Petri Net Techniques for Supervisory Control of Discrete Event Systems

Alessandro Giua,

Dip. di Ingegneria Elettrica ed Elettronica, Università di Cagliari,

Piazza d'Armi, 09123 Cagliari, Italy

Phone: +39-070-675-5892 – Fax: +39-070-675-5900 – Email: giua@diee.unica.it

Abstract

In this tutorial paper, several issues related to the use of Petri nets in the supervisory control of discrete event systems are discussed. The basic elements of supervisory control are presented using Petri nets as discrete event models. The language properties of Petri nets are studied in this framework to derive important results on the classes of control problems that can be effectively solved. Examples of the use of Petri net structural techniques are given, discussing the design of control structures for enforcing linear constraints on the reachability set of a net.

Published as:
A. Giua, "Petri Net Techniques for Supervisory Control of Discrete Event Systems," *Proc. First Int. Work. on Manufacturing and Petri Nets* (Osaka, Japan), pp. 1–30, June, 1996.

A longer version of this paper also appeared in the proceedings of the summer school *MOVEP96: Modélisation et verification des Processus Parallèles* (Nantes, France), June 1996.

1 Introduction

The application of discrete event systems (DES) control theory in industrial settings is a challenging problem.

Firstly, within the field of process automation, discrete event control systems are used not only in the area of discrete processes, where they find their natural application, but also in the area of batch and continuous processes. Secondly, logic control design is characterized by an intrinsic interdisciplinary feature that requires particular knowledge in a variety of sectors besides that of the control process: for example, those concerning communication protocols, operating systems, real time systems programming, microprocessors, computer integrated manufacturing. Finally, the types of problems that must be solved are very different, including alarm and protection management, local control of workstations, supervision and coordination, monitoring and diagnostic, quality control.

For these reasons, the design is often made empirically, according to trial and error methods, based on the control-expert's ingenuity and expertise.

To avoid this situation, it is necessary to identify a systematic approach to the design problem. The research community has been very productive in this field in the last years. Of particular interest are those approaches based on *system theory*, in which there is a clear distinction between the process or plant (e.g., a workcell, in a manufacturing system) and the controller, which is a decision-making unit. Both models are given an input/output description, and it is extremely useful to model the exchange of information between these models.

Supervisory control, originated by the work of Ramadge and Wonham [32, 41, 31], is a system theory approach that has been gaining increasing importance because it provides a unifying framework for the control of DES's. Supervisory control is a mathematical theory, based on formal languages, that allows the designer to model specifications and to solve the given control problem with standard algorithms. The basic elements of supervisory control are discussed in a section of this paper.

In the original work of Ramadge and Wonham finite state machines (FSM) were used model the plant and the specifications. FSM's 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, and because of the lack of structure.

More efficient models have been proposed in the DES literature. Here the attention will be drawn to Petri net (PN) models.

PN's have several advantages over FSM's. Firstly, PN's have a higher language complexity than FSM, since Petri net languages are a proper superset of regular languages. Secondly, the states of a PN are represented by the possible markings and not by the places: thus they give a compact description, i.e., the structure of the net may be maintained small in size even if the number of the markings grows. Thirdly, PN's can be used in modular synthesis, i.e., the net can be considered as composed of interrelated subnets, in the same way as a complex system can be regarded as composed of interacting subsystems.

Although PN's have a greater modeling power than FSM's, computability theory shows that the increase of modeling power often leads to an increase in the computation required to solve problems. As an example, all properties of interest of FMS's 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. This is why a section of this paper focuses on the decidability properties of Petri nets by studying the corresponding languages. It will be shown that Petri nets represent a good trade-off between modeling power and analysis capabilities

Petri nets offer a structured model of DES dynamics that can be exploited in developing more efficient algorithms for controller synthesis. The last section of the paper discusses a class of constraints, that can be seen as a generalization of *mutual exclusion* (or *mutex*) constraints. Several PN structural techniques [4, 34], have been used by different authors to find efficient solutions to the problem of enforcing these constraints. These approaches will be reviewed and compared.

The paper is structured as follows. In Section 2 is introduced the notation on Petri nets and formal languages. In Section 3 the basic elements of supervisory control are presented. In Section 4 the theoretical issues related to the use of PN's in supervisory control are discussed. In Section 5 *generalized mutual exclusion constraints* are studied to provide an example of the advantages offered by PN's in the design of efficient control structures.

2 Petri Nets and Formal Languages

2.1 Net systems

A *Place/Transition net* (P/T net) [28] is a structure $N = (P, T, Pre, Post)$, where P is a set of *places*; T is a set of *transitions*; $Pre : P \times T \to \mathbb{N}$ is the *pre-incidence function*; $Post : P \times T \to \mathbb{N}$ is the *post-incidence function*.

If the net is *pure* (i.e., it has no selfloops) the incidence functions can be represented by a single matrix, the *incidence matrix* of the net, defined as $C(p,t) = Post(p,t) - Pre(p,t)$.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place of a P/T net a non-negative integer number of tokens, represented by black dots. A *P/T system* or *net system* $\langle N, M_0 \rangle$ is a net N with an initial marking M_0.

A transition $t \in T$ is *enabled* at a marking M iff $M \geq Pre(\cdot, t)$. If t is enabled at M, then t may fire yielding a new marking M' with $M' = M + C(\cdot, t)$. One writes $M\ [t\rangle\ M'$ to denote that t may fire at M yielding M'.

A *firing sequence* from M_0 is a (possibly empty) sequence of transitions $\sigma = t_1 \ldots t_k$ such that $M_0\ [t_1\rangle\ M_1\ [t_2\rangle\ M_2 \cdots\ [t_k\rangle\ M_k$. A marking M is *reachable* in $\langle N, M_0 \rangle$ iff there exists a firing sequence σ such that $M_0\ [\sigma\rangle\ M$.

If marking M is reachable in $\langle N, M_0 \rangle$ by firing a sequence σ, then the following *state equation* is satisfied: $M = M_0 + C \cdot \vec{\sigma}$, where $\vec{\sigma} : T \to \mathbb{N}$ is a vector of non-negative integers, called the *firing count vector*. $\vec{\sigma}(t)$ represents the number of times transition t appears in σ. The set of nonnegative integer vectors M such that there exists a vector $\vec{\sigma}$ satisfying the previous state equation is called *potentially reachable set* and is denoted $PR(N, M_0)$. Note that $PR(N, M_0) \supseteq R(N, M_0)$.

Let $Y : P \to \mathbb{Z}$ be a vector and $P' \subseteq P$. The *support* of Y is $Q_Y = \{p \in P \mid Y(p) \neq 0\}$. The *projection* of Y on P' is the restriction of Y to P' and will be denoted $Y \uparrow_{P'}$. This definition is extended in the usual way to the projection of a set of vectors $\mathcal{Y}$, i.e., $\mathcal{Y} \uparrow_{P'} = \{Y \uparrow_{P'} \mid Y \in \mathcal{Y}\}$.

A net system $\langle N, M_0 \rangle$ is: *Bounded*, if there exists a nonnegative integer k such that $M(p) \leq k$ for every place p and for every marking $M \in R(N, M_0)$; *Safe* (or *1-bounded*), if $M(p) \leq 1$ for every place p and for every marking $M \in R(N, M_0)$.

2.2 Petri net languages

A *labeled Petri net* (or *Petri net generator*) [17, 30] is a 4-tuple $G = (N, \ell, M_0, F)$ where: $N = (P, T, Pre, Post)$ is a P/T structure; $\ell : T \to \Sigma$ is a labeling function that assigns to each transition a label from the alphabet of events Σ and will be extended to a mapping $T^* \to \Sigma^*$ in the usual way; M_0 is an initial marking; F is a finite set of final markings.

A discrete event system will be represented as a labeled Petri net. Given a DES $G = (N, \ell, M_0, F)$: the *L-type language* of G is $L_\ell(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_0\ [\sigma\rangle\ M, M \in F\}$; the *G-type language* of G (also called *weak language*) is $L_g(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_0\ [\sigma\rangle\ M, (\exists M' \in F) M \geq M'\}$; the *P-type language* of G (also called *prefix language*)

is $L(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_0\,[\sigma\rangle\}$.
Example 1. Consider the labeled net G in Figure 4.(c) with set of final markings $F = \{(0\ 0\ 0\ 1)^T\}$. The languages of this net are: $L_\ell(G) = \{a^m b a^m b \mid m \geq 0\}$, $L_g(G) = \{a^m b a^n b \mid m \geq n \geq 0\}$, $L(G) = \{a^m \mid m \geq 0\} \cup \{a^m b a^n \mid m \geq n \geq 0\} \cup \{a^m b a^n b \mid m \geq n \geq 0\}$. $\diamond$

Note that in the definition of labeled net ℓ is a λ-free labeling function, according to the terminology of Peterson [30], i.e., no transition is labeled with the empty word λ and two (or more) transitions may have the same label. The classes of L-type, G-type, and P-type languages generated by λ-free labeled nets are denoted $\mathcal{L}$, $\mathcal{G}$, and $\mathcal{P}$ respectively. It is known [17] that $\mathcal{P} \subsetneq \mathcal{G} \subsetneq \mathcal{L}$.

A *deterministic* PN generator [17] is such that the word of events generated from the initial marking uniquely determines the marking reached. Formally, a DES G is deterministic iff for all $t, t' \in T$, with $t \neq t'$, and for all $M \in R(N, M_0)$: $M\ [t\rangle \wedge M\ [t'\rangle \implies [\ell(t) \neq \ell(t')] \vee [Post(\cdot,t) - Pre(\cdot,t) = Post(\cdot,t') - Pre(\cdot,t')]$. Note that we are slightly extending the definition of determinism introduced in [40] and used by [17, 30, 35]. In fact, we accept as deterministic a system in which two transitions with the same label may be simultaneously enabled at a marking M, provided that the two markings reached from M by firing t and t' are the same (it is a kind of parallelism of transitions).

Systems of interest in supervisory control theory are deterministic, hence we will always assume that the generators considered in this paper are deterministic. The classes of L-type, G-type, and P-type PN languages generated by deterministic PN generators are denoted $\mathcal{L}_d$, $\mathcal{G}_d$, and $\mathcal{P}_d$. Each class of deterministic language here defined is strictly included in the corresponding class of λ-free languages. There is however a clear advantage in restricting the model to deterministic PN generators: properties of interest are decidable for deterministic net while they are not for λ-free nets. This will be discussed in Section 4.3.

