as defined by an acceptance criterion) is called the *Petri net language*. In general, Petri net languages are defined in terms of a separate set of event labels which can be assigned to some or all of the transitions.

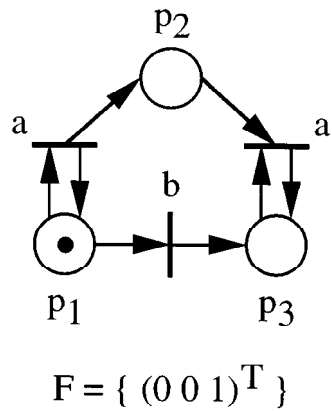

Figure 3. A labeled net.

The notion of the language of a net is useful for specifying and analyzing problems of sequential control which are considered in section 4.

A *labeled Petri net* (or *Petri net generator*) (Jantzen 1987, Peterson 1981), is a 5-tuple $G = (N, \Sigma, \ell, m_0, F)$ where

- $N = (P, T, \mathcal{E})$ is a Petri net structure;
- Σ is a finite set (alphabet) of *events*;
- ℓ: $T \to \Sigma$ is a labeling function that assigns an event to each transition and can be extended to a mapping $T^* \to \Sigma^*$ in the usual way;
- $m_0 \in \mathcal{M}$ is an initial marking;
- $F \subset \mathcal{M}$ is a finite set of final markings.

The labeling function ℓ, as defined above, is a so-called λ-free labeling function (Peterson 1981), i.e., no transition is labeled with the empty string λ and two (or more) transitions may have the same label. Figure 3 shows a labeled Petri net. Each transition is labeled with a symbol from an alphabet $\Sigma = \{a, b\}$, and the set of final markings is $F = \{(001)^T\}$.

The two languages usually associated with G are the *P-type language* (also called the *closed behavior* in the context of supervisory control) and the *L-type language* (also called the *marked behavior*). The closed behavior represents all possible evolutions of the labeled net, while the marked behavior is used to represent the terminal behavior, i.e., all evolutions that reach a terminal state. It is also possible to define a different notion of terminal behavior considering the *G-type language* (also called *weak behavior*). These languages are defined as follows (Peterson 1981).

Given a labeled net $G = (N, \Sigma, \ell, m_0, F)$, the *P-type language* of G is

$$L(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, m_0\ [\sigma\rangle\};$$

the *L-type language* of G is

$$L_m(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, m_0\ [\sigma\rangle\ m, m \in F\};$$

and the *G-type language* of G is

$$L_w(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, m_0\ [\sigma\rangle\ m, m \geq m' \text{ for some } m' \in \mathcal{F}\}.$$

The classes of P-type, L-type, and G-type languages generated by labeled nets are denoted $\mathcal{P}$, $\mathcal{L}$, and $\mathcal{G}$ respectively. These definitions of Petri net languages reflect the NC assumption: only one event can occur at a time, resulting in *sequential languages*. (In section 4 we briefly discuss a class of Petri net languages in which strict concurrency is allowed.) As an illustration of the different languages that can result from a given Petri net structure, the languages associated to the labeled net G in Figure 3 are $L(G) = \{a^m \mid m \geq 0\} \cup \{a^m b a^n \mid m \geq n \geq 0\}$, $L_m(G) = \{a^m b a^m \mid m \geq 0\}$, $L_w(G) = \{a^m b a^n \mid m \geq n \geq 0\}$. Note that none of these languages is a regular language, that is, none of these languages can be accepted by a finite state automaton.

If each transition is identified with a distinct event, the labeled net is a so-called *free-labeled* Petri net in the theory of Petri net languages (Peterson 1981). Other useful subclasses of languages that we consider in this paper, are those generated by *deterministic* Petri net generator (Jantzen 1987), i.e., nets such that the string of events generated from the initial marking uniquely determines the reached marking. Formally, a labeled net $G = (N, \ell, m_0, F)$ is deterministic if $\forall t, t' \in T$, with $t \neq t'$ and $\forall m \in R(N, m_0)$, $(m\ [\sigma'\rangle m') \wedge (m\ [\sigma''\rangle m'') \wedge (\ell(\sigma') = \ell(\sigma'')) \Longrightarrow m' = m''$. We assume the generators are deterministic throughout this paper. The classes of P-type, L-type and G-type Petri net languages generated by deterministic Petri net generators are denoted, respectively, $\mathcal{P}_d$, $\mathcal{L}_d$ and $\mathcal{G}_d$.

For the classes of λ-free Petri net languages the following strict inclusions hold: $\mathcal{P} \subset \mathcal{G} \subset \mathcal{L}$ (Peterson 1981). Hence, a λ-free Petri net language in the class $\mathcal{P}$ or $\mathcal{G}$ can always be represented as the marked language for some labeled Petri net, i.e., a net can be constructed so that the language is identified with the strings leading to final markings in the net as in the definition of an L-type language. In the case of deterministic nets, however, it is possible to prove that although $\mathcal{P}_d \subset \mathcal{G}_d$ classes $\mathcal{L}_d$ and $\mathcal{G}_d$ are incomparable (Giua and DiCesare 1995). Therefore, for deterministic nets one obtains more general results by considering $\mathcal{G}_d \cup \mathcal{L}_d$ (weak behaviors in addition to marked behaviors), rather than simply $\mathcal{L}_d$ (Giua and DiCesare 1995).

The classes of λ-free Petri net languages are supersets of the corresponding classes of deterministic languages. There is a very good reason, however, for restricting attention to deterministic languages: the decidability of the inclusion problem. It is well known that the inclusion problem: "Is $L_1 \subseteq L_2$?" is undecidable for with $L_1, L_2 \in \mathcal{P}$ (Peterson 1981). For deterministic Petri net languages, however, the following lemma holds.

LEMMA 1 (Giua and DiCesare 1995) *The inclusion problem: "Is $L_1 \subseteq L_2$?" is decidable if $L_1 \in \mathcal{L}$ and $L_2 \in \mathcal{L}_d \cup \mathcal{G}_d$.*

This result follows from the fact that if $L_2 \in \mathcal{L}_d \cup \mathcal{G}_d$, then its complement with respect to Σ^*, denoted $\mathbf{C}L_2$, belongs to the class $\mathcal{L}$ (this is not always true if L_2 is not deterministic). Hence checking for language inclusion can be reduced to checking for emptiness of the L-type language $L_1 \cap \mathbf{C}L_2$, which is known to be decidable. This lemma is used extensively in section 4.4 to prove the decidability of properties of interest for discrete event systems modeled with deterministic labeled nets.

3. State Specifications and State Feedback

In this section we consider state specifications, i.e., specifications given as a set of legal markings for the system to be controlled. In this setting the aim of the control is that of restricting the behavior of a system so that only legal markings can be reached. The corresponding control policy is called *state feedback*. We show here how it can be computed for the CtlPN model presented in section 2.2.

A state feedback policy for a CtlPN is a function $U\colon \mathcal{M} \longrightarrow 2^{\mathcal{U}}$. In general, a state feedback policy is *nondeterministic* because it identifies a *set* of possible controls. The policy is *deterministic* if $U(m)$ is a singleton for all markings $m \in \mathcal{M}$. We extend the notation for immediately reachable markings from a marking $m \in \mathcal{M}$ for a feedback policy U by defining $R_1(m, U) = \bigcup_{u \in U(m)} R_1(m, u)$. Similarly, the set of reachable markings from a marking m for a state feedback policy U, denoted $R_\infty(m, U)$, is defined by

1. $m \in R_\infty(m, U)$;
2. if $m' \in R_\infty(m, U)$, then $R_1(m', U) \subseteq R_\infty(m, U)$; and
3. all $m' \in R_\infty(m, U)$ are defined by 1 and 2.

Extending the concept of relative permissiveness of controls to state feedback policies, we say state feedback policy U_1 is as permissive as state feedback policy U_2, denoted by $U_1 \geq U_2$, if for each $m \in M$, $U_1(m) \supseteq U_2(m)$. It follows that $U_1 \geq U_2$ implies $R_\infty(m, U_1) \supseteq R_\infty(m, U_2)$ for any marking $m \in M$.

State feedback policies for CtlPNs have been investigated by a number of researchers (Holloway and Krogh 1990, Li and Wonham 1993, Holloway and Hossain 1992, Banaszak and Krogh 1990, Ushio 1990, Ushio and Matsumoto 1988, Haoxun and Baosheng 1993, Holloway, Guan and Zhang 1996). In most cases the fundamental problem is to design a state feedback policy that guarantees the system remains in a specified set of allowed states, or, equivalently, that the marking of the CtlPN is never in a specified set of *forbidden markings*. Taking the latter viewpoint, for a given CtlPN N^c with initial marking m_0, let $\mathcal{M}_F$ denote the set of forbidden markings. The objective is to find a state feedback policy $U_F\colon \mathcal{M} \longrightarrow 2^{\mathcal{U}}$ for which:

1. $R_\infty(m_0, U_F) \bigcap \mathcal{M}_F = \emptyset$; and
2. for any policy U' such that $U' \geq U_F$, if U' satisfies 1 above, then $U' = U_F$.

We call a state feedback policy satisfying 1 and 2 above a *maximally permissive state feedback policy* for the given forbidden state specification $\mathcal{M}_F$.

A necessary and sufficient condition for the existence of a maximally permissive state feedback policy is determined by an analysis of the CtlPN behavior under the control u_{zero}. Specifically, define the set of *admissible markings* for a CtlPN N^c with respect to a set of forbidden markings $\mathcal{M}_F$ as

$$\mathcal{A}(\mathcal{M}_F) = \{m \in \mathcal{M} \mid R_\infty(m, u_{zero}) \bigcap \mathcal{M}_F = \emptyset\}. \tag{6}$$

Necessary and sufficient conditions for the existence of a state feedback control policy that keeps a CtlPN out of a given set of forbidden markings are then given by the following theorem.

THEOREM 2 (Krogh and Holloway 1991) *Given a CtlPN N^c with initial marking m_0 and a forbidden marking specification $\mathcal{M}_F$, a unique maximally permissive state feedback policy exists if and only if $m_0 \in \mathcal{A}(\mathcal{M}_F)$.*

The *unique* maximally permissive state feedback policy in Th. 2 is *nondeterministic* because the firing rule 2 allows multiple transitions in the CtlPN to fire simultaneously. In general, there will not be a unique *deterministic* maximally permissive policy because the set of controls $U_F(m)$ does not have necessarily a unique maximal element (Holloway and Krogh 1990, Krogh 1987). Takai et al. obtained necessary and sufficient conditions under which a unique maximal element exists for $U_F(m)$, which means the maximally permissive policy can be implemented deterministically even though concurrent transitions can occur (Takai, Ushio and Kodama 1994). On the other hand, under the NC assumption where the firing rule (2) is restricted to singleton sets, a unique *deterministic* maximally permissive state feedback policy exists. This is the case considered by Li and Wonham (1993). Li and Wonham consider the ramifications of relaxing the NC assumption with respect to achieving maximal reachability with a nondeterministic control policy in (Li and Wonham 1995).

When the initial marking for a CtlPN satisfies the condition $m_0 \in \mathcal{A}(\mathcal{M}_F)$ in Th. 2, the maximally permissive state feedback policy can be described simply as the policy which does not allow any state transitions to markings outside $\mathcal{A}(\mathcal{M}_F)$ (Holloway and Krogh 1990). Since the control set is updated at each state transition, only the immediately reachable markings need to be considered in determining the set of admissible controls from a given marking. This reasoning leads to the following theorem which, like Th. 2, is an extension of the Ramadge and Wonham supervisory control for automata to the case of CtlPNs with strict simultaneity.

THEOREM 3 (Holloway and Krogh 1990, Krogh and Holloway 1991) *Given a CtlPN N^c with a forbidden marking specification $\mathcal{M}_F$ and an initial condition $m_0 \in \mathcal{A}(\mathcal{M}_F)$, if $m \in R_\infty(m_0, U_F)$, then*

$$U_F(m) = \{u \in \mathcal{U} \mid R_1(m, u) - \mathcal{A}(\mathcal{M}_F) = \emptyset\}.$$

Li and Wonham present results similar to Theorems 2 and 3 in terms of predicates on the state space (markings) under the NC assumption (Li and Wonham 1993). In their

terminology, the set of admissible markings $\mathcal{A}(\mathcal{M}_F)$ corresponds to the maximal *control invariant* set (predicate) in $\mathcal{M} - \mathcal{M}_F$. They add the notion of *reachability* to define *controllable predicates*.

