Blocking and Controllability of Petri Nets in Supervisory Control

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

Frank DiCesare

Dept. of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute

Troy NY 12180-3590, USA – Email: dicesare@ecse.rpi.edu

Abstract

This note discusses the use of Petri net languages in Supervisory Control Theory. First it is shown that the trimming of an unbounded Petri net is not always possible and a new class of Petri net languages, that may be generated by nonblocking nets, is defined. Secondly, necessary and sufficient conditions for the existence of a Petri net supervisor, under the hypothesis that the system's behavior and the legal behavior are both Petri net languages, are derived. Finally, by means of an example, it is shown that Petri net languages are not closed under the supremal controllable sublanguage operator.

Published as:
A. Giua, F. DiCesare, "Blocking and Controllability of Petri Nets in Supervisory Control", *IEEE Trans. on Automatic Control*, Vol. 39, No. 4, pp. 818–823, April, 1994.

I. Introduction

This note discusses the use of Petri net languages in Supervisory Control Theory (SCT). We are interested in deriving necessary and sufficient conditions for the existence of a Petri net supervisor under the hypothesis that the system's behavior and the legal behavior are both Petri net languages.

SCT has been developed by Ramadge, Wonham, et al. [12], as a control theory for discrete event systems (DES). The mathematical framework of this approach is based on formal languages. A DES is regarded as a formal language generator and its properties are described in terms of language properties. The advantages of such an approach are clear: it is possible to obtain theoretical results whose validity does not depend on the particular model used to represent the DES.

The Supervisory Control Problem consists in designing a supervisor that restricts the traces generated by a system within a legal behavior. If the legal behavior is a *controllable language* [11] (and satisfies other closure properties) a supervisor exists. If the legal behavior is not controllable, we have to further restrict the system's behavior to the *supremal controllable sublanguage* [18] for which a supervisor exists.

In the case of systems and specifications modeled by finite state machines, i.e., by generators of regular languages, Ramadge and Wonham [18] have proved that the supervisor may also be modeled as a finite state machine. Ushio [16] has discussed under which condition a finite state machine supervisor exists in the more general case in which the system's behavior and the legal behavior are not regular.

In this note we use Petri nets (PN) as discrete event models. Petri net models have been used in the framework of SCT by Giua and DiCesare [1, 2], Holloway and Krogh [4], Kumar and Holloway [7], Sreenivas [13], Sreenivas and Krogh [14], Ushio [15], Wang [17]. Li and Wonham [5, 9] have also discussed the use of vector addition systems, that are equivalent to Petri nets, as discrete event models. The advantages of PN over simpler models, such as state machines, is given by the fact that the net structure does not require to enumerate explicitely all the states of the system and also by the fact that there often exist analysis techniques that permit an efficient solution of the control problem based solely on the net structure, without the need of an exhaustive enumeration of the state space, as shown in [3, 4].

The PN models generally used in SCT are bounded PN, a subclass of place/transition nets whose set of reachable markings is finite. Thus the class of languages generated by these models is equivalent to the class of regular languages, and all the results for regular languages may apply to them as well. Here we discuss the use of more general models with a possibly infinite set of reachable markings. The class of languages generated by these nets is a proper superset of regular languages [6, 8], thus they have a greater modeling power. The objective is that of determining whether these models may be used in SCT. Our approach is different from that of Sreenivas and Krogh [14] because we do not consider Petri nets with inhibitor arcs for the reasons that will be discussed in Example 3.1. Also in this note we do not consider the problem of strict concurrency, discussed by Li and Wonham [9] and Wang [17].

We consider three problems.

- The first problem regards the trimming of the net, i.e., the modification of its structure in order that no blocking state may be reached while preserving its marked behavior. We prove that the trimming of a PN is not always possible and we define a new class of PN languages that may be generated by nonblocking generators.
- The second problem regards the existence of a PN supervisor, i.e., a supervisor whose control action is implicit in its net structure [1, 3, 5]. The advantage of such a supervisor, as opposed to a supervisor given as a feedback function, is that a closed loop model of the controlled system may be constructed and analyzed using the same PN analysis techniques that are used to analyze the system. By suitable modification of the results given by Ramadge and Wonham [11, 12] and reported in Section II.C, we are able to derive necessary and sufficient conditions for the existence of supervisors as nets.
- The final problem regards the closure of PN languages under extraction of the supremal controllable sublanguage. In this case, by means of an example, we show that PN languages are not closed under this operator.