It is known that $\mathcal{P}_d \subset \mathcal{G}_d$ [17]. However, in [11, 6] is was shown that the classes $\mathcal{G}_d$ and $\mathcal{L}_d$ are incomparable, and furthermore $\mathcal{G}_d \cap \mathcal{L}_d = \mathcal{R}$, where $\mathcal{R}$ is the class of regular languages. Hence in the deterministic setting, taking also into account the G-type language of deterministic nets (in addition to the L-type language) one extends the class of control problems that can be modeled by PN's.

3 Introduction to Supervisory Control Theory

3.1 Introduction

Supervisory control is a theory for discrete event systems originated by the work of Ramadge and Wonham [32, 41, 31].

Three different areas of interest in the supervisory control approach may be recognized.

The **formal language level** is concerned with theoretical issues. Qualitative properties of DES's, such as *stability*, *controllability*, *observability*, *nonblockingness*, etc., are defined from a very general perspective as properties of languages. This "abstract" perspective has made possible the creation of a model-independent theory.

The **system level** deals with the concept of DES, seen as a language generator over an alphabet of events Σ. Design algorithms have been devised to compute a controller (called supervisor) for a given control problem, given the knowledge of the system and of the specification. These algorithms are based on language operators, thus they are model independent.

At the **model level** a particular model is chosen to describe the physical device to be controlled. In classical control theory there exist different models for, say, linear systems, such as transfer function, state variables, matrix-fraction description, polynomial matrix description, etc., all capable of describing systems whose behavior is defined by a differential linear equation. In the case of a DES, the behavior is defined as a formal language; hence the models used are language generators such as finite state machines, Petri nets, communicating sequential processes and related formalism, etc.

In this section some basic elements of the theory are presented. For a more detailed survey the reader is referred to [33, 20].

3.2 Discrete event systems and properties

Let L be a language on alphabet Σ. Its *prefix closure* is the set of all prefixes of words in L: $\overline{L} = \{\sigma \in \Sigma^* \mid \exists \tau \in \Sigma^* \ni \sigma\tau \in L\}$. A language L is said to be *closed* if $L = \overline{L}$.

In the supervisory control theory a DES is simply a generator of a formal language, defined on an alphabet Σ. The two languages associated with a DES G are: the *closed behavior* $L(G) \subseteq \Sigma^*$, a prefix-closed language that represents the possible evolutions of the system; the *marked behavior* $L_m(G) \subseteq L(G)$, that represents the evolutions corresponding to the

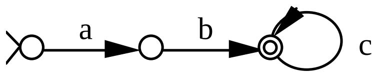

Figure 1: A discrete event system modeled by a finite state machine.

completion of certain tasks.

When finite state machines are used as language generators, the closed behavior is given by the words that can be generated starting from the initial state and reaching any other state, while the marked behavior is given by the words that can be generated reaching a final state.

Example 2. The FMS in Figure 1 represents a discrete event system G. The initial state is denoted by an arrow, the final states (just one in this example) are denoted by a double circle. The languages associate to G are: $L_m(G) = \{abc^n \mid n \geq 0\}$ and $L(G) = \{\lambda\} \cup \{a\} \cup \{abc^n \mid n \geq 0\} = \overline{L_m(G)}$, where the overline bar represent the prefix closure operator. $\diamond$

Although Ramadge and Wonham have used in their seminal papers a FSM based representation for DES's, the theory they built is very general. Any other formalism that can represent the closed and marked behavior of a system may be used in this framework. In the examples presented in this paper PN models are mostly used, while FSM's are only used in a few examples in this section.

When PN's are used as discrete event models, the closed behavior is given by the P-language. The marked behavior can be represent by any of the terminal languages discussed in Section 2.2, i.e., by the L-language, or by the G-language. In Section 4.4 other possible choices for the set of final markings — that will lead to the definition of other marked behaviors — are mentioned.

A common assumption is that of considering *deterministic systems*, i.e., deterministic language generators. For this class of systems the knowledge of the word executed by the system is sufficient to reconstruct the marking reached by the system starting from the (unique) initial marking. In Section 4.3 it will be shown that for deterministic PN generators the properties of interest in supervisory control are decidable.

A DES is said to be *nonblocking* if any word $w \in L(G)$ can be completed into a word $wx \in L_m(G)$, i.e., into a word that belongs to the marked language. A deterministic DES G is nonblocking iff $\overline{L_m(G)} = L(G)$.

This property may be restated for FSM's as follows. Let us define as *reachable* a state that may be reached from the initial state, and as *coreachable* a state from which it is possible to reach a final state. Then a FSM is nonblocking if and only if all reachable states are coreachable. A FSM that is reachable and coreachable is said to be *trim*. Note that blocking FSM's may be trimmed removing the states that are unreachable or uncoreachable and the associated transitions.

Similarly, a PN is nonblocking if from any reachable marking it is possible to reach a final marking. Note that this property is significantly different from *deadlock-freeness*. Deadlock-freeness means that the reachability set does not contain a dead marking, i.e., a marking that enables no transition. However a nonblocking system may have a dead marking (if it is a final marking)

The trimming of PN's is not as simple as the trimming of FMS's. See Section 4.1 for a discussion of this problem.

3.3 Controllable events and supervisor

A fundamental concept in supervisory control is that of uncontrollable event. The events labels in Σ are partitioned into two disjoint subsets: the set Σ_c of *controllable events* (that can be disabled if desired), and the set Σ_u of *uncontrollable events* (that cannot be disabled by an external agent).

The control specification in supervisory control is given as a *specification language*, i.e., as a legal behavior. The aim of the control is that of restricting the behavior of a system within the limits of the legal behavior. We may only enable and disable the controllable events to achieve this behavior. The agent that specifies which events are to be enabled and disabled is called a *supervisor*.

Let us 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). If $a \in \gamma$, the event a is enabled by γ (permitted to occur), otherwise a is disabled by γ (prohibited from occurring); the uncontrollable events are always enabled. Let $\Gamma \subseteq 2^{\Sigma}$ denote the set of all the possible control inputs. A supervisor controls a DES G by switching the control input through a sequence of elements $\gamma_1, \gamma_2, \ldots \in \Gamma$, in response to the observed word of previously generated events.
Example 3. Consider the system in Figure 2.(a), where $\Sigma_c = \{b, c\}$, $\Sigma_u = \{a\}$, and $F = \{(0\ 0\ 1)^T\}$. Assume one wants the system to reach the final marking executing b exactly twice (i.e., one wants a terminal word to contain the event b exactly twice). The table summarizes the behavior of a supervisor that solves this problem. In the table for each word w that can be legally generated by the system is listed the corresponding marking reached by the system and the required control input γ. ◇

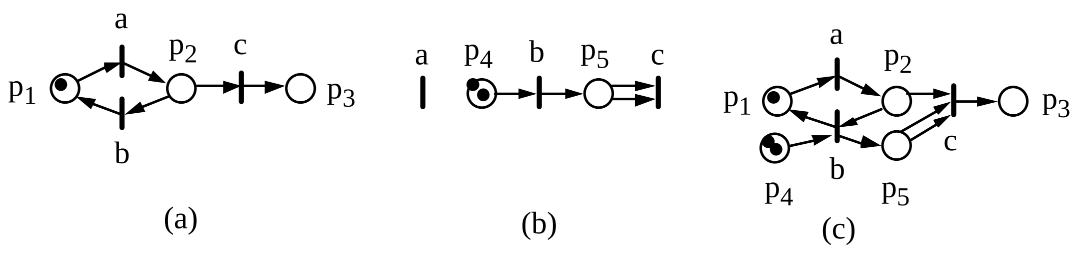

Figure 2: A DES (a), its net supervisor (b), and the closed loop system (c).

marking	w	γ
$(1\ 0\ 0)^T$	λ	$\{a, b\}$
$(0\ 1\ 0)^T$	a	$\{a, b\}$
$(1\ 0\ 0)^T$	ab	$\{a, b\}$
$(0\ 1\ 0)^T$	aba	$\{a, b\}$
$(1\ 0\ 0)^T$	$abab$	$\{a, c\}$
$(0\ 1\ 0)^T$	$ababa$	$\{a, c\}$
$(0\ 0\ 1)^T$	$ababac$	$\{a\}$

3.3.1 State feedback vs. trace feedback

A point that should be stressed is the fact that the supervisor implements a *trace feedback*, rather than a *state feedback*. That is, the control input is not a function of the present state of the system, but of the word of events generated. If the system is deterministic, as assumed, it is possible to reconstruct its state from the word of events generated. Thus trace feedback is more general than state feedback.

The supervisor given in the table in Section 3.3.1 implements a trace feedback and not a state feedback because when the system is in the marking $(0\ 1\ 0)^T$ the control input may be $\{a, b\}$ or $\{a, c\}$ depending on the word previously generated.

There are however interesting classes of supervisory control problems that can be solved by simple state feedback such as a generalized mutual exclusion constraints that will be discussed in Section 5.

The notion of state vs. trace feedback is the discrete event counterpart of the notion of constant vs. time-varying gain feedback for continuous time systems.

3.3.2 Compiled vs. mapping supervisors

A supervisor has been defined as a mapping $f : L(G) \to \Gamma$ specifying the control input $f(w)$ to be applied for each possible word of events w generated by the system G.

It may possible to represent a supervisor as another DES S, that runs in parallel with the system G. Whenever an event occurs in G the same event will be executed by S. The events enabled at a given instant on S determine the control input. Thus one may distinguish between *mapping* supervisors, whose control law is a function computed after each new event generated by the system, and *compiled* supervisors, whose control law is represented as a DES structure.

The supervisor described in tabular format in Section 3.3.1 can be represented with the net structure shown in Figure 2.(b). Here, the transition labeled "a" has no input places, hence it is always enabled. The transition labeled "b" can fire twice at most. The transition labeled "c" can fire once but only after the transition labeled "b" has fired twice.

When the supervisor is a net, it is easy to construct a new net (called S/G) representing the closed loop model of the system under supervision. As an example, Figure 2.(c) shows the net S/G, for the system G in Figure 2.(a) and the supervisor S in Figure 2.(b).

There are several advantages in fully compiling the supervisor action into a net structure. Firstly, the computation of the control action is faster, since it does not require separate on-line computation. Secondly, the same PN execution algorithms may be used for both the original system and the supervisor. Finally, a closed-loop model of the system under control can be built with standard net composition constructions, as was shown in Figure 2.(c). See also [9].

It is important to note that a compiled supervisor may not exist.
Example 4. Consider again the system in Figure 2.(a) with $F = \{(0\ 0\ 1)^T\}$. A possible specification may require that in a terminal word "$(ab)^i ac$" i be a prime number. Clearly such a supervisor cannot be represented as a PN since the language $L = \{(ab)^i ac \mid i \text{ is prime}\}$ is not a PN language. ◇

Theorem 12 in the following section, gives some sufficient conditions for the existence of FMS supervisors. The existence of PN supervisors will be discussed in Section 4.3.