Having characterized maximally permissive state feedback policies for forbidden state specifications in general, the issue is how to compute the set of admissible controls for a given marking. One approach is to simply create the equivalent controlled automaton for the CtlPN, which is a matter of generating the *reachability graph* for the Petri net structure with the associated control information, and then apply standard algorithms from the Ramadge and Wonham theory to compute the control. Note that the controlled transitions in the CtlPN lead to non-standard control sets for the equivalent controlled automaton (hence the lack of a unique maximally permissive control), so the generalization of the standard theory for controlled automata to the case of arbitrary control sets would have to be applied (Golaszewski and Ramadge 1988a). The standard Ramadge and Wonham controlled automaton is obtained from the CtlPN reachability graph under the NC assumption when each transition in T_c is controlled independently; that is, when there is a distinct, unique control place connected to each controlled transition. This is the case considered in Li and Wonham (1993).

Since the reachability graph can grow exponentially with respect to the size of the CtlPN model (Krogh, Magott, and Holloway 1991, Watson and Derochers 1994), alternative methods are desirable for computing feedback policies. This is the primary motivation for considering CtlPN models as an alternative to unstructured automata. Sections 5 and 6 present two approaches that have been developed to use the structure of the CtlPN model directly, thereby avoiding the generation of the equivalent controlled automata.

Li and Wonham (1993) develop relationships between predicates on the state space for a CtlPN (i.e., sets of allowable markings) and the language of the CtlPN (sequences of transitions) under state feedback control. In particular, they introduce the concept of *balanced controllers* which are state feedback policies that allow all transitions to fire from a given marking which lead to admissible markings. Thus, balanced controllers are maximally permissive controllers, and the language generated under such a control policy is the largest language (with respect to set containment) possible corresponding to the set of reachable markings from the given initial marking.

We conclude this section with a brief summary of various extensions and generalizations of state feedback policies considered in the literature. Limited state observability is the situation where the controller is unable to distinguish between certain markings. An example is the case where a controller is only able to observe the number of tokens in a subset of places, and thus does not have full knowledge of the current net marking (Haoxun and Baosheng 1991, Li and Wonham 1993). Limited state observability is modeled in our framework by considering a function $O\colon \mathcal{M} \longrightarrow \{o_1, o_2, \ldots o_n\}$ mapping the set of all markings onto a set of observability classes $o_1, \ldots, o_n$. The controller is unable to distinguish between markings in the same observability class, so a control law operating under partial state observation must produce the same controls for any two markings which are observationally equivalent; that is, $U(m) = U(m')$ for any two markings m, m' for which $O(m) = O(m')$. Baosheng and Haoxun (1991) consider partial observability in the context of distributed state feedback control systems, and Li and Wonham (1993) consider partial

observability for centralized controllers. In the problem of distributed control of CtlPNs formulated by Haoxun and Baosheng, control capability is distributed among several predefined controllers, and each controller has only limited observations of the net marking. The distributed control synthesis problem for forbidden state avoidance is solved by augmenting the system net with *coordination places* and then applying the approach of Holloway and Krogh (1990) for enforcing the forbidden state avoidance specification. The coordination places act as a type of semaphore which are used by the individual controllers to coordinate the individual control actions.

Li and Wonham (1993) define a notion of observability for predicates and prove a maximally permissive state feedback policy exists if and only if the given predicate (set of admissible states) is both controllable and observable. Li and Wonham also considered so-called *modular state feedback* policies where the overall specification for the admissible states for the closed-loop system is given as the conjunction (intersection) of a collection of *subspecifications*.

Holloway and Hossain (1992) extended state feedback to a class of dynamic specifications for which the objective is to mark the state places in a given sequence. The controller contains an internal representation of the specified place-marking sequences, and has an internal state to indicate where the system is with respect to the sequence specifications. The control law changes according to this internal state, such that over time different markings of the system are avoided while others are permitted. This is implemented by associating different forbidden conditions (see section 6) with each internal state of the controller.

State targeting is another problem that has been addressed using state feedback. McCarragher and Asada (1995) consider the problem of determining control inputs to steer a system such that a desired place in its plant model becomes marked. They address the problem using dynamic programming to evaluate the optimal path in the controlled Petri net to reach the target marking. They apply the technique to a robotic assembly problem.

4. Sequential Specifications and Event Feedback

In the previous section we considered state specifications, i.e., specifications given as a set of legal markings for the system to be controlled. In supervisory control a specification can also be given as a *specification language*, i.e., as a *legal sequential behavior*. The aim of the control is that of restricting the behavior of a system within the limits of the legal behavior. The corresponding control policy is called *event feedback* and the agent that implements it is called a *supervisor*.

To discuss event feedback, we use the labeled Petri net model presented in section 2.3. In the section 4.1, following Ramadge and Wonham (1989), some basic notions of supervisory control are recalled. In section 4.2 we compare Petri net state feedback with event feedback and present a general procedure to design a supervisor. In section 4.3 we discuss when a supervisor can be represented as a Petri net. In section 4.4 we discuss some issues related to the decidability of properties of interest for Petri net models.

4.1. Supervisory control

Let L be a language on alphabet Σ. Its *prefix closure*, denoted $\overline{L}$, is the set of all prefixes of strings in L; i.e., $\overline{L} = \{\omega \in \Sigma^* \mid \exists \tau \in \Sigma^* \ni \omega\tau \in L\}$. A language L is said to be *prefix-closed* if $L = \overline{L}$.

A DES can be viewed as a language generator on alphabet Σ. Here we assume that a DES is a labeled Petri net $G = (N, \Sigma, \ell, m_0, F)$. G is *nonblocking* if any string that belongs to its closed behavior may be completed to a string that belongs to its marked behavior. A deterministic G is nonblocking if $L(G) = \overline{L_m(G)}$. If the DES behavior is modeled by the weak language (G-type language), G is (weakly) nonblocking if $L(G) = \overline{L_w(G)}$ (Giua and DiCesare 1995).

The alphabet of events Σ is partitioned into two disjoint subsets: Σ_c, the set of *controllable events*, and Σ_u, the set of *uncontrollable events*. The controllable events may be disabled by a controlling agent in order to restrict the behavior of the system, while uncontrollable events may never be disabled.

We define a *control input* as a subset $\gamma \subseteq \Sigma$ satisfying $\Sigma_u \subseteq \gamma$ (i.e., all the uncontrollable events are present in the control input). The control input specifies which events are permitted to occur. If $\Gamma \subseteq 2^{\Sigma}$ is the set of all the possible control inputs, under the NC assumption (which is assumed throughout this section) an event feedback policy consists in switching the control input through a sequence of elements $\gamma_1, \gamma_2, \ldots \in \Gamma$, in response to the observed string of previously generated events. Thus, we may say that an event feedback policy is a mapping:

$$f\colon L(G) \longrightarrow \Gamma. \tag{7}$$

Using notation similar to the notation introduced for state feedback policies, the behaviors of the system G under an event feedback policy f are: the *closed behavior* $L(f \mid G)$, that is the set of strings generated under control, and the *controlled behavior* $L_m(f \mid G) = L(f \mid G) \cap L_m(G)$ (or the *weakly controlled behavior* $L_w(f \mid G) = L(f \mid G) \cap L_w(G)$). Note that we are assuming that the supervisor does not mark strings, i.e., the marked strings of the system under control are precisely the marked strings of the uncontrolled system that survive under control.

The basic supervisory control problem is the following: *Given a net G and a specification language $K \subseteq L(G)$, is it possible to find an event feedback policy f such that: (a) $\overline{L(f \mid G)} = \overline{K}$, and (b) f is nonblocking, i.e., $L(f \mid G) = \overline{L_m(f \mid G)}$ (or $L(f \mid G) = \overline{L_w(f \mid G)}$)?*

The solution to this problem is based on the notion of controllable and nonconflicting languages (Wonham and Ramadge 1987). A language $K \subset \Sigma^*$ is said to be *controllable* (with respect to $L(G)$ and Σ_u) if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. In words, a language K is controllable if whenever a prefix of one of its strings is generated, the occurrence of an uncontrollable event (that cannot be prevented) does not lead to a string that is not a prefix of a string of K any more.

Two languages L_1 and L_2 are said to be *nonconflicting* if $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$. This means that any string that is in the closure of each language, can be completed into a string that belongs to both languages.

Ramadge and Wonham (1989) showed that given a nonblocking DEG G and a nonempty closed language $K \subseteq L(G)$ there exists an event feedback policy f such that $L(f \mid G) = K$ if and only if K is controllable. Furthermore f is nonblocking if and only if K and $L_m(G)$ (or $L_w(G)$) are nonconflicting.

If a language $K \subset \Sigma^*$ is not controllable with respect to $L(G)$ one can compute its *supremal controllable sublanguage* (Wonham and Ramadge 1987) denoted by $K^{\uparrow}$, which is the largest subset of strings in K which is controllable with respect to $L(G)$. Thus, given a noncontrollable specification language K there exists an event feedback policy f that ensures that the behavior of the controlled system is $K^{\uparrow} \subset K$, a subset of the specification language. This event policy is minimally restrictive.

Similarly, if K is a prefix-closed language conflicting with $L_m(G)$, one can consider the sublanguage $K' = \overline{K \cap L_m(G)}$, that is the *supremal nonconflicting sublanguage*.

Most research on supervisory control of labeled Petri nets has been developed in the context of the NC assumption. If transitions are allowed to fire simultaneously in a labeled Petri net, the Petri net language must be defined in general over *bags* of events (Peterson 1981). Given a set of events Σ, a bag of events is a collection of events from Σ in which distinct events can occur more than one time. Bags are required to handle the simultaneous occurrence of transitions with the same event label. Languages that admit strict concurrency have been called *trace languages* in Wang (1993) (although the term "trace" does not necessarily imply strict concurrency in the DES literature, cf. Smedinga (1988)). Wang considered supervisory control of Petri nets without the NC assumption where the specification language is defined over bags of events, and control can be enforced by a simple (i.e., nonconcurrent) event feedback policy as given by equation (7) (Wang 1992). Ushio also used bags to represent concurrent transition firing in CtlPNs and derived necessary and sufficient conditions for the existence of supervisors to (i) achieve given firing sequences and (ii) drive the CtlPN to a specified target marking (Ushio 1989).

4.2. Supervisor Design

The supervisory control scheme described in the previous section is based on the notion of event feedback, i.e., the control pattern computed at each step by the supervisor is a function of the string of events generated by the plant.

Consider the Petri net in Figure 4 where $\Sigma_u = \{a\}$. Assume that the specification is given by the legal language: $K = \{\overline{\omega b \omega' c} \mid \omega, \omega' \in \{a\}^*\}$. Clearly K can be enforced by event feedback; one possible choice of f is the following: $f(\omega) = \{a, b\}$ *if* $b \notin \omega$ *else* $f(\omega) = \{a, c\}$. We also note that this specification cannot be enforced by state feedback, since the control applied by the supervisor when the state marking is $m = [0\ 1\ 0]^T$ depends on whether the plant has generated the event b before reaching m.

If the plant $\langle N, m_0 \rangle$ is deterministic, it is possible to reconstruct its state from the string of events generated. That is, there exists a function $g\colon \Sigma^* \to R(N, m_0)$ that associates to each string generated by the plant a marking. Hence if we are given a state feedback policy $f'\colon R(N, m_0) \to \Gamma$ we can always construct an equivalent event feedback $f\colon f' \circ g$. In this sense, we can say that, for deterministic plants, event feedback can also be used to enforce state specifications and is more general than state feedback.

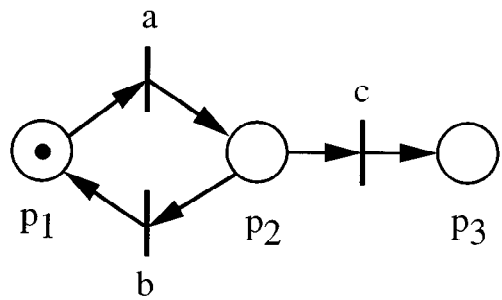

Figure 4. A Petri net to be controlled with event feedback.