II. Background

A. Petri Nets

A *Place/Transition net* (P/T net) [10] is a structure $N = (P, T, I, O)$ where: P is a set of *places* represented by circles; T is a set of *transitions* represented by bars; $I : P \times T \to I\!N$ is the *input function* that specifies the arcs directed from places to transitions; $O : P \times T \to I\!N$ is the *output function* that specifies the arcs directed from transitions to places. Here $I\!N = \{0, 1, 2, \ldots\}$ and it is also assumed that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

A *marking* is a vector $M : P \to I\!N$ that assigns to each place of a P/T net a non negative integer number of tokens, represented by black dots. $M(p)$ indicates the number of tokens assigned by marking M to place p. A transition $t \in T$ is *enabled* by a marking M if $(\forall p \in P)\ [M(p) \geq I(p,t)]$. The firing of transition t generates a new marking M' with: $M'(p) = M(p) + O(p,t) - I(p,t)$. When a marking M' can be reached from marking M by executing a *firing sequence* of transitions $\sigma = t_1 \ldots t_k$ we write $M\ [\sigma\rangle\ M'$. We write $M\ [\sigma\rangle$ to denote that σ may be executed from M. The set of markings reachable on a net N from a marking M is called *reachability set* of M and is denoted as $R(N,M)$.

A P/T net N with initial marking M_0 is *bounded* if $\exists k$ such that for all $p \in P$ and for all $M \in R(N, M_0)$, $M(p) \leq k$.

A *labeled Petri net* (or *Petri net generator*) [6] is a 4-tuple $G = (N, \ell, M_0, F)$ where: $N = (P, T, I, O)$ is a Petri net 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.

The two languages associated with G are the *L-language* (called *marked behavior* in the framework of SCT) and the *P-language* (or *closed behavior*) defined in the following [6].

Given a DES $G = (N, \ell, M_0, F)$, 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 *P-type language* of G is

$$L(G) = \{\ell(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_0\ [\sigma\rangle\}.$$

Note that in this definition of labeled net we are assuming that ℓ is a λ-free labeling function, i.e., no transition is labeled with the empty string λ and two (or more) transitions may have the same label. The classes of L-type and P-type languages generated by labeled nets is denoted $\mathcal{L}$ and $\mathcal{P}$ respectively.

A *deterministic* PN generator [6] is such that the string of events generated from the initial marking uniquely determines the sequence of transitions fired. Formally, a DES $G = (N, \ell, M_0, F)$ is deterministic iff $\forall t, t' \in T$, with $t \neq t'$ and $\forall M \in R(N, M_0)$, $M\ [t\rangle \wedge M\ [t'\rangle \implies \ell(t) \neq \ell(t')$. Systems of interest in SCT are deterministic, hence we will always assume that the generators considered here are deterministic. The classes of L-type and P-type PN languages generated by deterministic PN generators are denoted, respectively, $\mathcal{L}_d$ and $\mathcal{P}_d$.
Proposition 2.1 ([6]). The following properties hold.

1. $\mathcal{P}_d \subset \mathcal{P}$; $\mathcal{L}_d \subset \mathcal{L}$; $\mathcal{P}_d \not\subseteq \mathcal{L}_d$; $\mathcal{P} \subset \mathcal{L}$.

2. $\mathcal{P}_d, \mathcal{L}_d, \mathcal{P}, \mathcal{L}$ are closed under intersection.

B. Languages and Controllability

Let L be a language on alphabet Σ. Its *prefix closure* is the set of all prefixes of strings 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 $L = \overline{L}$.

Two languages L_1 and L_2 are said to be *non-conflicting* [19] if $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$.

Let in the following G be a DES with alphabet of events Σ.

G is *nonblocking* if any string that belongs to its closed behavior may be completed to a string that belongs to the marked behavior. This implies: $L(G) = \overline{L_m(G)}$.

A language $K \subset \Sigma^*$ is said to be $L_m(G)$-*closed* if $K \cap L_m(G) = \overline{K} \cap L_m(G)$. In the case that $K \subseteq L_m(G)$, this definition is reduced to the usual definition of $L_m(G)$-closed: $K = \overline{K} \cap L_m(G)$ [12].

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 within a legal behavior, while uncontrollable events may never be disabled.

A language $K \subset \Sigma^*$ is said to be *controllable* with respect to $L(G)$ [11] if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. The set of all languages controllable wrt $L(G)$ is denoted $\mathcal{C}(G)$.

If a language $L \subset \Sigma^*$ is not controllable with respect to $L(G)$ we may compute its *supremal controllable sublanguage* [18] defined as: $L^\uparrow = \sup\{K \subseteq L \mid K \in \mathcal{C}(G)\}$.

C. Supervisor

A *supervisor* [11] is an agent that disables the controllable transitions of a system in order to restrict its behavior within a legal behavior. 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. Formally a supervisor is a map: $f : L(G) \to \Gamma$, specifying for each possible string of events $\omega \in L(G)$ generated by the system G the control input $\gamma = f(\omega)$ to be applied at that point. The supervisor may also be represented as another discrete event system, called S.

The objective is to design a supervisor that selects control inputs in such a way that the controlled system's behavior satisfies various constraints. The behaviors of the system under control are denoted: $L(S/G)$ (closed behavior) and $L_m(S/G) = L(S/G) \cap L_m(G)$

(controlled behavior). Note that we are assuming that the supervisor does not mark strings, i.e., the marked strings of the system under control are all and only the marked strings of the uncontrolled system that survive under control.