3.3.3 Uncontrollability increases the complexity

It should be remarked that the existence of uncontrollable transitions in a DES often greatly increases the complexity involved in the solution of a control problem.

In Section 5 generalized mutual exclusion constraints for PN's are discussed. As will be seen, each constraint of this kind can be implemented on nets without uncontrollable transitions by a single place called monitor.

However, it will proved that there are cases in which the presence of uncontrollable transitions makes a monitor-based solution unpractical because it requires a number of monitors that grows exponentially with the number of constrained places. Furthermore, for some net it may happen that a monitor-based solution does not exist if some of the transitions are uncontrollable.

3.4 Closed loop system

3.4.1 Closed loop behaviors

The closed loop system under control will be denoted S/G and may be considered as a new discrete event system [32]. Let us assume in the following that the supervisor is given as a DES S. The closed behavior of S/G is $L(S/G) = L(G) \cap L(S)$, i.e., the subset of the uncontrolled behavior that survives under supervision.

The marked behavior of S/G is $L_m(S/G) = L_m(G) \cap L(S/G)$, i.e., the subset of the uncontrolled marked behavior that survives under supervision.

For Petri net models, the net counterpart of the intersection operator is described in [30]. The net S/G will have set of places given by the union of the set of places of S and G, while the transitions with the same label of each net will be merged as was shown in Figure 2.(c).

Any marking $M_{S/G}$ of S/G can be written $M_{S/G} = \begin{bmatrix} M_G \\ M_S \end{bmatrix}$ where M_G and M_S are markings of G and S. Thus one can also write: $L(S/G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_{S/G,0}\ [\sigma\rangle\}$.

If $L_m(G) = L_\ell(G)$ and F is the set of final markings of G then $L_m(S/G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_{S/G,0}\ [\sigma\rangle M_{S/G}, M_G \in F\}$.

If $L_m(G) = L_g(G)$ and F is the set of final markings of G then $L_m(S/G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_{S/G,0}\ [\sigma\rangle M_{S/G}, (\exists M'_G \in F) M_G \geq M'_G\}$.

There is a slight difference between L-type and G-type languages in the definition of the closed-loop marked behavior.

When $L_m(G) = L_\ell(G)$, the language $L_m(S/G)$ may not be a deterministic L-type language, i.e., it may not exist a deterministic Petri net whose L-type language is $L_m(S/G)$.
Example 5. Let S and G be the deterministic PN generators of the following languages: $L(S) = \{a^m b^n \mid m \geq n \geq 0\}$, and $L(G) = L_m(G) = L_\ell(G) = \{a^* b^*\}$. Hence, the marked behavior of the closed-loop system is $L_m(S/G) = L_m(G) \cap L(S) = L(S) \notin \mathcal{L}_d$ as shown in [11]. $\diamond$

When deterministic G-type PN languages are used as marked behaviors, this problem disappears. In fact, if $L_m(G) = L_g(G)$ the marked behavior of the closed-loop system is $L_m(S/G) = L_g(G) \cap L(S) \in \mathcal{G}_d$, since $L(S) \in \mathcal{P}_d \subset \mathcal{G}_d$ and $\mathcal{G}_d$ is closed under intersection. Thus, there exists a deterministic Petri net whose G-type language is $L_m(S/G)$.

3.4.2 Properties of a supervisor

Now let us consider two given DES's G and S. The DES S qualifies as a proper supervisor for G if the two following properties are ensured.

Controllability. S is Σ_u-enabling, i.e., it does not disable any uncontrollable event that may occur in G. This means that $(\forall w \in L(S/G), \forall a \in \Sigma_u)\ wa \in L(G) \Longrightarrow wa \in L(S/G)$.

Nonblockingness. S is proper, i.e., the behavior of the system under supervision must be nonblocking. This means that $L(S/G) = \overline{L_m(S/G)}$.
Example 6. Let G, S_1, and S_2 be the DES's in Figure 3.(a). Here $\Sigma_c = \{b, c\}$ and the set of final markings of G is $F = \{(0\ 0\ 1)^T\}$. S_1 is not a supervisor for G because in the initial marking it disables the uncontrollable event a that is enabled in the initial marking of G. S_2 is not a proper supervisor for G because if the event a is executed it prevents the system from reaching the final marking (in fact the event b is never enabled). $\diamond$

3.5 Supervisory control problem

Controlling a DES consists in restricting its open loop behavior within a given *specification* language. Thus one may state the two following Supervisory Control Problems.

SCP1 Given a DES G and a specification language $L \subseteq L(G)$, there exists a supervisor S such that $L(S/G) = L$?

SCP2 Given a DES G and a specification language $L \subseteq L_m(G)$, there exists a nonblock-

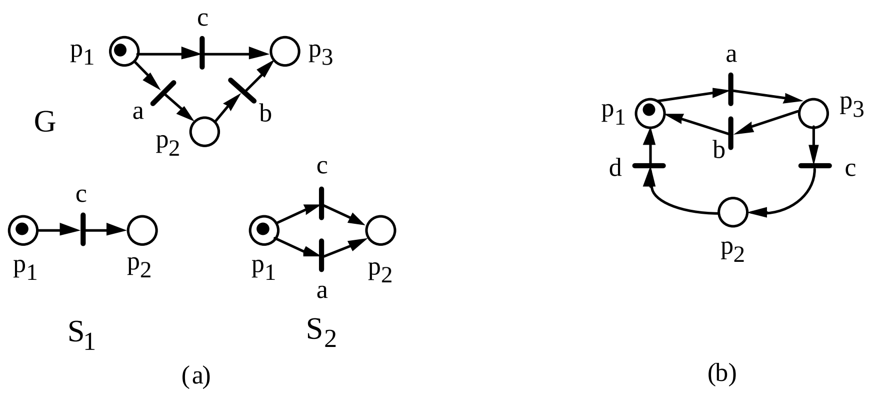

Figure 3: Systems G, S_1, and S_2 in Example 6 (a). System to control in Example 11 (b).

ing supervisor S such that $L_m(S/G) = L$?

The solution of these two problems is based on the notions of controllable language and $L_m(G)$-closure, defined in [32, 33].

Definition 7. *A language $K \subset \Sigma^*$ is said to be* controllable *(with respect to $L(G)$ and Σ_u) if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$.*

This means that it is always possible, disabling only controllable transitions, to restrict $L(G)$ within K.

Definition 8. *If $K, L \subset \Sigma^*$ are languages, K is said to be* L-closed *if $K \cap L = \overline{K} \cap L$.*

This means that any word of L that is the prefix of some word of K is also a word of K.

The following two theorems, due to Ramadge and Wonham [32, 33], give necessary and sufficient conditions for the existence of a supervisor.

Theorem 9. *For nonempty $L \subseteq L(G)$ there exists a supervisor S such that $L(S/G) = L$ if and only if L is prefix closed and controllable.*

Theorem 10. *For nonempty $L \subseteq L_m(G)$ there exists a nonblocking supervisor S such that $L_m(S/G) = L$ if and only if L is $L_m(G)$-closed and controllable.*

Example 11. The DES G in Figure 3.(b) with set of final markings $F = \{(1\ 0\ 0)^T\}$, has closed behavior $L(G) = \overline{(a(b+cd))^*}$, and marked (L-type) behavior $L_m(G) = L_\ell(G) = (a(b+cd))^*$.

(a) Let $L = \overline{(ab)^*}$ be the desired closed behavior of the closed loop system. i) If $\Sigma_u = \{a, d\}$, L is controllable and may be enforced by a supervisor. The supervisor simply needs to disable the controllable event c whenever the system is in marking $(0\ 0\ 1)^T$. ii) If $\Sigma_u = \{b, c\}$, L is not controllable and cannot be enforced by a supervisor. In fact the supervisor cannot disable the event c, which is now uncontrollable, when the system is in marking $(0\ 0\ 1)^T$.

(b) Let $L = (abab)^*$ be the desired controlled behavior of the closed loop system and assume $\Sigma_u = \{a, d\}$. L is controllable, but is not $L_m(G)$-closed. Hence there does not exit a supervisor S such that $L_m(S/G) = L$. This can be also be shown by contradiction in this particular example. By definition of controlled language, $L_m(S/G) = L_m(G) \cap L(S/G)$. Now assume $L_m(S/G) = L$; then $w = abab \in L_m(S/G) \subseteq L(S/G)$ and $w' = ab \notin L_m(S/G)$. However $w \in L(S/G) \implies w' \in L(S/G)$, i.e., $w' \in L_m(G) \cap L(S/G) = L_m(S/G)$, a contradiction. ◇

Another important result due to Ramadge and Wonham [41] is concerned with the existence of FMS supervisors.

Theorem 12. *Let G be a FSM (i.e., its closed and marked behavior are* regular *languages) and let L be a* regular *specification language. If L satisfies the conditions of Theorem 9, the supervisor S such that $L(S/G) = L$ may be represented as a FMS. If L satisfies the conditions of Theorem 10, the supervisor S' such that $L_m(S'/G) = L$ may be represented as a FMS.*

3.6 Supremal controllable sublanguage

Assume one wants to restrict the closed behavior of a system G within the limits of a legal language L that is not controllable. By Theorem 9 a supervisor S such that $L(S/G) = L$ does not exist. However one may consider a controllable sublanguage $K \subseteq L$ that may be enforced by a supervisor S'. Thus the behavior of the closed loop system is now $L(S'/G) = K \subset L$, i.e., it is a subset of the legal behavior.

It is also necessary to minimally restrict the behavior of the system, i.e., one wants to construct the supervisor which allows the largest behavior of the system within the limits given by the specification L. This behavior is called the *supremal controllable sublanguage*.

Let us define $\mathcal{C}(G) = \{K \mid \overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}\}$ as the set of all languages controllable with respect to $L(G)$ and Σ_u. Given a language L the supremal controllable sublanguage of L [32] is defined as $L^\uparrow = \sup\{K \mid K \subseteq L, K \in \mathcal{C}(G)\}$.

An extremely general result is the following [32].

Theorem 13. *The supremal controllable sublanguage for a given supervisory control problem exists and is unique.*

However it may well be the case that $L^\uparrow$ contains only the empty word or is even the empty language.

Example 14. In Example 11.(a)ii $L^\uparrow = \{\lambda\}$. ◇

It may be interesting to observe [18] that the uniqueness ensured by Theorem 13 does not correspond to a unique maximally permissible control if we consider *strictly concurrent systems*, i.e., systems where two or more events may occur simultaneously, as shown in Example 36 in Section 5.3.1.