On the other hand, sequential specifications for a given model may be converted into state specifications for an *augmented* model in which the dynamics of the plant are augmented with an additional automaton or Petri net that "encodes" the desired sequential behavior of the system in the augmented states. One can then apply state feedback to the augmented system to achieve the desired sequential behavior. This approach has been developed by Li and Wonham (1993), using an automaton called a *memory* to augment the plant dynamics; and by Giua and DiCesare (1991, 1994) and Kumar and Holloway (1996) using a Petri net to augment the plant dynamics. In all cases it is shown that the maximally permissive state feedback control for avoiding an appropriately defined predicate or set of forbidden states on the augmented state space results in the supremal controllable sublanguage $K^{\uparrow}$ for a given sequential specification K.

This technique as developed in Giua and DiCesare (1991) is briefly described here. Consider a DES represented by a Petri net generator G on alphabet Σ. The specifications to be enforced on its behavior is represented by a labeled Petri net H on alphabet $\Sigma' \subseteq \Sigma$ whose closed behavior is $L(H)$. The desired legal behavior is the language $K \subseteq L(G)$ such that the projection of K on Σ' is $K|_{\Sigma'} = L(H)$. The supervisor E is computed by the *concurrent composition*, $E = G \parallel H$. The concurrent composition of two nets can be constructed by fusing the transitions with the same labels on both nets. Thus $L(E) = L(G) \parallel L(H) = K$, i.e., its behavior is given by all the strings of G that are also legal. The controlled behavior of E is $L_m(E) = L_m(G) \parallel L(H) = L_m(G) \cap K$ (and the weak controlled behavior is $L_w(E) = L_w(G) \parallel L(H) = L_w(G) \cap K$).

If the language K is controllable and nonconflicting, the net E will have the following properties: *trimness*, the net E does not admit blocking markings, i.e., reachable markings from which a final marking cannot not be reached; and *controllability*, it is not possible to reach "uncontrollable markings", i.e., markings from which a transition labeled by an uncontrollable event is enabled in G but is not enabled in E. If E enjoys these two properties it is called a *monolithic supervisor*, being at the same time a proper supervisor and a closed-loop model of the system under control. The net E can be used to compute the event feedback policy in this fashion: E runs in parallel with the system G, i.e., whenever an event occurs in G the same event will be executed on E. The events enabled at a given instant on E determine the control input.

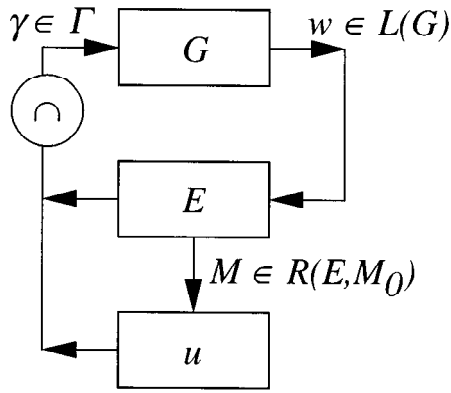

Figure 5. Control scheme for event feedback.

If E is not trim and controllable as defined above, it is necessary to refine this net, further restricting its behavior to avoid reaching all blocking and uncontrollable markings. This operation is the counterpart of computing the supremal controllable and nonconflicting sublanguage. A possible way of implementing this refinement is through an additional state feedback law u that prevents E from reaching the undesirable markings, as illustrated in Figure 5. In this control scheme the control input γ is the intersection of the two control inputs computed by E and u.

An efficient algorithm for the computation of the set of undesirable markings has been derived by Barroso, Lima and Perkusich (1996).

Additionally, it may be possible to compile this state feedback law into a net structure, i.e., to construct a new net E', obtained from E adding additional structure (arcs, places, or transitions) to avoid reaching the undesirable markings. The control input computed by E' is the same as that computed by E and u together.

One advantage of representing the supervisor as a Petri net is that the computation of the control action is faster, since it does not require separate computation of the control. An additional advantage is that a closed-loop model of the system under control can be built and analyzed for properties of interest using Petri net techniques.

In another interesting approach, presented by Makungu, Barbeau, and St-Denis (1994), supervisory design has been applied to systems represented by colored Petri nets. Colored Petri nets with a finite color set do not extend the modeling power of ordinary place/transition nets, but they offer a more compact representation of large systems consisting of many similar interacting components (Jensen 1995). In Makungu, Barbeau and St-Denis (1994) a forbidden state avoidance problem is considered but since the controller action is based on event feedback this approach may possibly be extended to enforce specification languages. Another interesting feature of this approach, is the fact that by partitioning the reachability set into *equivalence classes* it is possible to solve the forbidden state problem by exploring a set of markings significantly smaller than the overall state set.

In addition to these design techniques, there have been other approaches in which for given classes of control problems a Petri net supervisor has been found in "closed form solution" by enforcing linear constraints on the reachable marking set and of finding the corresponding control structure. Giua et al. have solved this problem for plants represented by controlled safe marked graphs (Giua, DiCesare and Silva 1992, 1993), while Li and Wonham (1994) have discussed the case of plants where the uncontrollable subnets have a tree-like structure. These supervisors are usually less complex than the controllers obtained through the construction of the monolithic supervisor described above because they represent only controller states, rather than all the states of the closed-loop system. In this case the control structure is simpler and a closed-loop model of the system under control can still be built with standard Petri net composition operators.

4.3. Computability of Event Feedback Net Supervisors

In the case of systems and specifications modeled by finite state machines, it is well known the supervisor can be modeled as a finite state machine and constructed with a finite number of steps, polynomial in the number of states in the system and specification models (Wonham and Ramadge 1987). All properties of interest are also trivially decidable in this case, as they can be checked by searching a finite state space. For supervisors modeled as deterministic Petri nets, which admits the possibility of nonregular specification languages that cannot be modeled by finite state automata, Giua and DiCesare (1994a, 1995), have obtained the following results (in the following E denotes the Petri net obtained by concurrent composition as described in the previous section).

– E can always be constructed efficiently using the concurrent composition operator because the operator is applied to the net structure, which is always finite even when the reachability set is not. For deterministic Petri nets it is always possible to check if E is weakly nonblocking and controllable, as we discuss in the next section.

– If E is blocking, we need to modify the structure of the net so that no blocking marking may be reached. For the marked language, however, the trimming of the net may not be possible because there exist languages in $\mathcal{L}_d$ whose prefix closure is not a P-type Petri net language. A good class of languages is the class of *deterministic P-closed* Petri net languages, defined as:

$$\mathcal{L}_{DP} = \{L \in \mathcal{L} \mid \overline{L} \in \mathcal{P}_d\},$$

i.e., as the set of all L-type Petri net (not necessarily deterministic) languages whose prefix closure can be generated by a deterministic nonblocking Petri net generator (Giua and DiCesare 1994a). It is the case that $\mathcal{G}_d \in \mathcal{L}_{DP}$, which means it is always possible to trim a weakly blocking net.

– If $L(E)$ is not controllable, it may not always be possible to construct a net E' such that $L(E') = L(E)^{\uparrow}$. In fact, the classes $\mathcal{P}_d$, $\mathcal{L}_d$, and $\mathcal{G}_d$, are not closed under the supremal controllable sublanguage operator.

These results show that a Petri net supervisor does not always exist for control problems where plant and specifications are nonregular. Nevertheless, supervisors for nonregular Petri net languages can often be constructed by intuition. Supervisory design for infinite state plants with regular specifications using Petri nets is considered in Sreenivas and Krogh (1992).

4.4. Decidability Properties of Petri Net Languages

Although Petri nets have a greater modeling power than finite state machines, computability theory shows that the increase of modeling power often leads to an increase in computation required to solve problems. As an example, all properties of interest of finite state machines are decidable since they may be checked with a finite procedure. On the other end, if we consider very powerful models, such as Turing machines, even simple problems, such as the halting problem, are undecidable. In this section we discuss some issues related to the decidability properties of Petri nets obtained by studying the corresponding languages and show that Petri nets represent a good trade-off between modeling power and analysis capabilities.

We first clarify which class of Petri net languages we are considering. In section 2.1 we presented the ordinary Petri net model that will be considered in this section. A common extension to that model is the so called "Petri net with inhibitory arcs" (PNIA) (Peterson 1981). An inhibitory arc is an arc from place p to transition t that prevents the firing of t when p is marked. Sreenivas and Krogh (1992) have pointed out that since PNIA are linguistically equivalent to Turing machines, most properties of interest, such as controllability, are not decidable. Thus, we will not consider PNIA in this paper.

Another possible extension of the Petri net model presented is to allow weights on the arcs to consume and generate multiple tokens in the firing rule. These are called *general* Petri nets as opposed to *ordinary* Petri nets. Lafortune and Yoo have shown that pure ordinary Petri nets have the same descriptive power of (possibly not pure) general Petri nets, in the sense that they generate the same class of L-type languages (Lafortune and Yoo 1991). This justifies the choice of ordinary nets as basic models in this paper.

When defining the various classes of Petri net languages in section 2.3, we distinguished arbitrary languages from deterministic languages. There are clear advantages to restricting attention to the study of deterministic languages from the standpoint of decidability properties. To decide controllability of a language K with respect to a generator G we need to check for the subset inclusion $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. Sreenivas showed that this inclusion is decidable if K is a closed free-labeled language and G is a free-labeled Petri net generator (Sreenivas 1993). The proof is based on two steps. First, since free-labeled languages are closed under intersection it is possible to construct two nets generating the languages $\overline{K}\Sigma_u \cap L(G)$ and $\overline{K}$. Second, a net construction is given that reduces the subset inclusion to checking the reachability of a finite set of markings, and the reachability problem for Petri nets is known to be decidable (Kosaraju 1982, Mayr 1984). This proof may also be extended to deterministic closed languages.

To generalize these results one may use Lemma 1, which applies to deterministic Petri net languages, to prove the following theorems.

THEOREM 4 (Giua and DiCesare 1995) *It is possible to decide if a deterministic Petri net generator G is blocking with respect to a terminal language in $\mathcal{L}_{DP}$.*

Note that this theorem holds for all weak languages, but only for some marked languages. In fact, we have seen that there are marked languages that are not in $\mathcal{L}_{DP}$.

THEOREM 5 (Giua and DiCesare 1995) *It is possible to decide if a language $K \in \mathcal{L}_{DP}$ is controllable with respect to a Petri net generator G.*

Sreenivas has also discussed two different notions of controllability (Sreenivas 1993). The strongest notion of controllability requires that we may also test for the inclusion: $K \subseteq L$. In this case it is necessary that G be a deterministic generator, as in the next proposition.

THEOREM 6 (Giua and DiCesare 1995) *It is possible to decide if a language $K \in \mathcal{L}_{DP}$ is: (1) controllable with respect to a deterministic Petri net generator G; and (2) contained in $L(G)$.*

To illustrate the complexity of these decision procedures, suppose we have two nets G_1 and G_2 whose closed (or marked or weak) behaviors are the languages L_1 and L_2. As we suggested when presenting Lemma 1, to check whether $L_1 \subseteq L_2$ we may follow these steps:

1. construct a Petri net G_2' generating the complement of L_2, i.e., the language $\mathbf{C}L_2$ (this is possible if G_2 is deterministic);

2. construct the net G as the intersection of the nets G_1 and G_2';

3. check whether the language generated by G is empty.

The first step may be carried out with the construction in Pelz (1987), whose complexity has not been computed. The second step may be done efficiently. We expect the last step to have the same complexity of checking the reachability of a given marking, which is at best decidable in exponential space (Jones, Landweber and Lien 1977, Jantzen 1987). Reutenauer (1990) has noted that the proof of decidability of the reachability problem due to Kosaraju (1982) and Mayr (1984) does not provide efficient algorithms for verifying whether a marking is reachable or not. Therefore, although many important properties are decidable in the Petri net framework, in general the computational complexity makes it impractical to solve all but the simplest problems algorithmically.

5. Linear Algebraic Approach

In this section we describe approaches relying on the linear algebraic representation of the Petri net model of the plant. We first consider an approach proposed by Li (1993) and Li and Wonham (1993, 1994) for computing state feedback policies based on the matrix-vector representation of the Petri net state transition equation (3). Under the NC assumption, which applies throughout this section, Li and Wonham consider the synthesis of maximally

permissive feedback policies when the allowable states are specified by a *linear predicate* P of the form

$$P = \{m \in \mathcal{M} \mid a^T \cdot m \leq b\} \tag{8}$$

where a is an n-vector and b is a scalar. The control objective is to guarantee the linear constraint is satisfied for all markings reachable under the control.

Li and Wonham assume a particular case of the CtlPN model in which the set of Petri net transitions, T, is decomposed into sets of controllable transitions, T_c, and uncontrollable transitions T_u. The set of admissible markings corresponding to a predicate P of the form (8), denoted as $[P]$ by Li and Wonham, is given by

$$[P] = \{m \in \mathcal{M} \mid (\forall \sigma \in T_u^*, m[\sigma\rangle m')\ a^T \cdot m' \leq b\}, \tag{9}$$

In words, an allowable marking is admissible if and only if all markings reachable by firing uncontrollable transitions satisfy (8).

The unique maximally permissive control input for a given $m \in [P]$, can be computed by an on-line controller as follows: for each controllable transition t state enabled in the plant, if $m[t\rangle m'$ with $m' \in [P]$ then t will also be enabled by the controller, else t will be disabled. That is, the controller enables each controllable transition which, if fired, leads to an admissible marking. Note that without the NC assumption, markings reached by simultaneous firings of transitions would have to be considered and a unique maximally permissive control may not exist for some markings.

It is evident that computing the control input becomes a matter of determining if a marking is in $[P]$. Li and Wonham show this problem can be reduced to solving a linear integer program provided the Petri net satisfies a particular structural condition, namely, the uncontrollable subnet N_u, i.e., the net obtained from N by removing all controllable transitions, must be *loop free*.

Using the state equation (3), let us rewrite equation (9) as follows:

$$\begin{aligned} [P] &= \{m \in \mathcal{M} \mid (\forall \sigma \in T_u, m[\sigma\rangle) a^T \cdot (m + E \cdot \overline{\sigma}) \leq b\} \\ &= \{m \in \mathcal{M} \mid a^T \cdot m + a^T \cdot E \cdot \overline{\sigma}^*) \leq b\}, \end{aligned} \tag{10}$$

where $\overline{\sigma}^*$ is the solution to the following optimization problem

$$\max_{\sigma \in T_u, m[\sigma\rangle} a^T \cdot E \cdot \overline{\sigma}. \tag{11}$$

The reduction of the optimization problem (11) to a *linear integer program* is based on the fact that, as we discussed in section 2.1, if the uncontrollable subnet N_u is loop free then equation (3) gives necessary and sufficient conditions for reachability under firing sequences σ containing only uncontrollable transitions. That is, if N_u is loop free, then $\sigma \in T_u$ can be fired from m if and only if $m + E \cdot \overline{\sigma} \geq 0$. Thus the optimization problem (11), becomes the *linear integer program*:

$$\begin{aligned} \max_{\overline{\sigma}} \quad & a^T \cdot E \cdot \overline{\sigma} \\ \text{s.t.} \quad & m + E \cdot \overline{\sigma} \geq 0 \\ & \overline{\sigma}(t) = 0 \text{ for } t \notin T_u \\ & \overline{\sigma} \geq 0 \end{aligned} \tag{12}$$

Li and Wonham develop this basic approach in several ways, including the generalization to multiple linear predicates (modular synthesis) and developing a closed-form expression for the maximally permissive control under further structural assumptions. They also consider sequential specifications in the form of linear predicates on the firing vector which they call *linear dynamic specifications*. For sequential specifications, they convert the problem to a state feedback problem using a *memory* as described in the previous section. In summary, the attraction of the general linear integer programming approach to Petri nets is that the synthesis of state feedback control policies is reduced to the solution of a standard optimization problem, thereby eliminating the need to compute the reachability graph for the Petri net.

A linear programming approach has also been used by Giua and DiCesare (1994) where *elementary composed state machines* (ECSM), a class of Petri nets with a convex reachability set, are defined. If the net E obtained by the monolithic supervisory design belongs to this class, then integer programming techniques may be used to validate properties of the closed loop system such as blocking and controllability.

The integer programming approach used by Li and Wonham exploits the structural properties of Petri nets. Linear specifications on the marking set of the form (8) where a and b are non-negative have also been considered by other authors and studied using structural Petri net analysis. Giua, DiCesare, and Silva (1992, 1993) have called these specifications *generalized mutual exclusion constraints*. Constraints of this kind can be enforced by *monitor* places on nets where all transitions are controllable. As an example, in Figure 6.a, the monitor place p_0 with its dotted arcs has been added to enforce the specification: $m(p_1) + m(p_2) \leq 2$. When some of the transitions are not controllable, and thus cannot be disabled by monitor places, a monitor based solution may not exist or may require an exceedingly high number of monitors.

Moody et al. (1994) and Yamalidou et al. (1996) have considered both linear specifications of the form (8) and linear specifications on the firing vector. They use a monitor based solution that they call *place invariant*, because the effect of the monitor place is that of creating an invariant on the net. As an example, in Figure 6.a, the addition of place p_0 creates the invariant: $m(p_0) + m(p_1) + m(p_2) = 2$. To determine the invariants that must be added to the net, the linear constraints on the marking or transition firings are framed in terms of a linear matrix inequality. This linear inequality is then converted into an equality through the addition of slack variables which represent the controller places. In Moody and Antsaklis (1995) and Moody, Antsaklis and Lemmon (1996), the method is extended to explicitly consider uncontrollable and unobservable transitions.

It is interesting to note that the use of monitors is not restricted to the case of nonnegative coefficients a and b. Basically the same construction may be used in the general case with negative coefficients. As an example, the monitor in Figure 6.b enforces the specification: $m(p_1) - m(p_2) \leq 2$. The sign of the inequalities may also be reversed. As an example, the monitor in Figure 6.c enforces the specification: $m(p_1) + m(p_2) \geq 2$.

Finally, we remark that there is a fundamental difference between the approach of Li and Wonham, and Giua et al., and the approach of Moody et al. In fact, in the first two approaches special PN structures are considered, for which the maximally permissible control policy can be easily computed and implemented. In the latter approach, the requirement that

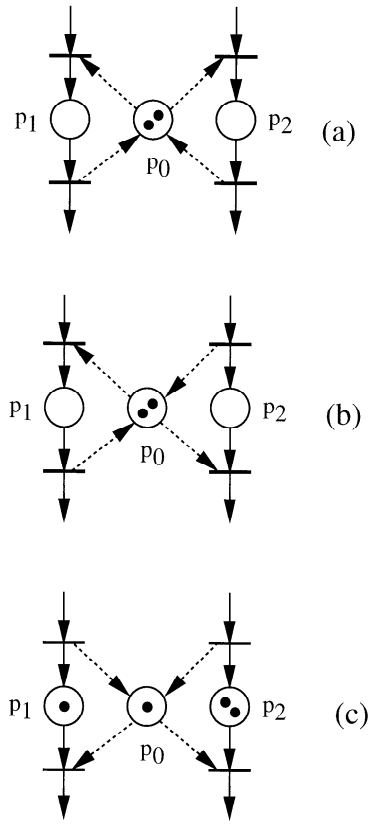

Figure 6. Examples of monitor places.

the control policy be maximally permissible is given up, and one is willing to accept a more restrictive control policy that can however be easily computed and implemented with monitor places.

6. Path-Based Algorithms

Path-based algorithms approach the forbidden state control problem by controlling the flow of tokens within directed paths within the CtlPN. Path control policies use the structure of the CtlPN to determine how the flow of tokens will lead to forbidden markings. Path control has been used to address a variety of control specifications, including state control (Holloway and Krogh 1990, Krogh and Holloway 1991, Boel, Ben-Naoum and Van Breusegem 1995), sequence control (Holloway and Hossain 1992), and distributed control (Haoxun and Baosheng 1991). For some classes of CtlPNs it has been shown that this exploitation of

the net structure can lead to significant gains in computational efficiency for on-line control synthesis (Krogh, Magott and Holloway 1991).

Path control for forbidden state problems assumes the set of forbidden states is expressed in a special form, called *forbidden conditions*, which are specifications of sets of forbidden markings based on linear inequalities on the marking vectors. The most general forbidden condition, introduced by Boel, Ben-Naoum, and Van Breusegem (1995), is represented by the triple (F, v, k), where $F \subseteq P$ is a subset of places, v: $F \longrightarrow \mathbf{N}$ is a weighting function over the places in F, and k is the *threshold* of the forbidden condition. The forbidden condition specifies a set of forbidden markings as

$$M_{(F,v,k)} := \{m \in \mathcal{M} \mid \sum_{p \in F} m(p)v(p) > k\}. \tag{13}$$

A marking is forbidden according to the condition (F, v, k) if the weighted sum of tokens is greater than the threshold k. Given a collection $\mathcal{F}$ of forbidden conditions, the set of forbidden markings for $\mathcal{F}$ is

$$M_{\mathcal{F}} := \{m \in M_{(F,v,k)} \mid (F, v, k) \in \mathcal{F}\}. \tag{14}$$

Note that the linear inequality in the definition of $M_{(F,v,k)}$ defines a set of minimal markings in $M_{(F,v,k)}$, where any other marking which covers (in the Petri net sense) any of these minimal markings is also in $\mathcal{M}_{(F,v,k)}$. In the general case, some arbitrary forbidden marking sets may not be representable as a forbidden condition, and it becomes necessary instead to consider a superset or subset of the forbidden markings. However, for the special case of live and safe cyclic controlled marked graphs, it has been shown that any subset of live and safe markings can be defined by a set $\mathcal{F}$ of forbidden set conditions (Holloway 1988).

Given a set of forbidden conditions, to prevent the reachability of a marking $m \in \mathcal{M}_{\mathcal{F}}$ the controller must control the number of tokens in the places in set F such that the inequality in (13) is not satisfied for each $(F, v, k) \in \mathcal{F}$. Path control methods avoid direct construction of the reachability graph by characterizing the reachability of forbidden conditions in terms of the markings of paths in the CtlPN.

6.1. Characterizing Uncontrollable Reachability

A *path* is an alternating sequence of transitions and places along a directed path in the CtlPN. We exclusively consider paths which begin with a transition and end with a state place. For a given path $\pi = (t_0 p_0 \ldots t_n p_n)$, the starting transition t_0 is denoted t_π. We define set operations on paths to be over the sets of places and transitions in the path. Thus, $\pi \cap P$ is the set of places in the path, and $\pi \cap T$ is the set of transitions in the path. We extend the marking notation such that $m(\pi)$ is the sum of $m(p)$ over all $p \in \pi$.

The set $\Pi_c(p)$ indicates the set of all *precedence paths* for a place p, where $\pi = (t_0 p_0 \ldots t_n p_n) \in \Pi_c(p)$ implies that: (1) path π ends at place p; (2) t_π is a controlled transition or has ${}^{(p)}t_\pi \cap \pi \neq \emptyset$; (3) all other transitions t in the path are uncontrolled transitions; and (4) a path does not include a cycle except possibly at the initial transition t_π. A path set $\Pi_c(p)$ is said to be a marked graph structure if for any state place $p' \in \pi$ and any $\pi \in \Pi_c(p)$, p' has at most one input transition and at most one output transition among all

paths in $\Pi_c(p)$. The path set $\Pi_c(p)$ is said to be a state graph structure if for any transition $t \in \pi$ and any $\pi \in \Pi_c(p)$, t has at most one input state place and one output state place among all paths in $\Pi_c(p)$.

The number of tokens that can uncontrollably mark a place p from a current marking m depends upon the number of tokens in the paths $\Pi_c(p)$ under the current marking. This is seen by considering the question: For a given marking m, does there exist an uncontrollably reachable marking, $m' \in R_\infty(m, u_{zero})$, for which $m'(p) \geq k$? This problem is referred to as the *uncontrollable k-coverability problem*. The role of precedence paths in determining the uncontrollable k-coverability problem is most easily illustrated by considering the case where $k = 1$. For a marking m and a precedence path π, define the predicate

$$\Lambda_m(\pi) := \begin{cases} 1 & \text{if } m(p) \geq 1 \text{ for some } p \in \pi \\ 0 & \text{else} \end{cases} \tag{15}$$

This then leads to the following complementary results.