A second point to be stressed is related to the complexity of solving a given control problem. Theorems 9, 13 are extremely general and seem to imply that a solution always exists. However, it may well be the case that: (a) it is not decidable whether a given language is controllable or $L_m(G)$-closed; (b) there is no algorithm to compute the supremal controllable sublanguage; (c) the supervisor cannot be implemented using the same discrete event model used to represent the system and the specification.

In the FSM case, all properties are trivially decidable by state space search and furthermore Ramadge and Wonham have shown that in the regular case (i.e., when both the system behavior and the specification behavior are regular languages) the supremal controllable sublanguage is a regular language and can be computed in a finite number of steps. In the next section these issues will be discussed in the PN framework.

4 Petri Net Languages for Supervisory Control

In this section the theoretical issues related to the use of PN's in supervisory control are treated by stydying the corresponding languages.

4.1 Petri nets and blocking

A first issue when using Petri nets as discrete events models for supervisory control regards the trimming of blocking nets. The problem is the following: given a PN generator G with languages $L_m(G)$ and $L(G) \supset \overline{L_m(G)}$ one wants to modify the structure of the net to obtain a new DES G' such that $L_m(G') = L_m(G)$ and $L(G') = \overline{L_m(G')} = \overline{L_m(G)}$.

On a simple model such as a state machine this may be done, trivially, by removing all states that are reachable but not coreachable (i.e., no final state may be reached from them) and all their input and output edges.

On Petri net models the trimming may be more complex. If the Petri net is bounded, it was shown in [9] how the trimming may be done without major changes of the net structure, in the sense that one has to add new arcs and eventually duplicate transitions without introducing new places.

Unbounded Petri net models may require more extensive changes of the net structure, as

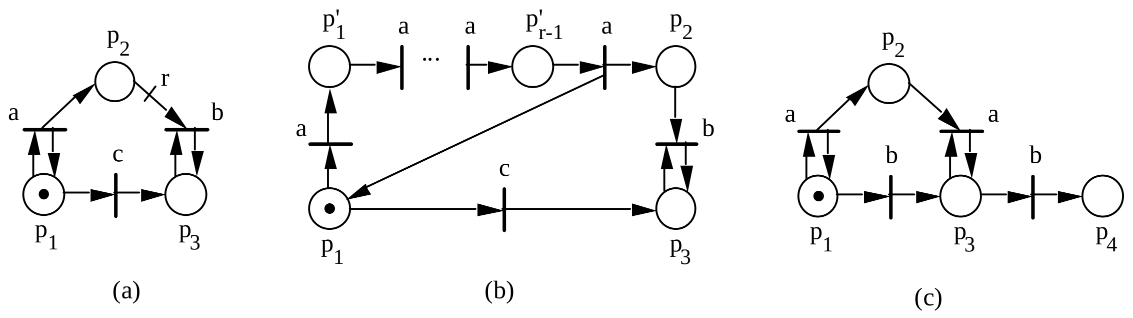

Figure 4: Blocking net in Example 15 (a) and its trimmed structure (b). Blocking net in Example 17 (c).

the following example shows.

Example 15. Let G be the PN generator in Figure 4.(a), with $M_0 = (100)^T$ and set of final markings $F = \{(001)^T\}$. The marked (L-type) and closed behaviors of this net are: $L_m(G) = L_\ell(G) = \{a^m cb^n \mid m = rn, n \geq 0\}$ and $L(G) = \{\overline{a^m cb^n} \mid m \geq rn, n \geq 0\}$. The net is clearly blocking, since the word, say, $\omega = a^{r+1}b \in L(G)$ cannot be completed to a word in $L_m(G)$ (assuming $r > 1$). To avoid reaching a blocking state one requires that p_2 contain a multiple of r tokens when the transition labeled c is allowed to fire. A possible solution is shown in Figure 4.(b), where $r-1$ new places and new transitions labeled a have been introduced. $\diamond$

When the marked language of a net is its L-type Petri net language, it may happen that the trimming of the net is not possible. The reason for this is given by the following theorem.

Theorem 16. There exist L-type Petri net languages whose prefix closure is not a P-type Petri net language, i.e., $\exists L \in \mathcal{L} \ni \overline{L} \notin \mathcal{P}$.

The proof of this theorem will be given by means of the following example.

Example 17. Let G be the deterministic PN generator in Figure 4.(c), with $M_0 = (1000)^T$ and set of final markings $F = \{(0001)^T\}$. The marked (L-type) and closed behaviors of this net are: $L_m(G) = \{a^m ba^m b \mid m \geq 0\}$ and $L(G) = \{\overline{a^m ba^n b} \mid m \geq n \geq 0\}$. To avoid reaching a blocking state one requires that p_2 be empty before firing the transition inputting into p_4. However, since p_2 is unbounded this may not be done with a simple place/transition structure. It is possible to introduce an inhibitor arc from p_2 to the transition inputting into p_4. Inhibitor arcs increase the modeling power, and the analysis complexity, of Petri nets to that of a Turing machine [39], thus these models cannot be properly considered as P/T nets. $\diamond$

It is possible to prove formally that the prefix closure of the marked language of the net discussed in Example 17 is not a P-type Petri net language. The proof is based on the pumping lemma for P-type PN languages, given in [17].

Lemma 18. (*Pumping lemma*). Let $L \in \mathcal{P}$. Then there exist numbers k, l such that any word $\omega \in L$, with $|\omega| \geq k$, has a decomposition $\omega = xyz$ with $1 \leq |y| \leq l$ such that $xy^iz \in L, \forall i \geq 1$.

Proposition 19. $L = \{\overline{a^m b a^m b} \mid m \geq 0\}$ is not a P-type Petri net language.

Proof. Given k according to the pumping lemma, consider the word $\omega = a^k b a^k b \in L$. Obviously, there is no decomposition of this word that can satisfy the pumping lemma. $\square$

Let us define a new class of L-type Petri net languages whose prefix closure can be generated by a deterministic nonblocking Petri net generator. This class will play an important role in characterizing the existence of Petri net supervisors, as will be discussed in Section 4.2.

Definition 20. A language $L \in \mathcal{L}$ (not necessarily deterministic) is said to be *deterministic P-closed* (DP-closed for short) if and only if its prefix closure is a deterministic P-type Petri net language, i.e., $\overline{L} \in \mathcal{P}_d$. The class of DP-closed Petri net languages is denoted $\mathcal{L}_{DP}$.

As a side note, it should be pointed out that the class $\mathcal{L}_{DP}$ that we have defined does not coincide with any of the classes of PN languages generally considered in literature [17]. The proof follows from the fact that $\mathcal{L}_{DP}$ is not closed under intersection, as shown in [10], while all previously defined classes of PN languages are closed under intersection.

When the marked language of a net is its G-type Petri net language, the trimming of the net is always possible. In fact, next theorem proves that all deterministic G-type languages are DP-closed. In [11], was actually proven a slightly stronger property, namely that given a deterministic PN generator $G = (N, \ell, M_0, F)$ with $\overline{L_g(G)} \subsetneq L(G)$, there exists a finite procedure to construct a new deterministic PN generator G' such that $L_g(G') = L_g(G)$ and $L(G') = \overline{L_g(G')}$.

Theorem 21. *The class $\mathcal{G}_d$ is included in $\mathcal{L}_{DP}$.*

4.2 Supremal controllable sublanguage and Petri net supervisors

In this section the closure of PN languages under the *supremal controllable sublanguage operator* $^\uparrow$ [41] is discussed and some necessary and sufficient conditions for the existence of PN supervisors are given.

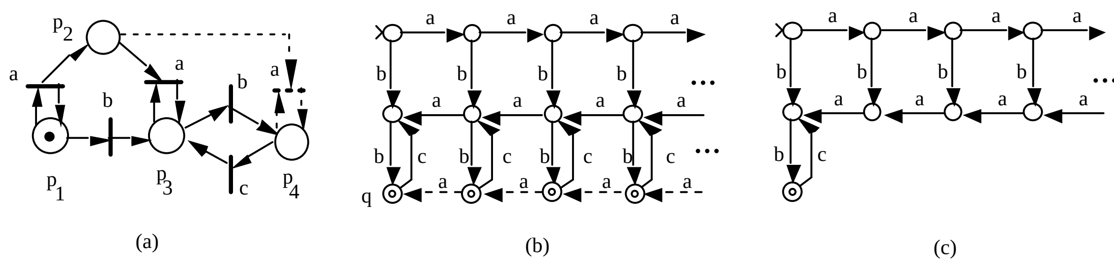

Figure 5: Generator in Example 23 (a); its rechability tree (b); generator of $L(E)^{\uparrow}$ and $L_{\ell}(E)^{\uparrow} = L_g(E)^{\uparrow}$ in Example 23.

Theorem 22. The classes $\mathcal{P}_d$, $\mathcal{G}_d$ and $\mathcal{L}_{DP}$ of PN languages are not closed under the $^{\uparrow}$ operator.

The proof of this proposition will be given by means of the following example.
Example 23. Let G be the PN generator in Figure 5.(a) (including the dotted arcs and transition) with $\Sigma_u = \{a\}$, and set of final markings $F = \{(0001)^T\}$. Let now E be the same net without the dotted arcs and transition and with the same set of final markings $F = \{(0001)^T\}$. The languages $L(E) \in \mathcal{P}_d$, $L_g(E) \in \mathcal{G}_d$ and $L_{\ell}(E) \in \mathcal{L}_{DP}$ are not controllable. To show this in Figure 5.(b) are drawn the reachability trees of the two nets; since E refines G, the arcs that belong to the reachability tree of both nets have been represented with continuous lines, while the arcs that only belong to the reachability tree of G have been represented by dotted lines. $L(E)$ is the prefix language of the generator in Figure 5.(b), $L_g(E)$ is the language accepted reaching any final state, and $L_{\ell}(E)$ is the language accepted reaching state q. None of these languages is controllable because of the presence of the dotted arcs associated to the uncontrollable transition a. Applying the $^{\uparrow}$ operator, one obtains the supremal controllable sublanguages: $L(E)^{\uparrow} = \{\overline{a^m b a^m (bc)^* b} \mid m \geq 0\}$ and $L_{\ell}(E)^{\uparrow} = L_g(E)^{\uparrow} = \{a^m b a^m (bc)^* b \mid m \geq 0\}$, that are the closed and marked behavior of the generator in Figure 5.(c). $L(E)^{\uparrow} \notin \mathcal{P}_d$, as can be proved using the pumping lemma (the word $\omega = a^k b a^k b$ may be used to prove that no pumping is possible). $L_{\ell}(E)^{\uparrow} \notin \mathcal{L}_{DP}$, since $\overline{L_{\ell}(E)^{\uparrow}} = L(E)^{\uparrow} \notin \mathcal{P}_d$. $L_g(E)^{\uparrow} \notin \mathcal{G}_d$, since $\mathcal{G}_d \subset \mathcal{L}_{DP}$. $\diamond$

Given a language L in any of the previous classes, say in $\mathcal{G}_d$, when the language $L^{\uparrow}$ is not in $\mathcal{G}_d$, one may be tempted to consider the supremal element of the class: $\mathcal{C}_g(L) = \{K \mid K \subseteq L, K$ is controllable, $K \in \mathcal{G}_d\}$. Note, however, that this supremal element does not always exist. In fact, the existence and uniqueness of the supremal controllable sublanguage in [41] follows from the fact that the class of controllable languages is closed under arbitrary union, while the classes of deterministic languages are not closed under union (see [11]). In the previous example, for instance, for all $i \in \mathbb{N}$ the language $K_i = \{a^n b a^n (bc)^* b \mid n \leq i\}$ is in $\mathcal{C}_g(L_g(E))$ and $K_i \subset K_{i+1}$. As shown in Example 23, $K_\infty = L_g(E)^{\uparrow}$ is not in $\mathcal{G}_d$.