THEOREM 7 (Holloway and Krogh 1990) *Given a CtlPN N^c and place p such that $\Pi_c(p) = \{\pi_1, \pi_2, \ldots \pi_n\}$ is a marked graph structure, for a marking m there exists an uncontrollably reachable marking $m' \in R_\infty(m, u_{zero})$ with $m'(p) \geq 1$ if and only if*

$$\Lambda_m(\pi_1) \odot \Lambda_m(\pi_2) \odot \cdots \Lambda_m(\pi_n) = 1 \tag{16}$$

where $\odot$ indicates the Boolean AND *(conjunction) operator.*

THEOREM 8 *(follows from both Boel, Ben-Naoum and Van Breusegem (1993) and Holloway, Guan and Zhang (1996)) Given a CtlPN N^c and place p such that $\Pi_c(p) = \{\pi_1, \pi_2, \ldots \pi_n\}$ is a state graph structure, for a marking m there exists an uncontrollably reachable marking $m' \in R_\infty(m, u_{zero})$ with $m'(p) \geq 1$ if and only if*

$$\Lambda_m(\pi_1) \oplus \Lambda_m(\pi_2) \oplus \cdots \Lambda_m(\pi_n) = 1 \tag{17}$$

where $\oplus$ indicates the Boolean OR *(disjunction) operator.*

The above theorems show that the marked graph structure and state graph structure lead to the complementary characterizations of the uncontrolled 1-coverability of a place. For the marked graph case, the coverability is characterized by a Boolean conjunction of path predicates. For the state graph structure, the coverability is characterized by a Boolean disjunction of the path predicates. These notions are generalized by Holloway, Guan, and Zhang (1996) for a general class of CtlPNs, where the uncontrolled region of the net leading to a given place may have a very general mixture of both marked graph structures and state graph structures. An algebra is defined containing forms of both disjunctive and conjunctive operations to characterize uncontrolled 1-coverability as an expression with a Boolean evaluation. A related method of characterizing uncontrolled reachability is proposed by Hanisch et al. for a class of systems modeled by interacting Petri nets (condition-event nets) under partial observation (Hanisch, Luder and Rausch 1996). Uncontrolled 1-converability for colored controlled Petri nets is considered in Ashley and Holloway (1994). Xing et al. present a dynamic programming method for characterizing uncontrollable reachability for a general class of Petri nets (Xing, Li and Hu 1996).

Uncontrolled k-coverability does not have the same convenient Boolean characterization as the uncontrolled 1-coverability problem has. The complementary nature of state graph structures and marked graph structures is still evident, however, in the following results.

THEOREM 9 *Consider a CtlPN N^c and place p such that $\Pi_c(p)$ is a marked graph structure with $t_\pi \in T_c$ for all $\pi \in \Pi_c(p)$. For a marking m there exists an uncontrollably reachable marking $m' \in R_\infty(m, u_{zero})$ with $m'(p) \geq k$ if and only if $m(\pi) \geq k$ for all $\pi \in \Pi_c(p)$.*

THEOREM 10 (Boel, Ben-Naoum and Van Breusegem 1993) *Given a CtlPN N^c and place p such that $\Pi_c(p)$ is a state graph structure with $t_\pi \in T_c$ for all $\pi \in \Pi_c(p)$, for a marking m there exists an uncontrollably reachable marking $m' \in R_\infty(m, u_{zero})$ with $m'(p) \geq k$ if and only if*

$$\sum_{p' \in \pi, \pi \in \Pi_c(p)} m(p') \geq k$$

The proof of theorem 9 is similar to the proof of theorem 7 from (Holloway and Krogh 1990). As shown by both of the above theorems, for given net structures we can characterize the uncontrolled k-coverability of a given place in terms of the current marking of its precedence paths. Defining the predicate $\Lambda_m(p, k)$ to represent the uncontrolled k-coverability of a place p under a marking m, i.e. $\Lambda_m(p, k) = 1$ if and only if there exists some $m' \in R_\infty(m, u_{zero})$ with $m'(p) \geq k$, for a given marking m, the value of the predicate $\Lambda_m(p, k)$ can be directly computed from the Theorems 7 through 10 for marked graph and state graph precedence path structures. For more general net structures, the predicate $\Lambda_m(p, 1)$ can be computed directly from the markings of the precedence paths in a similar manner (Holloway, Guan and Zhang 1996).

6.2. Determining Control using path Predicates

Characterizing uncontrollable reachability as done in the previous section is the first step towards determining a control for avoiding forbidden states. When the uncontrollable reachability problem is broken into conditions on the number of tokens in precedence paths, the state avoidance control becomes a problem of regulating the entry of tokens into these paths.

We illustrate the basic approach of path-based control with the method of Holloway and Krogh (1990) and Krogh and Holloway (1991) for controlled marked graphs with safe (binary) markings. The reader is referred to Boel, Ben-Naoum and Van Breusegem (1993) and Holloway, Guan and Zhang (1996) for control laws for other net structures. Let $\mathcal{F}$ be a set of forbidden conditions of the form (F, v, k), where $v(p) = 1$ for all $p \in F$ and $1 \leq k < |F|$. For a forbidden condition $(F, v, k) \in \mathcal{F}$ and marking m, define $L_F(m)$ as the set of places in G such that $\Lambda_m(\pi) = 1$ for all $\pi \in \Pi_c(p)$. $L_F(m)$ is thus the set of places which already can be uncontrollably marked. Control must ensure that $|L_F(m)| \leq k$ for all reachable markings. Let $B_F(m)$ be the set of places not in $L_F(m)$ such that for each $\pi \in \Pi_c(p)$, either $\Lambda_m(\pi) = 1$ or else t_π is state enabled. $B_F(m)$ is thus the set of places which could join set $L_F(m)$ in a next marking unless the appropriate controls are disabled.

For any place $p \in B_F(m)$, we can prevent p from becoming uncontrollably reachable by disabling any controlled transition t_π, $\pi \in \Pi_c(p)$, for which $\Lambda_m(\pi) = 0$. Let $D_F(m, u) \subseteq B_F(m)$ be the set of places for which u disables such a transition. It can then be shown Krogh and Holloway (1991) under the appropriate assumptions that for any marking $m \in \mathcal{A}(\mathcal{M}_F)$, the maximally permissive control policy $\mathcal{U}_F(m)$ is the set of controls for which

$$|L_F(m)| + |B_F(m)| - |D_F(m, u)| \leq k \tag{18}$$

The control synthesis procedure for determining control policy U_F is divided into an off-line algorithm and an on-line algorithm (Krogh and Holloway 1991). In the off-line algorithm, the precedence paths for places $p \in F$ for $(F, v, k) \in \mathcal{F}$ are identified. The on-line computations for determining a control $u \in U_{\mathcal{F}}(m)$ for a marking $m \in \mathcal{A}(\mathcal{M}_{\mathcal{F}})$ then use condition (18) (Krogh and Holloway 1991).

The complexity of a the path-based control described above method for marked graphs is examined in (Krogh, Magott and Holloway 1991) where it is shown that the algorithm is polynomial in the number of the transitions in the Petri net and in the number of forbidden set conditions. The maximally permissive elements of the control set $U_{\mathcal{F}}(m)$ can be identified using the results of Holloway and Krogh (1990).

7. Feedback Control for Liveness and Deadlock Avoidance

Another class of specifications for closed-loop behavior pertain to ensuring liveness or avoiding deadlocks in a system. Feedback control for avoiding deadlocks is related to the notion of closed-loop control for state avoidance, presented in section 3. It is not sufficient for the controller to simply avoid the known deadlock states of the uncontrolled system, however, since in doing so the controller may impose new deadlock states in the closed loop system. The notion of avoiding deadlocks can also be related to the concept of nonblocking, given in section 4.1. A control is defined to be nonblocking if a final marking is reachable from every reachable marking. Nonblocking implies an activity can always proceed to completion, and thus will not encounter a deadlock before reaching a final state. The final state may be a deadlock state, however, so the resulting closed-loop system may not be live. Alternatively, a system may be live, but a final state may not be reachable. In this section, we focus only on controllers to ensure liveness or absence of deadlocks. Much research has been done on conditions to verify liveness in uncontrolled Petri nets (Murata 1989), but only recently has the problem avoiding deadlocks through external control policies been considered.

Holloway and Krogh (1992) consider liveness of controlled marked graphs operating under the forbidden state avoidance control policy described in section 6. Requirements on the forbidden conditions were determined and the forbidden state control synthesis algorithm was extended to ensure that the Petri net is live under the maximally permissive control policy.

Viswanadham, Narahari, and Johnson (1990) propose a deadlock avoidance policy based on a state lookahead. They consider systems modeled by Generalized Stochastic Petri Nets (GSPNs). Given a current marking, the controller determines all markings reachable within a given number of transition firings. By identifying deadlocks in this set of markings,

the controller can disable transition firings which lead to these deadlocks. It is possible, however, that the lookahead horizon will not be long enough to identify deadlock states before they become inevitable. In such cases, a deadlock can occur, and some recovery action must be initiated.

Banaszak and Krogh (1990) present a deadlock avoidance algorithm (DAA) for prevention of deadlocks in a class of models for automated manufacturing systems. In the manufacturing systems considered, deadlock must be prevented by avoiding jobs entering into circular waits for resources. The Petri net models used are composed of resource places and *production sequences*, sequences of places and transitions corresponding to steps that a job must undergo. It is assumed that each step in a production sequence is associated with exactly one resource, and every transition may be externally disabled.

Each production sequence is divided into *zones* consisting of a sequence of steps requiring resources that are shared with other production sequences, followed by a sequence of steps requiring resources that are not shared with other production sequences. The DAA avoids deadlock by enforcing two conditions. First, the total number of tokens in a zone must not exceed the total number of unshared resources associated with the zone. Second, a job can only claim a shared resource if all subsequent shared resources in the zone are also available. The DAA enforces these conditions by disabling transitions associated with the entry to a zone or with the start of each job step requiring a shared resource. Banaszak and Krogh prove that the resulting closed-loop behavior will not have deadlocks.

The deadlock avoidance algorithm (DAC) of Hsieh and Chang (1994) applies to a slightly more general class of models than the method of Banaszak and Krogh because it allows a jobs to claim more than one resource at a time. The class of models, *Controlled Production Petri Nets* (CPPN), is again motivated by manufacturing systems. The models consist of job subnets, resource places, and control places. Like the production sequences of Banaszak and Krogh (1990), each job subnet is a sequence of places and transitions representing the sequence of activities necessary to complete a job. Arcs between the job subnets and a resource place indicate the resource is used in a job sequence. Arcs from control places to the job subnet indicate which transition firings can be disabled by the external control.

The CPPN can be decomposed and analyzed to determine a minimal number of resources necessary to complete any job. This defines a minimal marking which must always be coverable from any subsequent marking. Liveness can be guaranteed if the current control always ensures that this minimal marking is coverable from all next possible markings. Since determining coverability is not computationally feasible except for small nets, a sufficiency test is established that determines if this coverability condition is satisfied under a particular *job-clearance transition firing rule*. This test is sufficient to determine if a control under consideration ensures coverability of the minimal marking, and thus will maintain liveness. A dispatching policy proposes an initial control to evaluate based on the current marking. If the proposed control does not pass the liveness sufficiency test, then a sequence of more restrictive controls are evaluated until one is found that satisfies the test.

Hsieh and Chang emphasize that the DAC guarantees liveness, which is stronger than guaranteeing deadlock avoidance. The algorithm is shown to have polynomial complexity. The algorithm is shown to be strictly more permissive than the DAA of Banaszak and Krogh (1990), as well as apply to a somewhat larger class of models.

Ezpeleta, Colom, and Martinez generalize the production models of Hsieh and Chang (1994) and Banaszak and Krogh (1990) by allowing choice in the job subnets, indicating alternative job routings or resource utilizations (Ezpeleta, Colom and Martinez 1995). Their control policy depends on the notion of a *siphon*, a set of places P such that ${}^{(t)}P \subseteq P^{(t)}$. Once all places in a siphon become unmarked, the places will remain unmarked and the transitions in $P^{(t)}$ will never again be enabled. For the class of models considered, liveness of the system can be ensured by preventing any siphon from becoming unmarked.

The control policy of Ezpeleta et al. is implemented by creating a Petri net supervisor which is run in synchrony with the plant. The supervisor consists of the model of the plant with additional places and arcs to modify the net operation. One place is added for each siphon in the uncontrolled net, and its initial marking is set at one less token than the initial number of tokens in the siphon. Jobs that might remove a token from the siphon require a token from this place before they can start, in effect reserving the future use of a token from the siphon. The token is returned to the place after the job has used a token from the siphon or after an alternative routing is taken. This reserving of tokens before a job starts ensures that the siphon will never become entirely unmarked.

The deadlock prevention control of Ezpeleta et al. requires an off-line determination of all minimal siphons in the net. The number of minimal siphons can be at worst case exponential, so such computations may be lengthy or infeasible for some systems. Once the minimal siphons have been determined, however, the on-line control is very rapid and simple. The on-line control only requires a check of the number of tokens in the controller places to determine which transitions at the start of jobs are enabled.

Recently, Xing, Hu, and Chen (1996) considered deadlock avoidance policies in a class of models similar to those considered in Banaszak and Krogh (1990), where the model consists of resource places and production sequences. A deadlock structure (DS) is a set of transitions (excluding production sequence initiation transitions) for which the set of resources used in the output places of the transition set is equal to the set of resources used in the input places of the transition set. It is shown that if the number of resources used by the deadlock structure equals the capacity of the resource, the system will be in deadlock. The control policy then ensures that for each involved resource, the deadlock structure always has less than that resource's capacity. One claimed advantage of the method is that it can be easily enforced by creating a p-invariant that ensures that the number of a resource involved in a deadlock structure will be strictly less than the capacity of the resource.

Sreenivas (1996) considers deadlock avoidance for Petri nets with no restrictions except that all transitions are individually controllable. Sreenivas's deadlock avoidance algorithm relies on the fact that liveness can be assured from a marking m_1 if the following condition is met: there exists firing sequences σ_1, σ_2, and markings m_2, m_3 such that: (1) $m_1[\sigma_1 > m_2[\sigma_2 > m_3$; (2) $m_3(p) \geq m_2(p)$ for all p; and (3) σ_2 includes the firing of every transition at least once. Sreenivas' on-line control for a given marking considers each enabled transition and the marking that would result. If the resultant marking satisfies the above condition, then the control enables that transition. It is proved that the closed loop system under this control policy will be live if and only if the initial marking also satisfies the condition.

Sreenivas gives a procedure to verify the above condition. The procedure relies on the coverability graph (Peterson 1981), and consists of three steps. First, the coverability graph from the considered next marking is constructed. Second, for any two nodes (markings) m_2 and m_3 in the graph such that $m_3(p) \geq m_2(p)$ for all p, a finite state automaton is created from the subgraph between m_2 and m_3. Finally, the language of this automata is tested for nonempty intersection with the language $L'_m = \{\tau \in T^* \mid \text{all transitions fire in } \tau\}$. The condition is decidable since there are known algorithms for each step. The required construction of the coverability graph for testing each potential marking is computationally expensive, however, and would not be reasonable for on-line computation except for very simple Petri nets. If the Petri net has a finite number of markings, then the coverability graph could be evaluated off-line, and the appropriate control for each marking could be determined in advance and stored in a table.

8. Extensions to Timed Petri Nets

Recently a number of researchers have been interested in developing methods for extending methods for synthesizing feedback control policies for untimed (logical) models of DESs to models and specifications which include explicit representations of real time (Brave and Heymann 1988, Ostroff and Wonham 1990, Wong-Toi and Hoffman 1991, Brandin and Wonham 1992, Sathaye and Krogh 1993, Takae et al. 1996). Petri nets offer an attractive framework for developing these extensions for controlled DESs because there are established methods for introducing and analyzing timing in uncontrolled Petri net models (Murata 1989). In this section we briefly describe two investigations of feedback control policies for *controlled time Petri nets* (CtlTPNs), an extension of untimed CtlPN models that includes real-time constraints on the firing times for enabled transitions. We first introduce the CtlTPN model for timed DES plant dynamics. We then discuss two approaches for control of CtlPN's with marked graph structures: the work of Brave and Krogh (1993) on extensions of the path-based approach for forbidden marking specifications, and the work of Cofer and Garg (1995, 1996) applying the max-plus algebra approach.

A CtlTPN N_T^c is a CtlPN N^c with a *firing interval* associated with each uncontrolled transition. Assuming the set of n transitions in N^c is indexed as $T = \{t_1, \ldots, t_{n_u}, \ldots, t_n\}$, where $n_u \leq n$ is the number of uncontrolled transitions given by the set $T_u = \{t_1, \ldots, t_{m_u}\}$, the firing interval I_k associated with transition t_k for $k \in \{1, \ldots, m_u\}$ is a closed interval in the extended nonnegative real numbers, $\Re^+ \bigcup \{\infty\}$. Let $minI_k$ and $maxI_k$ denote the lower and upper bounds on the interval I_k, i.e., $I_k = [minI_k, maxI_k]$ with $0 \leq minI_k \leq maxI_k \leq \infty$. The firing intervals define constraints on the firing times for the uncontrolled transitions as follows. If transition $t_k \in T_u$ becomes enabled at time $\tau = \tau_0$, then t_k cannot fire before time $\tau = \tau_0 + minI_k$, provided t_k has remained enabled throughout the interval $[\tau_0, \tau_0 + minI_k]$ (i.e., assuming t_k does not become disabled by the firing of some other transition with which it shares an input place). On the other hand, if transition $t_k \in T_u$ becomes enabled at time $\tau = \tau_0$ and remains enabled, then t_k *must* fire before or at time $\tau = \tau_0 + maxI_k$.

To define the evolution of the marking and transition firings in real time for a CtlTPN, *clocks* are introduced for the uncontrolled transitions. Specifically, $l_k \in \{-1\} \bigcup \Re^+$ denotes

the clock for transition $t_k \in T_u$, where for a marking $m \in \mathcal{M}$ valid values for l_k are:

$$l_k \begin{cases} = -1 & \text{if } t_k \text{ is not state-enabled by } m \\ \in [0, max\, I_k) & \text{if } t_k \text{ is state-enabled by } m \end{cases} \tag{19}$$

The *state* of a CtlTPN at any instant is given by an ordered pair $x = (m, l)$ where m is the marking and $l = (l_1, \ldots, l_{m_u})^T$ is the vector of clock values for the uncontrolled transitions. As a function of time, $x(\tau) = (m(\tau), l(\tau))$ evolves according to the firing rule described above for uncontrolled transitions and the control influence on the controlled transitions as described in the following paragraph. For a complete formal development of the transition firing rules based on firing intervals, which were first introduced by Merlin who defined the class of uncontrolled Petri nets called *time Petri nets*, the reader is referred to Sathaye and Krogh (1993), Brave and Krogh (1993).

A primary difference between CtlPNs and CtlTPNs is the influence of control inputs. For CtlPNs, the controls are only inhibiting in nature, that is, the control input can prohibit controlled transitions from firing, but it cannot force a transition to fire at a particular instant. In the case of CtlTPNs as defined in Sathaye and Krogh (1993), Brave and Krogh (1993), control inputs are *forcing* inputs. It is assumed the controlled transitions are controlled independently, so a *control* v is defined simply in terms of the controlled transitions which are to be forced for fire, i.e., $v \subseteq T_c$. Letting $v(\tau)$ denote the control input at time τ, a controlled transition $t \in v(\tau)$ *must* fire if t is state-enabled by the marking $m(\tau^-)$, unless another transition fires at time τ which disables transition t. Note that since the firing of a transition at time τ changes the marking, the firing condition is defined in terms of $m(\tau^-)$ and the state trajectory (marking and clocks) is defined to be right continuous with respect to time. Sathaye and Krogh (1993) develop an approach to event-based sequential control of CtlTPNs under the NC assumption and allows only singleton control inputs, i.e., $|v(\tau)| \leq 1$ for all $\tau \in \Re^+$. Brave and Krogh (1993) allow concurrent firing of transitions in the development of state feedback policies for forbidden marking specifications for CtlTPNs and admit arbitrary subsets of controlled transitions as control inputs. Holloway (1996) considers the timing relationships between portions of a net model to determine whether control objectives can be met, and gives a simple example where the control objectives can only be met by considering the time delays within the system.

Brave and Krogh considered the forbidden state problem for controlled time marked graphs (CtlTMGs), which are CtlTPNs with marked graph structure. In their problem formulation the forbidden markings are specified by a collection of forbidden sets as described in Section 6 and the state feedback policy V is a mapping from the complete state space (markings and clocks) into subsets of controlled transitions; that is, $V\colon X \longrightarrow 2^{T_c}$. Since the controller is monitoring the system state rather than events, the sampling times must be generated by some external clock. Brave and Krogh consider feedback policies that are robust with respect to the sampling time sequence in the sense that the state feedback policy does not assume knowledge of when the future samples will be available. Moreover, the controller is not allowed to schedule future events; specifically, the controller can generate inputs which force state-enabled controlled transitions to fire only at the (unknown) sampling instants generated by an external clock. Given this control scenario described above,

the feedback policy must guarantee that no forbidden markings can occur any time in the future when no controlled inputs are fired.

Computation of the maximally permissive feedback policy for a given forbidden marking specification is based on the notion of the *sojourn times* for tokens in the critical places that make up the forbidden set conditions. Brave and Krogh show that with assumptions very similar to those made for the untimed case, the maximally permissive feedback policy can be stated and implemented with computations that are polynomial in the number of transitions in the CtlTMG model and the number of forbidden set conditions. The computations basically require the solution of a set of elementary minimal path problems that determine the sojourn times. The maximally permissive state feedback policy fires a maximal number of controlled transitions for which the forbidden set conditions will not be violated. Details can be found in Brave and Krogh (1993) which includes an illustration of how the use of the CtlTMG model can lead to less conservative control policies when compared to the maximally permissive policy for the untimed marked graph model. Takae et al. present a general condition for the existence of maximally permissive controllers for CtlTPNs in Takae et al. (1996).

Cofer and Garg (1995, 1996) consider the problem of controlling the timed behaviors of Petri nets using a max-plus algebra approach. The dynamics of a timed event graph can be analyzed as a linear system in a max-plus algebra, where maximization and addition replace the conventional operations of addition and multiplication, respectively. (An introduction to the max-plus algebra approach to the modeling and analysis of systems can be found in Bacelli et al. (1992), Cohen, et al. (1989). Cofer and Garg consider two control problems: the first problem is to maximize the delay in the system while ensuring that a minimal schedule is met (Cofer and Garg 1996), and the second problem is to enforce a minimal delay within the system to ensure that specific events always have a minimum separation time between occurrences (Cofer and Garg 1995, Cofer 1995).

The systems considered by Cofer and Garg are controlled timed event graphs, defined by a pair (T, A) where T is an indexed set of transitions and A is a *delay matrix*. If there exists a place between transitions t_i and t_j, the element A_{ij} is of the form $a\gamma^m$, where a is the delay associated with that place and γ is an index backshift function. The element $A_{ij} = a\gamma^m$ is to be interpreted (in the max plus algebra) that the kth firing of transition t_j will involve a token from the $k - m$ firing of t_i, and this token must reside in the place for a time units before it can participate in the firing of transition t_j. If no place connects the transitions, then $A_{ij} = -\infty$, which is the null element under the maximization operation. A specified subset T_c of the transitions can be externally controlled to delay their firing.

The kth firing time of each transition t_i is denoted $x_i[k]$. The sequence of vectors of these firing times satisfies the recursive equation

$$x = Ax \oplus v \oplus u$$

where x is the sequence of vectors of firing times, the multiplication operation represents addition, the $\oplus$ operation represents maximization, v represents an initial condition on token times, and u is the sequence of externally imposed delays.

In (Cofer and Garg 1996), the control goal is to restrict the behavior of the system so that all firing time vector sequences x satisfy $x \leq y$, for a given schedule represented by a

maximal firing time vector sequence y. It is shown that such a specification is achievable if and only if

$$A^*(I_c y \oplus v) \leq y$$

where $A^* = \bigoplus_{i=0}^{\infty} A^i$ and I_c is the matrix with identity on the diagonal elements corresponding to controllable transitions (Cofer and Garg 1996). If the specification is not achievable, then an algorithm can be used to determine a supremal achievable specification $y' < y$. This means that the control will give maximum delays within the system without exceeding the given upper limit delay specification y.

The complementary control problem is considered in Cofer and Garg (1995), Cofer (1995): control the system such that a set S of minimal event separation time specifications is met. The optimal control for this case is to impose the minimum delays on the system while meeting the specified minimal event separation times. The solution proposed for this approach is implemented as the synchronous composition of a controller timed event graph with the timed event graph model of the plant. Max-plus algebra analysis is used to help determine the necessary delays on the places in the controller, and to determine which controlled transitions will be inputs and or outputs of the places.