Finally, let us consider two theorems [10] that give necessary and sufficient conditions for the existence of PN supervisors.

The first theorem, similar to Theorem 9, regards the case in which one wants to restrict the closed behavior of G within the limits of a legal behavior L.
Theorem 24. *Let G be a PN generator and let $L \subseteq L(G)$ be a non empty language. There exists a PN supervisor S such that $L(S/G) = L$ iff $L \in \mathcal{P}_d$ and L is controllable.*

Let us consider now the case in which one wants to restrict the marked behavior of G within the limits of a legal behavior L. In this case, unfortunately, the necessary requirements that L be controllable and $L_m(G)-$closed (by Theorem 10) are not sufficient to insure the existence of a nonblocking PN supervisor even if $L \in \mathcal{L}_d$. The additonal assumption that legal behavior L be DP-closed is needed.
Theorem 25. *Let G be a nonblocking PN and let $L \subseteq L_m(G)$ be a nonempty language. There exists a nonblocking PN supervisor S such that $L_m(S/G) = L$ iff $L \in \mathcal{L}_{DP}$, L is controllable, and L is $L_m(G)-$closed.*

4.3 Decidability

In this section the decidability of some properties of discrete event systems are derived from the closure properties of the corresponding languages. The results were originally presented in [11].

It is well known [30] that the inclusion problem: "Is $L_1 \subseteq L_2$?", with $L_1, L_2 \in \mathcal{P}$, is undecidable. However, the emptiness problem: "Is $L = \emptyset$?", with $L \in \mathcal{L}$, is decidable since it may be reduced to the reachability problem, shown to be decidable [25].
Definition 26. *Given a language $L \subseteq \Sigma^*$, its complement language is $\mathbf{C}L = \Sigma^* \setminus L$. Given a class of languages $\mathcal{A}$, let us denote co-$\mathcal{A} = \{\mathbf{C}L \mid L \in \mathcal{A}\}$ the class of all the complements of languages in $\mathcal{A}$.*

For deterministic PN languages the following lemma holds.
Lemma 27. *The inclusion problem: "Is $L_1 \subseteq L_2$?" is decidable if $L_1 \in \mathcal{L}$ and $L_2 \in \mathcal{L}_d \cup \mathcal{G}_d$.*

Proof. Pelz [29] noted that if $\mathbf{C}L_2 \in \mathcal{L}$ the inclusion problem may be reduced to the emptiness problem for the language $L = L_1 \cap \mathbf{C}L_2 \in \mathcal{L}$. She also proved that co-$\mathcal{L}_d \subseteq \mathcal{L}$. In [8] it was shown that co-$\mathcal{G}_d \subseteq \mathcal{L}$. □

A different proof for the decidability of the inclusion problem when $L_1, L_2 \in \mathcal{P}_d$ has also been presented in [35].

Using this lemma, it is possible to prove the following propositions that shows that three important properties, blockingness, L-closure, and controllability are decidable for deterministic systems [11].

Proposition 28. *It is decidable whether a deterministic Petri net generator G is blocking if $L_m(G) \in \mathcal{L}_{DP}$.*

Proposition 29. *It is decidable whether a language $K \in \mathcal{L}_{DP}$ is controllable with respect to a Petri net generator G.*

Proposition 30. *It is decidable whether a language $K \in \mathcal{G}_d \cup (\mathcal{L}_d \cap \mathcal{L}_{DP})$ is L-closed with $L \in \mathcal{L}$.*

A remark on the complexity of these decision procedures. Assume G_1 and G_2 are two nets whose closed (or marked) behaviors are the languages L_1 and L_2. As suggested by the proof of Lemma 27, to check whether $L_1 \subseteq L_2$ one may follow these steps: (a) construct a PN G_2' generating $\mathbf{C}L_2$ (this is possible if G_2 is deterministic); (b) construct the net G as the intersection of the nets G_1 and G_2'; (c) check whether the language generated by G is empty.

The first step may be carried out with the construction shown in [29], whose complexity has not been computed. The second step may be done efficiently, as shown in [9]. The last step has the same complexity of checking the reachability of a given marking, that is at best decidable in exponential space [17].

More efficient techniques for deciding these properties may exist. This is an open area for further work.

4.4 Other issues

Gaubert and Giua [7] have explored the use of infinite sets of final markings in the definition of the marked behavior of a net. With each more or less classical subclass of subsets of $\mathbb{N}^m$ — finite, ideal (or upper), semi-cylindrical, star-free, recognizable, rational (or semilinear) subsets — it is possible to associate the class of Petri net languages whose set of accepting states belongs to the class. When comparing the related Petri net languages, it was seen that for arbitrary or λ-free PN generators, the above hierarchy collapses: one does not increase the generality by considering semilinear accepting sets instead of the usual finite ones. However, for free-labeled and deterministic PN generators, it is shown that one gets new distinct subclasses of Petri net languages, for which several decidability problems become solvable.

Another interesting class of problems, the existence and design of supervisors for PN generators subject to regular legal behaviors has been discussed by Kumar and Holloway

[21].

Sreenivas has also explored the decidability and the existence of control structures for enforcing on PN's several classes of specifications that cannot be easily expressed as languages.

In [36] a necessary and sufficient condition is derived for the existence of a supervisory policy that enforces *liveness* in partially controlled PN's. This condition is decidable if all transitions are controllable or if the PN is bounded. However, for unbounded nets with uncontrollable transitions the condition becomes undecidable. In [37] the supervisory policies that enforce liveness in non-live free-choice nets are characterized with a technique similar to Commoner's Liveness Theorem. Enforcing liveness is complex problem that may not always be solved using supervisory policies. However, Sreenivas showed that if a supervisory policy that enforces liveness exists, then there exist a minimally restrictive policy as well.

In [38] decidable procedures are given to check the existence of a supervisory policy that enforces *global fairness* and *bounded fairness* in partially controlled PN's and to check the existence of a minimally restrictive policy that enforces the above notions of fairness for bounded PN's. The class of policies that enforces fairness is not closed under disjunction, so a minimally restrictive policy may not exist.

5 Generalized Mutual Exclusion Constraints

A classic approach to discrete event modeling and control considers complex systems as being built out of interacting subsystems. Depending on the particular tasks demanded to the system, and on the way the subsystems are interconnected, specific constraints must be imposed on the system's behavior. In this section discrete event systems modelled by Petri nets are considered and a class of constraints, that can be seen as a generalization of *mutual exclusion* (or *mutex*) constraints, is discussed.

Several authors have presented solutions to this problem. The notation and definitions vary from author to author. Here some of these approaches are reviewed presenting them with a consistent notation.

5.1 Monitors

Let us define a *generalized mutual exclusion constraint* (GMEC) as a condition that limits a weighted sum of tokens contained in a subset of places [12, 13].

Definition 31. *Let $\langle N, M_0 \rangle$ be a net system with set of places P, $\vec{w} : P \to \mathbb{Z}$ a weight vector of integers, and $k \in \mathbb{Z}$ a constant. The* support *of $\vec{w}$ is the set $Q_w = \{p \in P \mid w(p) \neq 0\}$. A single* generalized mutual exclusion constraint *$(\vec{w}, k)$ defines a set of legal markings on $\langle N, M_0 \rangle$ $\mathcal{M}(\vec{w}, k) = \{M \in R(N, M_0) \mid \vec{w}^T \cdot M \leq k\}$. A set of GMEC's $(W, \vec{k})$, with $W = [\vec{w}_1 \dots \vec{w}_r]$ and $\vec{k} = (k_1 \dots k_r)^T$, defines a set of legal markings $\mathcal{M}(W, \vec{k}) = \{M \in R(N, M_0) \mid W^T \cdot M \leq \vec{k}\}$.*

Markings in $R(N, M_0)$ that are not legal will be denoted *forbidden* markings.
Example 32. As an example of the modeling power of GMEC's, consider the simple manufacturing process with two machines, a robot and a buffer shown in Figure 6.(a). There exists an infinite supply of parts of type 1 (type 2) that are loaded by the robot on machine 1 (machine 2). After machining the parts are directly deposited into the buffer. Machined parts are taken in pairs from the buffer to be assembled.

The process can be represented by the net in Figure 6.(b), where: t_1 and t_2 (t_4 and t_5) represent the start and the end of the loading operation of machine 1 (machine 2); t_3 (t_6) represents the storing of a part of type 1 (type 2) in the buffer; t_7 represents the withdrawal of two parts of different type from the buffer. The places have the following interpretation: p_1 (p_4) represents the parts being loaded on machine 1 (machine 2); p_2 (p_5) represents the parts being machined on machine 1 (machine 2); p_3 (p_6) represents the parts of type 1 (type 2) in the buffer.

The following constraints should be imposed on the system's behavior. a) Only one robot is available, hence only one loading operation may be executed at a given time, i.e., $M(p_1) + M(p_4) \leq 1$. b) Only one part can be machined at a given time on each machine, i.e., $M(p_2) \leq 1$, and $M(p_5) \leq 1$. c) The buffer has k slots, and each part of type 1 takes two slots, while each part of type 2 takes one slot. To avoid overflow one wants $2M(p_3) + M(p_6) \leq k$. d) In the buffer, the number of parts of one type should not exceed the number of parts of the other type by more that k' units, i.e., $M(p_3) - M(p_6) \leq k'$, and $M(p_6) - M(p_3) \leq k'$. $\diamond$

As this very simple example shows, mutual exclusion constraints are a natural way of expressing the concurrent use of a finite number of resources, shared among different processes. The use of weights permits to assign different units of resources to the various processes. The use of negative weights permits to express fairness constraints in the allocation of the shared resources. Note also that constraints to prevent underflow (i.e., of the form $M(p_1) + M(p_2) \geq k \geq 0$) may also be expressed, using negative weights, as $-M(p_1) - M(p_2) \leq -k$.