Boissel and Kantor (1995) address the control synthesis problem for timed controlled Petri nets using simulated annealing. Given a Petri net model of the plant with transitions which may be uncontrollable or unobservable, the problem is to determine a controller, also represented as a Petri net, such that the closed loop system avoids forbidden states, is live, and minimizes certain delays. The method begins by considering a controller as a set of places with connections to plant transitions according to the observability and controllability constraints. Simulated annealing techniques are used to perturb the connections between places within the controller and the connections between the controller and the plant. An objective function is used to evaluate the effectiveness of the resultant controller, and eventually the investigation of new controller designs ceases when new designs cease to offer improved effectiveness.

9. Directions for Future Research

This paper surveys research on feedback control policies for discrete event systems using Petri net models. The primary objective in this research is to develop modeling, analysis and synthesis procedures that take advantage of the structural properties of Petri nets to reduce computational complexity. Toward this end there are several open directions for further research.

A complete formal comparison of the computational complexity of Petri net methods versus unstructured automata-based approaches has not been conducted to date. A comparison of the complexity for one class of problems is presented in Krogh, Magott and Holloway (1991) where it is shown that the computational complexity of computing maximally permissive state feedback problems for forbidden state problems for DESs that can be modeled as controlled marked graphs (CtlMGs) is polynomial in the number of transitions and the number of set conditions in the forbidden state specifications. This compares favorably to

the equivalent problem using an unstructured automaton model of the plant dynamics since the number of states in this case is exponential with respect to the number of transitions in the CtlMG model, and the automata-based computations are polynomial in the number of states in the automaton model. There are efficient methods for solving classes of problems using automata models of independent concurrent systems (Ramadge 1989), but it has been shown that forbidden state problems for synchronized concurrent systems are in general computationally intractable (Golaszewski and Ramadge 1988b). CtlMGs fall in between these two classes of automata-based problems. The boundary between tractable and intractable problems needs to be explored more deeply.

Various classes of Petri net models have been studied thoroughly in the literature, both with respect to modeling power and computational complexity for various problems of analysis. Among these classes we have discussed acyclic Petri nets, marked graphs and state machines. Interesting extensions of marked graphs that have been object of research are the classes of *forward conflict-free nets*, i.e., nets such that $|p^{(t)}| = 1$, and *backward conflict-free nets*, i.e., nets such that $|{}^{(t)}p| = 1$ (Haoxun 1994).

An interesting extension of state machines that may be worth considering in the context of supervisory control are the classes of *fork-free nets*, i.e., nets such that $|t^{(p)}| = 1$, and *joint-free nets*, i.e., nets such that $|{}^{(p)}t| = 1$. An even richer class is that of *free-choice nets*, i.e., nets such that each arc is either the only input arc of a transition or the only output arc of a place. Jones, Landweber and Lien (1977) showed that the reachability problem for general nets, i.e., deciding if a given marking is reachable, can be reduced to the reachability problem for free-choice nets. Since the former problem is known to be DSPACE (exp) hard, then the latter problem is of exponential complexity as well. However, the liveness problem for free-choice nets, i.e., deciding if a net is live, is NTIME (poly) complete, while the liveness problem for general nets has the same complexity of the reachability problem (Jones, Landweber and Lien 1977). In effect, the *rank theorem* (Desel 1992) shows that structural liveness for this class can be decided by computing the rank of the incidence matrix. This suggests that free-choice nets offer computational advantages when studying control problems that can be expressed as liveness problems (Sreenivas 1996).

Another interesting Petri net analysis technique worth exploring is called *analysis by transformation* (Berthelot 1987). A net N_1 is transformed, according to particular rules, into a net N_2 while maintaining the properties of interest. The analysis of the net N_2 is assumed to be simpler than the analysis of the net N_1. Properties usually considered in the Petri net literature are conservativeness, proper termination, existence of home states and liveness. Properties of interest in supervisory control, such as controllability, nonblockingness, etc., could be considered as well.

We have discussed in this paper interesting work that presented Ramadge-Wonham control concepts in colored Petri net context (Makungu, Barbeau and St-Denis 1994). Colored Petri nets offer a more compact and readable representation of large systems consisting of numerous similar interacting components. Only forbidden state avoidance problems have been considered so far. Interesting classes of specification language problems may also be solved with colored nets.

Finally, control of timed discrete event systems has just begun to receive attention in the literature and the work described in the previous section on controlled time Petri net models

represents a small portion of the problems that can be formulated. Perhaps a more thorough understanding of the needs for practical applications of the theory should be a guide to the types of problems that should be studied in depth. If this is the case, there is clearly a need for more research into tools for the actual implementation and application of control DES theory in general.

References

Ashley, Jr., J., and Holloway, L. E. 1994. Characterizing uncontrollable reachability for colored controlled petri nets. *Proceedings of 1994 IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Texas.

Baccelli, F., Cohen, G., Olsder, G., and Quadrat, J. P. 1992. *Synchronization and Linearity: An Algebra for Discrete Event Systems*. Wiley.

Banaszak, Z. A., and Krogh, B. H. 1990. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. on Robotics and Automation* 6(6): 724–734.

Barroso, G. C., Lima, A. M. N., and Perkusich, A. 1996. Supervision of discrete event systems using Petri nets and Supervisory Control theory. *Proc. of 1st Int. Workshop on Manufacturing and Petri Nets, 17th Int. Conf. on Application and Theory of Petri Nets*, Osaka, Japan, pp. 77–96.

Berthelot, G. 1987. Transformations and decompositions of nets. *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986*, (Brauer, W., Reisig, W., and Rosenberg, G. eds.) *Lecture Notes in Computer Science*, vol. 254-I, New York: Springer Verlag, pp. 359–376.

Best, E. 1987. Structural theory of Petri nets: the free choice haitus. *Lecture Notes in Computer Science*, vol. 254, New York: Springer Verlag, pp. 168–206.

Boel, R. K., Ben-Naoum, L., and Van Breusegem, V. 1993. On forbidden state problems for controlled closed controlled state machines. *Preprints of the 12th IFAC World Congress*, Vol. 4, Sidney, Australia, pp. 161–164.

Boel, R. K., Ben-Naoum, L., and Van Breusegem, V. 1995. On forbidden state problems for a class of colored Petri nets. *IEEE Trans. on Automatic Control* 40(1): 1717–1731.

Boissel, O. R., and Kantor, J. C. 1995. Optimal feedback control design for discrete-event systems using simulated annealing. *Computers in Chemical Engineering* 19(3): 253–266.

Brandin, B. A., and Wonham, W. M. 1992. Supervisory control of timed discrete-event systems. *Proc. 31st IEEE Conf. on Decision and Control*, Tuscon, Arizona, pp. 3357–3362.

Brave, Y., and Heymann, M. 1988. Formulation and control of real-time discrete event processes. *Proc. 27th IEEE Conf. on Decision and Control*, Austin, Texas, pp. 1131–1132.

Brave, Y., and Krogh, B. H. 1993. Maximally permissive policies for controlled time marked graphs. *Proc. 12th IFAC World Congress*, Sydney, Australia, pp. I:263–266.

Bruno, G., and Marchetto, G. 1986. Process-translatable Petri nets for the rapid prototyping of process control systems. *IEEE Trans. on Software Engineering* 12(2): 346–357.

Cofer, D. 1995. *Control and Analysis of Real-Time Discrete Event Systems*. PhD thesis, Dept. of ECE, University of Texas at Austin.

Cofer, D., and Garg, V. K. 1995. Control of event separation times in discrete event systems. *Proceedings of the 34th Conference on Decision and Control*, New Orleans, LA, pp. 2005–2010.

Cofer, D., and Garg, V. K. 1996. Supervisory control of real-time discrete event systems using lattice theory. *IEEE Transactions on Automatic Control* 41(2): 199–109.

Cohen, G., Moller, P., Quadrat, J.-P., and Voit, M. 1989. Algebraic tools for the performance evaluation of discrete event systems. *Proceedings of the IEEE* 77(1): 39–58.

Colom, J. M., and Silva, M. 1991. Improving the linearly based characterization of P/T nets. *Advances in Petri Nets 1990*, (Rozenberg, G. ed.), *Lecture Notes in Computer Science*, vol. 483, New York: Springer Verlag, pp. 113–145.

David, R. 1993. Petri nets & Grafcet for specification of logic controllers. *Proc. 12th IFAC World Congress*, Sydney, Australia, pp. III:335–340.

David, R., and Alla, H. 1992. *Petri Nets and Grafcet*. New York: Prentice Hall.

Desel, J. 1992. A proof of the rank theorem for extended free choice nets. *Application and Theory of Petri Net 1992*, (Jensen, K., ed.), *Lecture Notes in Computer Science*, vol. 616, New York: Springer Verlag, pp. 134–154.

Coffman, E. G., et al. 1971. System deadlocks. *Computing Surveys* 3(2): 67–78.

Ezpeleta, J., Colom, J. M., and Martinez, J. 1995. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation* 11(2): 173–184.

Giua, A., and DiCesare, F. 1991. Supervisory design using Petri nets. *Proc. 30th IEEE Conf. on Decision and Control*, Brighton, UK, pp. 92–97.

Giua, A., and DiCesare, F. 1994a. Blocking and controllability of Petri nets in supervisory control. *IEEE Trans. on Automatic Control* 39(4): 818–823.

Giua, A., and DiCesare, F. 1994b. Petri net structural analysis for supervisory control. *IEEE Trans. on Robotics and Automation* 10(2): 185–195.

Giua, A., and DiCesare, F. 1995. Decidability and closure properties of weak Petri net languages in supervisory control. *IEEE Trans. on Automatic Control* 40(5): 906–910.

Giua, A., DiCesare, F., and Silva, M. 1992. Generalized mutual exclusion constraints on nets with uncontrollable transitions. *Proc. 1992 IEEE Int. Conf. on Systems, Man, and Cybernetics*, Chicago, Illinois, pp. 974–979.

Giua, A., DiCesare, F., and Silva, M. 1993. Petri net supervisors for generalized mutual exclusion constraints. *Proc. 12th IFAC World Congress*, Sidney, Australia, pp. I:267–270.

Golaszewski, C. H., and Ramadge, P. J. 1988a. Discrete event processes wtih arbitrary controls. *Advanced Computing Concepts and Techniques in Control Engineering*, (Denham, M. J., and Laub, A. J., eds.), New York: Springer Verlag, pp. 459–469.

Golassewaski, C. H., and Ramadge, P. J. 1988b. Mutual exclusion problems for discrete event systems with shared events. *Proc. 27th IEEE Conf. on Decision and Control*, Austin, Texas, pp. 234–239.

Hanisch, H.-M., Luder, A., and Rausch, M. 1996. Controller synthesis for net condition/event systems with incomplete state observation. *Proceedings of Computer Integrated Manufacturing and Automation Technology (CIMAT'96) Conference*, Grenoble, France.

Haoxun, C. 1994. Synthesis of feedback control logic for controlled Petri nets with forward and backward conflict-free uncontrolled subnet. In *Proc. 33rd IEEE Trans. on Decision and Control*, Lake Buena Vista, FL, pp. 3098–3103.

Haoxun, C., and Baosheng, H. 1991. Distributed control of discrete event systems described by a class of controlled Petri nets. *Preprints of IFAC Int. Symp. on Distributed Intelligence Systems*, Arlington, Virginia.

Haoxun, C., and Baosheng, H. 1993. Control of discrete event systems with their dynamics and legal behavior specified by Petri nets. *Proc. 32nd IEEE Trans. on Decision and Control*, San Antonio, TX, pp. 239–240.

Holloway, L. E. 1988. Feedback control synthesis for a class of discrete event systems using distributed state models. Laboratory for Automated Systems and Information Processing, Dept. of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. Technical Report LASIP-88-17.

Holloway, L. E. 1996. Time measures and state maintainability for a class of composed systems. *Proceedings of the 1996 International Workshop on Discrete Event Systems (WODES96)*, Edinburgh, UK.

Holloway, L. E., and Krogh, B. H. 1990. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Trans. on Automatic Control* 35(5): 514–523. Also appears in *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*, (Ho., Y. C., ed.), New York: IEEE Press, 1992.

Holloway, L. E., and Krogh, B. H. 1992. On closed-loop liveness of discrete event systems under maximally permissive control. *IEEE Trans. on Automatic Control* 37(5): 692–697.

Holloway, L. E., Guan, X., and Zhang, L. 1996. A generalization of state avoidance policies for controlled Petri nets. *IEEE Trans. on Automatic Control* 41(6): 804–816.