It is important to note that GMEC problems may be solved by *state feedback*. In fact, at each step the control law depends on the present marking of the net, and not on its

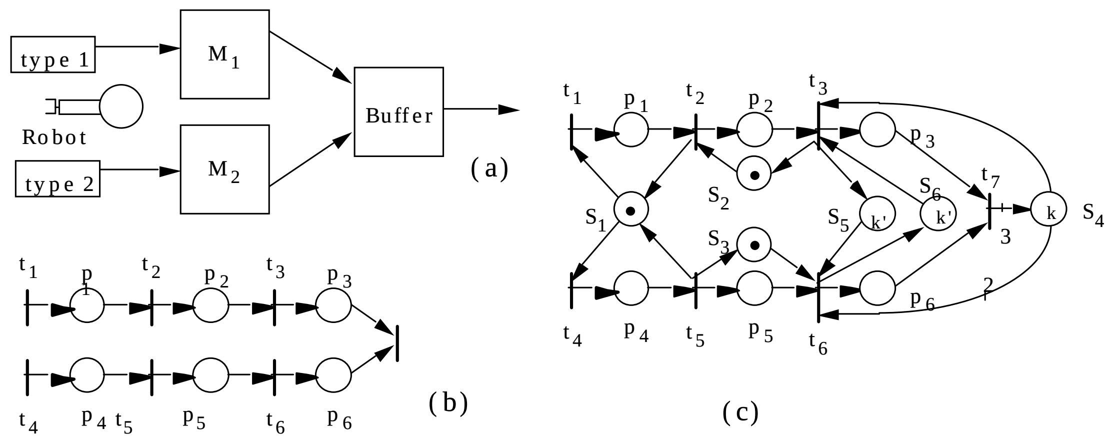

Figure 6: A manufacturing process: (a) layout; (b) Petri net model; (c) Petri net controlled with monitors.

previous evolution.

In traditional Petri net modeling all transitions are assumed to be *controllable*, i.e., may be prevented from firing by a control agent. A single GMEC may be easily implemented by a *monitor*, i.e., a place whose initial marking represents the available units of a resource and whose outgoing and incoming transitions represent, respectively, the acquisition and release of units of the resource (see [5, 43] for some manufacturing examples).

Definition 33. *Given a system $\langle N, M_0 \rangle$, with $N = (P, T, Pre, Post)$, and a GMEC $(\vec{w}, k)$, the* monitor *that enforces this constraint is a new place S to be added to N. The resulting system is denoted $\langle N^S, M_0^S \rangle$, with $N^S = (P \cup \{S\}, T, Pre^S, Post^S)$. Let C be the incidence matrix of N. Then N^S will have incidence matrix*

$$C^S = \left[\begin{array}{c} C \\ -\vec{w}^T \cdot C \end{array} \right].$$

Note that there are no selfloops containing S in N^S, hence Pre^S and $Post^S$ may be uniquely determined by C^S. The initial marking of $\langle N^S, M_0^S \rangle$ is

$$M_0^S = \left(\begin{array}{c} M_0 \\ k - \vec{w}^T \cdot M_0 \end{array} \right).$$

The initial marking M_0 of the system is assumed to satisfy the constraint $(\vec{w}, k)$.

According to the definition, the monitor that enforces a constraint $(\vec{w}, k)$ will have arcs going to (coming from) all input (output) transitions of a place $p \in |Q_w|$ and such that $w(p) > 0$. If the place p is such that $w(p) < 0$, then the directions of the arcs is reversed.

The weight of these arcs depends on the coefficient of $\vec{w}$ and on the coefficients of the incidence matrix C.

As an example, in Figure 6.(c) monitors have been added to the net in Figure 6.(b) to enforce the constraints discussed in Example 32. Monitor S_1 enforces the constraint a); monitors S_2 and S_3 constraints b); monitor S_4 constraint c); and monitors S_5 and S_6 constraints d).

A set of GMEC's can be enforced adding a monitor for each constraint in the set.

The following theorem was proved in [12].
Theorem 34. *Let $\langle N, M_0 \rangle$ be a system, $(\vec{w}, k)$ a GMEC, and $\langle N^S, M_0^S \rangle$ the system with the addition of the corresponding monitor S. 1] S ensures that the projection on P of the reachability set of $\langle N^S, M_0^S \rangle$ is contained in the set of legal reachable markings of $\langle N, M_0 \rangle$, i.e., $R(N^S, M_0^S) \uparrow_P \subseteq \mathcal{M}(\vec{w}, k)$.*
2] S minimally restricts the behavior of $\langle N^S, M_0^S \rangle$, in the sense that it prevents only transition firings that yield forbidden markings.

Part 1 of the previous theorem means that the addition of a monitor to the net structure modifies the behavior of a system, to avoid reaching markings that do not satisfy the corresponding GMEC. Part 2 implies that the monitor is also maximally permissive, i.e., it only disables transitions whose firing would yield a marking that violates the corresponding GMEC.

Let us compare GMEC's with the most general kind of constraint that can be defined on the marking set of a system, the *forbidden markings* constraint [15, 19]. A forbidden marking constraint consists of an *explicit list* of markings $\mathcal{F}$ that one wants to forbid.

Let $\mathcal{F}$ be any set of forbidden markings on a net system $\langle N, M_0 \rangle$. Is it possible to find a set of GMEC's $(W, \vec{k})$ equivalent to $\mathcal{F}$, i.e., such that $R(N, M_0) \setminus \mathcal{F} = \mathcal{M}(W, \vec{k})$? In general the answer is no. In fact given three markings $M_1, M_2, M_3 \in R(N, M_0)$ with $M_3 = (M_1 + M_2)/2$ it follows that $M_1, M_2 \in \mathcal{M}(W, \vec{k}) \Longrightarrow M_3 \in \mathcal{M}(W, \vec{k})$, since the set of vectors that satisfy a GMEC is a convex set. However, $\mathcal{F}$ may be chosen such that $M_1, M_2 \notin \mathcal{F}$ and $M_3 \in \mathcal{F}$. This proves that there may not exist a GMEC equivalent to a forbidden marking constraint.

For some classes of nets there exists a set of GMEC's equivalent to any forbidden marking constraint (see [12]).
Theorem 35. *Let $\langle N, M_0 \rangle$ be a safe net system. Then given a set of forbidden markings $\mathcal{F}$ there exists a set of GMEC's $(W, \vec{k})$ such that $R(N, M_0) \setminus \mathcal{F} = \mathcal{M}(W, \vec{k})$.*

5.2 Nets with uncontrollable transitions

In the framework of supervisory control the complexity of enforcing a GMEC is enhanced by the presence of *uncontrollable* transitions [14].

Let us assume that the set of transitions T of a net is partitioned into two disjoints subsets: T_u, the set of *uncontrollable* transitions, and T_c, the set of *controllable* transitions. A controllable transition may be disabled by a *supervisor*, a controlling agent which ensures that the behavior of the system is within a legal behavior. An uncontrollable transition represents an event which may not be prevented from occurring by a supervisor. Thus arcs from monitors to uncontrollable transitions are not allowed, since the effect of such an arc would be that of preventing the firing of the transition when the monitor place is not marked.

When the net has uncontrollable transitions, to enforce a given GMEC it is necessary to prevent the system from reaching a superset of the forbidden markings, containing all those markings from which a forbidden one may be reached by firing a sequence of uncontrollable transitions.

Given a system $\langle N, M_0 \rangle$ and a set of GMEC's $(W, \vec{k})$, in the presence of uncontrollable transitions the set of legal markings is given as: $\mathcal{M}_c(W, \vec{k}) = \mathcal{M}(W, \vec{k}) \setminus \{M \in R(N, M_0) \mid \exists M' \notin \mathcal{M}(W, \vec{k}), M[\sigma\rangle M' \wedge \sigma \in T_u^*\}$, i.e., one does not consider legal the markings that satisfy $(W, \vec{k})$ but from which a forbidden marking may be reached by firing only uncontrollable transitions. it is necessary to introduce this restriction because a firing sequence $\sigma \in T_u^*$ may not be prevented by a controlling agent.

Let us now discuss the concept of *maximally permissible control policy* [18], that is the counterpart of the supremal controllable sublanguage discussed in Section 3.6. When all transitions are controllable, it was shown that a *monitor* is capable of enforcing a given GMEC $(\vec{w}, k)$, with a maximally permissible control, in the sense that *a*) only markings in $\mathcal{M}(\vec{w}, k)$ can be reached by the system under control; *b*) all transition firings that yield a marking in $\mathcal{M}(\vec{w}, k)$ are be allowed.

In the case of uncontrollable transitions, the maximally permissible control policy should ensure that: *a*) only markings in $\mathcal{M}_c(\vec{w}, k)$ will be reached by the system under control; *b*) all transition firings that yield a marking in $\mathcal{M}_c(\vec{w}, k)$ should be allowed.

It is possible to prove [12] that there may not exist a GMEC $(W, \vec{k})$ such that $\mathcal{M}(W, \vec{k}) = \mathcal{M}_c(\vec{w}, k)$. Thus in presence of *uncontrollable* transitions, a problem of mutual exclusion is transformed into a more general *forbidden marking problem*, which is a qualitatively different problem, in the sense that it may not always be solved with the same techniques used when all transitions are controllable. Note, however, that for safe and conservative systems the result of Theorem 35 ensures that, even if some transitions are not controllable, $(\vec{w}, k)$ may be enforced by a set of monitors.

5.3 Petri net techniques for GMEC's

Two ways have been explored for reducing the computational complexity involved in solving a GMEC problem for nets with uncontrollable transitions.

On one hand, one may consider special PN structures for which the maximally permissible control policy can be easily computed and implemented. There have been several interesting approaches in this sense. Holloway and Krogh [15, 19] presented an approach in which the problem of controlling the marking of a place can be decomposed into the control of paths of uncontrollable transitions and used these techniques to enforce GMEC's on safe marked graphs. Li and Wonham [22, 23] showed how closed-form solutions for GMEC problems may be computed for restricted classes of nets. By closed-form solution the authors mean that the controller may be represented as a net. Giua et al. [13] have discussed several control structures (including monitors) capable of enforcing GMEC's on marked graphs with control safe places.

On an another hand, one may give up the requirement that the control policy be maximally permissible and may be willing to accept a more restrictive control policy provided it can be easily computed. This approach has been followed by Moody et al. [26, 27, 42]. In their approach the idea is that of always using very simple controllers in the form of monitor places that only constrain controllable transitions. An algorithm is given to compute such a monitor to ensure that a given GMEC will never be violated.

These approaches will be briefly reviewed in the following.

5.3.1 Holloway and Krogh's approach

The PN model considered by Holloway and Krogh is called *controlled PN*. A controlled PN is a P/T net with 2 sets of places, the state places P represented by circles and the input control places C represented as boxes (see Figure 7.(a)). Hence the marking has two components, M (related to P) and U (related to C). U is assumed to be binary.

The marking of places in C is computed by an external agent as a function of the marking of the state places, i.e., $U = f(M)$. The firing of a transition modifies M in the usual way. The control input U is computed again each time a new marking is reached. There are two difference wrt the firing policy of P/T nets: a) two enabled transitions may fire simultaneously; b) control places allow interpreted parallelism (no conflict), i.e., if a control place c is the input place for two or more transitions, and $U(c) = 1$, then all its output transitions may fire if they are also enabled by M.

The basic restriction Holloway and Krogh consider is that the net be a *safe marked graph*, i.e., a safe P/T net such that each place has exactly one input arc and one output arc. In the examples given here, only GMEC $(\vec{w}, k)$ where $w(p) \in \{0, 1\}$ will be considered.

Given a GMEC $(\vec{w}, k)$ the control law is computed in two steps.

Off-line computation. For each place in Q_w compute backwards the paths until controlled transitions are found.

On-line computation. Given a marking M define for each $p \in Q_w$: a) $\Lambda_p(M) = 1$ if all its path are marked (i.e., p may be marked uncontrollably) else $\Lambda_p(M) = 0$; b) $\Delta_p(M) = 1$ if, unless some control input is disabled, at the next marking all paths of p may be marked (M is a boundary marking for p) else $\Delta_p(M) = 0$.
Let: L be the number of places in Q_w such that $\Lambda_p(M) = 1$; B be the number of places in Q_w such that $\Delta_p(M) = 1$; $D(U)$ be the number of places in Q_w such that $\Delta_p(M) = 1$ but such that U disables some transition whose firing will increase Λ_p. Then a control input is admissible if: $D(U) \geq L + B - k$.
Example 36. Let us consider the GMEC $M(p_1) + M(p_2) \leq 1$. In Figure 7.(a), we have represented the subnet computed during the off-line computations for places p_1 and p_2.

Given the marking in the figure, $\Lambda_{p1}(M) = \Lambda_{p2}(M) = 0$, and $\Delta_{p1}(M) = \Delta_{p2}(M) = 1$, hence $L = 0$, $B = 2$. The possible markings for the control places and the corresponding values of D are: $D(0\ 0\ 0) = D(0\ 1\ 0) = 1$ (p_1 and p_2 cannot be marked); $D(0\ 0\ 1) = D(0\ 1\ 1) = 1$ (p_1 cannot be marked); $D(1\ 0\ 0) = D(1\ 1\ 0) = 1$ (p_2 cannot be marked); $D(1\ 0\ 1) = D(1\ 1\ 1) = 1$ (both p_1 and p_2 can be marked).

One needs $D(U) \geq L + B - k = 1$, hence one can choose either $U = (0\ 1\ 1)^T$ or $U' = (1\ 1\ 0)^T$, i.e., there are two maximally permissible controls. ◇

The approach of Holloway and Krogh is extremely efficient, since it requires very simple computations, both in the off-line and on-line steps. However, because the controller is given as a feedback law it is not possible to built a net model of the closed-loop system.

This approach has received a lot of attention in the literature and has also been extended to classes of nets other than marked graphs: controlled state machines [2], forward and backward conflict-free nets [3], colored nets [1, 24].

Finally, necessary and sufficient conditions for liveness under control have been presented in [16].

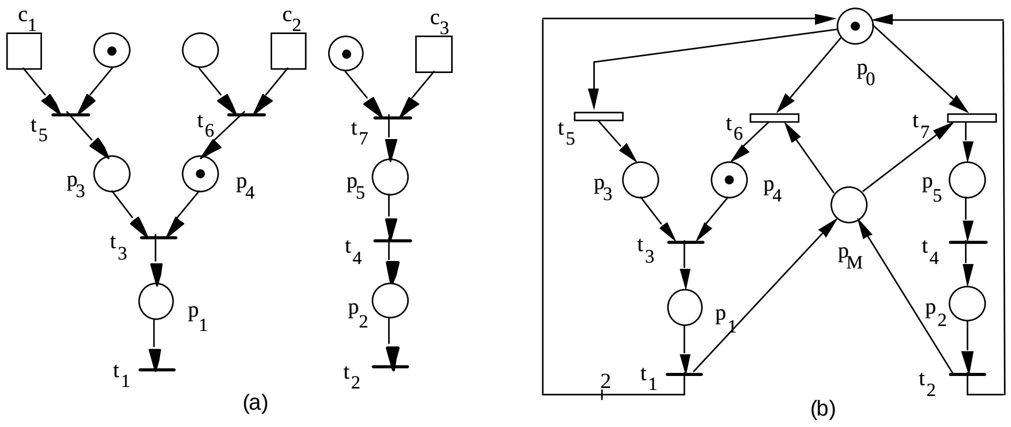

Figure 7: Nets in Examples 36-39.

5.3.2 Li and Wonham's approach

Li and Wonham have considered Vector Discrete Event Systems, a model that is known to be equivalent to Petri nets [30].

In a first approach they used incidence matrix analysis to compute the control law that enforces GMEC's (that they call Linear Predicates).

Let $\langle N, M_0 \rangle$ be a net system, and let N_u be the uncontrollable subnet, i.e., the subnet obtained from N by removing all controllable transitions. Let $(\vec{w}, k)$ be a GMEC. Then one can write the set of legal markings defined in Section 5.2 as: $\mathcal{M}_c(w,k) = \{M \mid (\forall M' \in R(N_u, M))\vec{w}^T \cdot M' \leq k\}$.

They also observed that the state equation of a net may be used to decide reachability if the net is *acyclic*, i.e., if no direct path in the net forms a cycle [28]. Thus if N_u is acyclic, a given marking M is in $\mathcal{M}_c(w,k)$ if and only if the following integer programming problem IPP (where C_u is the incidence matrix of N_u) has solution $x^* \leq k$:

$$\begin{array}{ll} x = & \max \vec{w}^T \cdot M' \\ s.t. & M' = M + C_u \cdot \vec{\sigma}, \quad . \\ & M', \vec{\sigma} \geq \vec{0}. \end{array}$$

The following algorithm can used to enforce a GMEC $(\vec{w}, k)$ on nets whose uncontrollable subnet is acyclic.

```
1. Let the initial marking $M_0$ be in $\mathcal{M}_c(w,k)$. (Solve IPP with $M = M_0$.)
2. Let $\overline{M}$ be the present marking. For any controllable $t$ enabled by $\overline{M}$:
    a) compute $M_t$ such that $\overline{M}[t\rangle M_t$;
    b) solve $IPP$ with $M = M_t$;
```

c) if $x^* \leq k$ then t should be enabled else it should be disabled.

3. As soon as a transition fires go back to step 2.

The condition that the uncontrollable subnet be acyclic is not too restrictive, in the sense that in many practical applications it holds. However, the problem is that the controller has to solve on-line at each step several IPP's. The complexity of IPP's is an open problem. It is doubtful that these problems have polynomial complexity in the size of the constraint set. Thus, the approach is unfeasible in practical applications.

This motivated Li and Wonham to study other classes of nets for which the controller may be represented as a net. The method can be applied to ordinary nets whose uncontrollable transitions form either tree structures of type TS1 (each transition has a single output arc) or tree structures of type TS2 (each transition has a single input arc) as defined in [23]. Another mild restriction is that the uncontrollable subnet be composed by non connected components and two or more places in Q_w cannot belong to the same component (mutual independence).

In the case of TS2 nets they show that given a GMEC $(\vec{w}, k)$, there exists a new constraint $(\vec{w}_c, k_c)$ such that the set of legal markings can be written as: $\mathcal{M}_c(w, k) = \{M \mid \vec{w}_c^T \cdot M \leq k_c\}$ and the monitor that enforces $(\vec{w}_c, k_c)$ does not have arcs going to uncontrollable transitions. Thus $(\vec{w}, k)$ can be enforced using the monitor for $(\vec{w}_c, k_c)$.

In the case of TS1 nets they show that given a GMEC $(\vec{w}, k)$, there exists a new set of constraints $(\vec{w}_i, k_i)$ $(i = 1, \ldots, r)$ such that: $\mathcal{M}_c(w, k) = \{M \mid \bigvee_{i=1}^{r} \vec{w}_i^T \cdot M \leq k_i\}$ ($\bigvee$ is the disjunction operator, i.e., the logical OR) and the monitor that enforces each $(\vec{w}_i, k_i)$ does not have arcs going to uncontrollable transitions. Note, however, that in this second case the net structure corresponding to the disjunction operator cannot be represented as a net. In fact the addition of r monitors, one for each $(\vec{w}_i, k_i)$, would enforce the conjunction (logical AND) of these constraints.

In [23] is defined a new structure, called *generalized vector addition system*, that can enforce a disjunction of constraints.

Example 37. Let us consider the GMEC $(\vec{w}, k)$ requiring $M(p_1) + M(p_2) \leq 1$ and the net in Figure 7.(a). The uncontrollable subnets for p_1 and p_2 (controllable transitions t_5, t_6 and t_7 should be removed), are TS1 nets.

Let $(\vec{w}_1, k_1)$ and $(\vec{w}_2, k_2)$ be the constraints requiring, respectively: $M(p_1) + M(p_2) + M(p_3) + M(p_5) \leq 1$ and $M(p_1) + M(p_2) + M(p_4) + M(p_5) \leq 1$. Clearly, $\mathcal{M}_c(w, k) = \{M \mid \vec{w}_1^T \cdot M \leq k_1 \vee \vec{w}_2^T \cdot M \leq k_2\}$. ◇

5.3.3 Giua, DiCesare, and Silva's approach

The authors consider marked graphs with *control safe places* and GMEC's with $\vec{w} \in \mathbb{N}^m$.