Holloway, L. E., and Hossain, F. 1992. Feedback control for sequencing specifications in controlled Petri nets. *Proc. Third Int. Conf. on Computer Integrated Manufacturing*, Troy, New York, pp. 242–250.

Hsieh, F.-S., and Chang, S.-C. 1994. Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation* 10(2): 196–209.

Ichikawa, A., and Hiraishi, K. 1988. Analysis and control of discrete event systems represented by Petri nets. *Discrete Event Systems: Models and Applications*, (Varaiya, P., and Kurzhanski, A. B., eds.), *Lecture Notes in Control and Information Sciences*, vol. 103, New York: Springer Verlag, pp. 115–134.

Jantzen, M. 1987a. Complexity of place/transition nets. *Petri Nets: Control Models and Their Properties, Advances in Petri Nets 1986*, (Reisig, W., Brauer, W., and Rozenberg, G., eds.), *Lecture Notes in Computer Sciences*, vol. 254-I, New York: Springer Verlag, pp. 413–434.

Jantzen, M. 1987b. Language theory of Petri nets. *Petri Nets: Control Models and Their Properties, Advances in Petri Nets 1986*, (Reisig, W., Brauer, W., and Rosenberg, G., eds.), Lecture Notes in Computer Sciences, Vol. 254-I, Springer Verlag, New York, pp. 397–412.

Jeng. M. D., and DiCesare, F. 1993. A review of synthesis techniques for petri nets with applications to automated manufacturing systems. *IEEE Trans. on Systems, Man, and Cybernetics* 23(1): 301–312.

Jensen, K. 1995. *Coloured Petri Nets*. Vol. 1 & 2, Springer.

Johnson, J. L., and Murara, T. 1985. Structure mmatrices for Petri nets. *Journal of the Franklin Institute* 319(3): 299–309.

Jones, N. D., Landweber, L. H., and Lien, Y. E. 1977. Complexity of some problems in Petri nets. *Theoretical Computer Science* 4: 277–299.

Kosaraju, S. R. 1982. Decidability of reachability in vector addition systems. *Proc. 14th Ann. ACM Symp. on Theory of Computing*, San Francisco, California, pp. 267–281.

Krogh, B. H. 1987. Controlled Petri nets and maximally permissive feedback logic. *Proc. 25th Annual Allerton Conference*, University of Illinois, Urbana, pp. 317–326.

Krogh, B. H., and Holloway, L. E. 1991. Synthesis of feedback control logic for discrete manufacturing systems. *Automatica* 27(4): 641–651.

Krogh, B. H., Magott, J., and Holloway, L. E. 1991. On the complexity of forbidden state problems for controlled marked graphs. *Proc. 30th IEEE Conf. on Decision and Control*, Brighton, UK, pp. 85–91.

Kumar, R., and Holloway, L. D. 1996. Supervisory control of deterministic Petri nets with regular specification languages. *IEEE Trans. on Automatic Control* 41(2): 245–249.

Lafortune, S., and Yoo, H. 1991. Some results on Petri net languages. *IEEE Trans. on Automatic Control* 35(4): 482–485.

Lewis, F. L., Pastravanu, O. C., and Huang, H.-H. 1993. Controller design and conflict resolution in discrete event manufacturing systems. *Proc. 32nd IEEE Conf. on Decision and Control*, IEEE Control Systems Society, pp. 3288–3293.

Li, Y., 1991. *Control of Vector Discrete-Event Systems*. PhD thesis, Systems and Control Group, Department of Electrical Engineering, University of Toronto, Toronto, Ontario.

Li, Y., and Wonham, W. M. 1993. Control of vector discrete-event systesm I—the base model. *IEEE Trans. on Automatic Control* 38(8): 1214–1227.

Li, Y., and Wonham, W. M. 1994. Control of vector discrete-event systesm II—controller synthesis. *IEEE Trans. on Automatic Control* 39(3): 512–531.

Li, Y., and Wonham, W. M. 1995. Concurrent vector discrete-event systems. *IEEE Trans. on Automatic Control* 40(4): 628–638.

Makungu, M., Barbeau, M., and St-Denis, R. 1994. Synthesis of controllers with colored Petri nets. *Proc. 32nd Annual Allerton Conference*, University of Illinois, Urbana, pp. 709–718.

Mayr, E. W. 1984. An algorithm for the general Petri net reachability problem. *SIAM J. of Computing* 13(3): 441–460.

McCarragher, B. J., and Asada, H. 1995. The discrete event modeling and trajectory planning of robitic assembly tasks. *Transactions of the ASME—Journal of Dynamic Systems, Measurement, and Control* 117(3): 394–400.

Memmi, G., and Roucairol, G. 1980. Linear algebra in net theory. *Lecture Notes in Computer Science*, vol. 84, Springer, pp. 213–223.

Moody, J. O., and Antsaklis, P. J. 1995. Petri net supervisors for des in the presence of uncontrollable and unobservable transitions. *Proceedings of 33rd Annual Allerton Conference on Communications, Control, and Computing*, Monticello, IL.

Moody, J. O., Yamalidou, K., Lemmon, M. D., and Antsaklis, P. J. 1994. Feedback control of Petri nets based on place invariants. *Proc. 33rd IEEE Conf. on Decision and Control*, Lake Buena Vista, Florida, pp. 3104–3109.

Moody, J. O., Antsaklis, P. J., and Lemmon, M. D. 1996. Petri net feedback controller design for a manufacturing system. *Proceedings of IFAC World Congress 1996*, vol. B, San Francisco: Pergamon/Elsvier Science Limited, pp. 67–72.

Murata, T. 1977. State equation, controllability and maximal matching of Petri nets. *IEEE Trans. on Automatic Control* AC-22(3): 412–415.

Murata, T. 1989. Petri nets: Propeties, analysis and applications. *Proceedings of the IEEE* 77(4): 541–580.

Ostroff, J. S., and Wonham, W. M. 1990. A framework for real-time discrete event control. *IEEE Trans. on Automatic Control* 35(4): 386–397.

Pelz, E. 1987. Closure properties of deterministic Petri net languages. *Proc. STACS 1987, Lecture Notes in Computer Sciences*, vol. 247, New York: Springer Verlag, pp. 373–382.

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

Ramadge, P. J. 1989. Some tractable supervisory control problems for discrete-event systems modeled by buchi automata. *IEEE Trans. on Automatic Control* 34(1): 10–19.

Ramadge, P. J., and Wonham, W. M. 1987a. Modular feedback logic for discrete-event systems. *SIAM J. of Control and Optimization* 25(5): 1202–1218.

Ramadge, P. J., and Wonham, W. M. 1987b. Supervisory control of a class of discrete-event processes. *SIAM J. of Control and Optimization* 25(1): 206–230.

Ramadge, P. J., and Wonham, W. M. 1989. The control of discrete event systems. *Proceedings of IEEE* 77(1): 81–98.

Reisig, W. 1982. Petri nets. *Monographs on Theoretical Computer Science*. New York: Springer Verlag.

Reutenauer, C. 1990. *The Mathematics of Petri Nets*. Masson and Prentice-Hall.

Sathaye, A. S., and Krogh, B. H. Synthesis of real-time supervisors for controlled time Petri nets. *Proc. 32nd IEEE Conf. on Decision and Control*, vol. 1, San Antonio, pp. 235–238.

Sifakas, J. 1980. Deadlocks and livelocks in transition systems. *Mathematical Foundations of Computer Science* 88: 587–600.

Sifakis, J. 1978. Structural properties of Petri nets. *Mathematical Foundations of Computer Science*, (Winkowski, J., ed.), Springer, pp. 474–483.

Smedinga, R. 1988. Using trace theory to model discrete event systems. *Discrete Event Systems: Models and Applications*, (Varaiya, P., and Kurzhanski, A. B., eds.), *Lecture Notes in Control and Information Sciences*, New York: Springer Verlag, pp. 81–99.

Sreenivas, R. S. 1993a. A note on deciding the controllability of a language K with respect to a language L. *IEEE Trans. on Automatic Control* 38(4): 658–662.

Sreenivas, R. S. 1993b. On a weaker notion of controllability of a language K with respect to language L. *IEEE Trans. on Automatic Control* 38(9): 1446–1447.

Sreenivas, R. S. 1996. Enforcing liveness via supervisors control in discrete event dynamic systems modeled by completely controlled petri nets. *IEEE Transactions on Automatic Control*, submitted.

Sreenivas, R. S., and Krogh, B. H. 1992. On Petri net models of infinite state supervisory. *IEEE Trans. on Automatic Control* 37(2): 274–277.

Stiver, J. A., and Antsaklis, P. J. 1993. On the controllability of hybrid control systems. *Proc. 32nd IEEE Conf. on Decision and Control*, IEEE Control Systems Society, pp. 294–299.

Suziki, I., and Murara, T. 1983. A method for stepwise refinement and abstraction of Petri nets. *Journal of Computer and System Sciences* 27(1): 51–76.

Takae, A., Takai, S., Ushio, T., Kumagai, S., and Kodama, S. 1996. Maximally permissive controllers for controlled time Petri nets. *Proceedings 33rd Conf. on Decision and Control*, Lake Buena Vista, FL, pp. 1058–1059.

Takai, S., Ushio, T., and Kodama, S. 1994. Concurrency and maximally permissive feedback in Petri nets with external input places. *International Journal of Control* 60(4): 617–629.

Ushio, T. 1989. On the controllability of controlled Petri nets. *Control-Theory and Advanced Technology* 5(3): 265–275.

Ushio, T. 1990. Maximally permissive feedback and modular control synthesis in Petri nets with external input places. *IEEE Trans. on Automatic Control* 35(7): 844–848.

Ushio, T., and Matsumoto, R. 1988. State feedback and modular control synthesis in controlled Petri nets. *Proc. 27th IEEE Conf. on Decision and Control*, Austin, Texas, pp. 1502–1507.

Valette, R. 1983. A petri net based programmable logic controller. *Computer Applications in Production and Engineering (CAPE '83)*, (Warman, E. A., ed.), North-Holland Publishing, pp. 103–116.

Viswanadham, N., Narahari, Y., and Johnson, T. L. 1990. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using petri net models. *IEEE Trans. on Robotics and Automation* 6(6): 713–722.

Wang, F. Y. 1992. Supervisory control for concurrent discrete event dynamic systems based on Petri nets. *Proc. 31st IEEE Conf. on Decision and Control*, Tuscon, Arizona, pp. 1196–1197.

Wang, H. 1993. Trace model for concurrent DEDS. *Proc. 12th IFAC World Congress*, Sidney, Australia, pp. V:1–4.

Watson, J. F., and Derochers, A. A. 1994. State-spae size estimation of Petri nets: a bottom-up perspective. *IEEE Trans. on Robotics and Automation* 10(4): 555–561.

Wong-Toi, H., and Hoffmann, G. 1991. The control of dense real-time discrete event systems. *Proc. 30th IEEE Conf. on Decision and Control*, Brighton, UK, pp. 1527–1528.

Wonham, W. M, and Ramadge, P. J. 1987. On the supremal controllable sublanguage of a given language. *SIAM J. on Control and Optimization* 25(3): 637–659.

Xing, K.-Y., Hu, B.-S., and Chen, H.-X. 1996. Deadlock avoidance policy for Petri-net modeling of flexible manufacturing systems wih shared resources. *IEEE Transactions on Automatic Control* 41(2): 289–295.

Xing, K.-Y., Li, J. M., Hu, B. S. 1996. A new method for synthexizing state avoidance policies for controlled petri nets. *Proceedings of IFAC World Congress 1996*, vol. J, San Francisco: Pergamon/Elsevier Science Limited,

pp. 377–382.

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

Zhou, M. C., and DiCesare, F. 1993. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer.

Zhou, M. C., DiCesare, F., and Desrochers, A. 1992. Hybrid methodology for the synthesis of Petri nets models for manufacturing systems. *IEEE Trans. on Robotics and Automation* 8(3): 350–361.

Zhou, M. C., DiCesare, F., and Rudolph, D. L. 1992. Design and implementation of a petri net based supervisor for a flexible manufacturing system. *Automatica* 28(6): 1199–1208.

Zurawski, R., and Zhou, M. 1994. Petri nets and industrial applications: a tutorial. *IEEE Trans. on Industrial Electronics* 41(6): 576–583.