A transition t belongs to the set of *control transitions* A_p of a place p iff: a) t is controllable; b) there exists a path from t to p that does not contain controllable transitions except t. A place p is *control safe* if on at least one path from each $t \in A_p$ to p the number of tokens cannot exceed one. A transition $t \in A_p$ is said to be *constraining* at a marking M if there exists a path from t to p that is unmarked at M.

For this classes of nets and constraints, the authors showed that given a GMEC $(\vec{w}, k)$, there exists a set of GMEC's $(W_c, \vec{k}_c)$ such that $\{M \mid W_c^T \cdot M \leq \vec{k}_c\} = \mathcal{M}_c(\vec{w}, k)$. Hence a maximally permissible control law for $(\vec{w}, k)$ may always be implemented by a set of monitors.

In particular, if $(\vec{w}, k)$ is such that $w(p) \in \{0, 1\}$ and $|Q_w| = k + 1$, then the set $(W, \vec{k})$ reduces to a single constraint. This constraint can be enforced by a monitor place p_0 with (for all $p \in Q_w$): a) one arc going from p_0 to each transitions in A_p ; b) $|A_p|$ arcs going from the output transition of p to p_0.

The initial marking of p_0 is equal to $d-1$, where d is the number of constraining transitions in $A = \cup_{p \in Q_w} A_p$ at the initial marking M_0.
Example 38. Let us consider the GMEC $(\vec{w}, k)$ requiring $M(p_1)+M(p_2) \leq 1$ and the net in Figure 7.(b) without places p_0 and p_M and their input/output arcs. Here the controllable transitions (t_5, t_6 and t_7) are shown as boxes.

Let us assume that places p_1 and p_2 are control safe. The set control transitions for place p_1 is $A_{p1} = \{t_5, t_6\}$. The set of control transitions for place p_2 is $A_{p2} = \{t_7\}$. Transitions t_5 and t_7 are constraining.

Since the constraint $(\vec{w}, k)$ is such that $|Q_w| = |\{p_1, p_2\}| = 2 = k + 1$ and $w(p_1) = 1$, $w(p_2) = 1$, this constraint can be enforced by a single monitor, place p_0 in Figure 7.(b). $\diamond$

Assume now $(\vec{w}, k)$ is such that (for some p) $w(p) > 1$, or $|Q_w| > k + 1$. The previous construction may not be used. However it was shown in [13] that the original constraint may be rewritten as a set of at least r constraints $(\vec{w}_j, k)$ where $|Q_{w_j}| = k + 1$ and each of these constraints may be enforced by a monitor. However the problem is that $r = \begin{pmatrix} |Q_w| \\ k+1 \end{pmatrix}$, thus in the worst case the number of monitors is exponential with respect to the cardinality of Q_w.

In the cases in which maximally permissible monitor solutions are not efficient, a different supervisory based control structure can be used [13]. This structure grows linearly with the number of places in the support of the weight vector.

5.3.4 Moody, Antsaklis, and Lemmon's approach

These authors use monitors as control structure to be added to the net structure for enforcing GMEC's (called *place invariants*). When there are uncontrollable transitions, they still use monitor based solutions but in this case the solution may not be maximally permissible.

Let us consider a set of GMEC's $(W, \vec{k})$ to be enforced on a net system $\langle N, M_0 \rangle$. Let W be a $(m \times r)$ matrix, where m is the number of places of the net and r the number of constraints in the set. Let C be the incidence matrix of the net and C_u the incidence matrix of the uncontrollable subnet (obtained removing all controllable transitions). The set of monitors corresponding to $(W, \vec{k})$ can be added to net without disabling any uncontrollable transition if all elements in $W^T \cdot C_u$ are less than zero, because in this case there will be no arcs going from the monitors to uncontrollable transitions.

If such is not the case, one can try to find a new constraint $(W_c, \vec{k_c})$ (where W_c is a matrix with the same dimension of W) such that: a) $W_c^T \cdot C_u$ has all elements less than zero; b) $\mathcal{M}(W_c, \vec{k_c}) \subseteq \mathcal{M}_c(W, \vec{k})$, i.e., such that all markings that are legal for $(W_c, \vec{k_c})$ are also legal for $(W, \vec{k})$. Note that the set of legal markings for the new constraint may be a strict subset of the set $\mathcal{M}_c(W, \vec{k})$ and thus the new constraint may prevent the net from reaching markings that are legal.

In [27] it was shown how one may try to find such a new set of constraints by performing row operations on the matrix $\begin{bmatrix} C_u \\ W^T \cdot C_u \end{bmatrix}$. The main idea is to add to the support of each constraint in W new places as will be shown in the next example.
Example 39. Let us consider the GMEC $(\vec{w}, k)$ requiring $M(p_1) + M(p_2) \leq 1$ and the net in Figure 7.(b) without places p_0 and p_M and their input/output arcs.

Here the uncontrollable incidence matrix is $C_u = [C(\cdot, t_1)\ C(\cdot, t_2)\ C(\cdot, t_3)\ C(\cdot, t_4) = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$. Since $\vec{w}^T \cdot C_u = (-1\ \ -1\ 1\ 1)$, the constraint cannot be enforced.

Let us add places p_4 and p_5 to the constraint (places p_3 and p_5 could also have been chosen) to obtain the new constraint $(\vec{w}_c, k_c)$ with $\vec{w}_c^T = (1\ 1\ 0\ 1\ 1)$ and $k_c = 1$. Now

$\vec{w}_c^T \cdot C_u = (-1 \ -1 \ 0 \ 0)$, hence this constraint can be enforced by a monitor. Note that all markings that satisfy the new constraint also satisfy the original one, since $M(p_1) + M(p_2) \leq M(p_1) + M(p_2) + M(p_4) + M(p_5) \leq 1$.

The monitor corresponding to $(\vec{w}_c, k_c)$ is place p_M in Figure 7.(b).

The constraint $(\vec{w}_c, k_c)$ from the marking in Figure 7.(a) prevents the firing of t_7 even if the marking $(0\ 0\ 0\ 1\ 1)^T$ reachable by firing t_7 is in $\mathcal{M}_c(\vec{w}, k)$. $\diamond$

It is also interesting to note that this approach can be extended by duality to unobservable transitions. In this case the requirement is that there should not be arcs from unobservable transitions to a monitor, because the occurrence of such a transition cannot be observed and cannot be used by the monitor to update its control law.

6 Conclusions

The paper has discussed several issues related to the use of Petri nets in the supervisory control of discrete event systems. The basic elements of supervisory control have been presented using Petri nets as discrete event models. The language properties of Petri nets have been studied in this framework to derive important results on the classes of control problems that can be effectively solved. Examples of the use of Petri net structural techniques have been given, discussing the design of control structures for enforcing linear constraints on the reachability set of a net.

References

[1] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem. On forbidden state problems for colored closed controlled state machines. In *Preprints of the 12th IFAC World Congress*, volume 4, pages 161–164, Sidney, Australia, July 1993.

[2] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem. On forbidden state problems for a class of controlled Petri nets. *IEEE Trans. on Automatic Control*, 40(10):1717–1731, 1995.

[3] H. Chen. 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*, pages 3098–3103, Lake Buena Vista, FL, Dec 1994.

[4] J.M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Sciences*, pages 113–145. Springer Verlag, New York, 1991.

[5] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F. B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman and Hall, London, 1993.

[6] S. Gaubert and A. Giua. Deterministic weak-and-marked Petri net languages are regular. *IEEE Trans. on Automatic Control*, 41(12), December 1996.

[7] S. Gaubert and A. Giua. Petri net languages with infinite sets of final markings. In *Proc. WODES96*, Edinburgh, Scotland, August 1996.

[8] A. Giua. On the closure properties of deterministic weak Petri net languages. Technical Report 59, Istituto di Elettrotecnica, University of Cagliari (Italy), February 1994.

[9] A. Giua and F. DiCesare. Supervisory design using Petri nets. In *Proc. 30th IEEE Conf. on Decision and Control*, pages 92–97, Brighton, UK, December 1991.

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

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

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

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

[14] C.H. Golaszewski and P.J. Ramadge. Mutual exclusion problems for discrete event systems with shared events. In *Proc. 27th IEEE Conf. on Decision and Control*, pages 234–239, Austin, Texas, December 1988.

[15] L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Trans. on Automatic Control*, 35(5):514–523, May 1990.

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

[17] M. Jantzen. Language theory of Petri nets. In W. Reisig W. Brauer and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986*, volume 254-I of *Lecture Notes in Computer Sciences*, pages 397–412. Springer Verlag, New York, 1987.

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

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

[20] R. Kumar and V.K. Garg. *Modeling and Control of Logical Discrete Event Systems.* Kluwer Academic Publisher, 1995.

[21] R. Kumar and L. E. Holloway. Supervisory control of deterministic Petri net languages with regular specification languages. *IEEE Trans. on Automatic Control*, 41(2):245–2494, February 1996.

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

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

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

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

[26] J.O. Moody and P.J. Antsaklis. Petri net supervisors for DES in the presence of uncontrollable and unobservable transitions. In *Proc. 33rd Annual Allerton Conference*, Monticello, IL., USA., October 1995.

[27] J.O. Moody, P.J. Antsaklis, and M.D. Lemmon. Automated design of a Petri net feedback controller for a robotic assembly cell. In *Proc. INRIA/IEEE Sym. on Emerging Technologies and Factory Automation*, volume 2, pages 117–128, Paris, France, October 1995.

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

[29] E. Pelz. Closure properties of deterministic Petri net languages. In *Proc. STACS 1987*, volume 247 of *Lecture Notes in Computer Sciences*, pages 373–382. Springer Verlag, New York, 1987.

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

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

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

[33] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, January 1989.

[34] M. Silva, J.M. Colom, and J. Campos. Linear algebraic techniques for the analysis of petri nets. In *Proc. Int. Symp. on Mathematical Theory of Networks and Systems*, Tokyo, Japan, 1992. MITA Press.

[35] R.S. Sreenivas. 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, April 1993.

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

[37] R.S. Sreenivas. On Commoner's liveness theorem and supervisory policies that enforce liveness in free-choice petri nets. *Systems and Control letters*, 1996. submitted.

[38] R.S. Sreenivas. On supervisory policies that enforce global fairness and bounded fairness in partially controlled Petri nets. *DEDS*, 1996. submitted.

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

[40] G. Vidal-Naquet. Deterministic Petri net languages. In C. Girault and W. Reisig, editors, *Application and Theory of Petri Net 1982*, volume 52 of *Informatick-Fachberichte*, New York, 1982. Springer Verlag.

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

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

[43] M.C. Zhou and F. DiCesare. *Petri net synthesis for discrete event control of manufacturing systems.* Kluwer, 1993.