The following two theorems, due to Ramadge and Wonham [11, 12], give necessary and sufficient conditions for the existence of a supervisor.
Theorem 2.1. For nonempty $L \subseteq L(G)$ there exists a supervisor S such that $L(S/G) = L$ iff L is prefix closed and controllable.
Theorem 2.2. For nonempty $L \subseteq L_m(G)$ there exists a nonblocking supervisor S such that $L_m(S/G) = L$ iff L is $L_m(G)$-closed and controllable.

We consider the case in which both the system G and the supervisor S are given as nets. In this case S and G are supposed to run in parallel, i.e., whenever on G a transition fires, a transition with the same label is concurrently fired on S. The control input at a given moment is represented by the transitions that are enabled in S. Thus the function f is implicit in the transition structure of S. The closed behavior of the controlled system is now $L(S/G) = L(S) \cap L(G)$ and the controlled behavior is $L_m(S/G) = L(S/G) \cap L_m(G) = L(S) \cap L_m(G)$.

If the supervisor is a net, it is possible to construct a PN model of the system under control, denoted G_c, using the net counterpart of the intersection operator on languages [1]. If S and G are deterministic generators (as we always assume) G_c is deterministic as well.

III. 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 $L(G) \supset \overline{L_m(G)}$ we want to modify the structure of the net to obtain a new generator 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 transitions.

On Petri net models the trimming may be more complex. If the Petri net is bounded, we have discussed in [1] how the trimming may be done without major changes of the net structure, in the sense that we have 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 shown in [2]. There are some cases, furthermore, in which the trimming of the net is not possible. The reason for this is given by the following theorem.

Theorem 3.1. There exist L-type Petri net languages whose prefix closure is not a P-type Petri net language, i.e., $\exists L \in \mathcal{L}$ such that $\overline{L} \notin \mathcal{P}$.

Proof: See Proposition 3.1. $\diamond$

Example 3.1. Let G be the PN generator in Figure 1.(a), with $M_0 = (1000)^T$ and set of final markings $F = \{(0001)^T\}$. The behaviors of this net are: $L_m(G) = \{a^m b a^m b \mid m \geq 0\}$ and $L(G) = \{\overline{a^m b a^n b} \mid m \geq n \geq 0\}$. From the reachability tree of this net, shown in Figure 1.(b), it is clear that the system is blocking. To avoid reaching a blocking state we require 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 [14], thus we cannot properly consider these models as P/T nets.

We may formally prove that the prefix closure of the marked language of the net discussed in Example 3.1 is not a P-type Petri net language. The proof is based on the pumping lemma for P-type PN languages, given in [6].

Lemma 3.1. (*Pumping lemma*). Let $L \in \mathcal{P}$. Then there exist numbers k, l such that any string $\omega \in L$, with $\mid \omega \mid \geq k$, has a decomposition $\omega = xyz$ with $1 \leq \mid y \mid \leq l$ such that $xy^iz \in L, \forall i \geq 1$.

The pumping lemma for P-type PN languages is similar to the pumping lemma for regular languages. The difference is that in latter it is required that $xy^iz \in L, \forall i \geq 0$. As an example, given the P-type PN language $L = \{a^m b^n \mid m \geq n \geq 0\}$ it is easy to prove that the string $a^k b^k$ does not satisfy the pumping lemma for regular languages.

Proposition 3.1. $L = \{\overline{a^m b a^m b} \mid m \geq 0\}$ is not a P-type Petri net language.

Proof. Given k and l according to the pumping lemma, let $\omega = a^k b a^k b$. Consider a partition $\omega = xyz$, with $1 \leq \mid y \mid \leq l$. Clearly $b \notin y$, since in this case $xy^iz \notin L$ for $i \neq 1$. However if $\omega = \underbrace{a^n}_{x}\underbrace{a^p}_{y}\underbrace{a^q b a^k b}_{z}$, $(n+p+q=k)$, or if $\omega = \underbrace{a^k b a^n}_{x}\underbrace{a^p}_{y}\underbrace{a^q b}_{z}$, $(n+p+q=k)$, then $xy^iz \notin L$ for $i \neq 1$. Hence L is not a P-type Petri net language. $\diamond$

Theorem 3.1 is the source of major difficulties using PN in supervisory control when one considers marking and blocking (see Example 4.1). We conclude this section with the definition of 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 we will discuss in Section IV.

Definition 3.1. A language $L \in \mathcal{L}$ (not necessarily deterministic) is said to be *deterministic P-closed* (DP-closed for short) iff 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, we point 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 [6]. The proof follows from the fact that $\mathcal{L}_{DP}$ is not closed under intersection, as we show in Example 4.3, while all previously defined classes of PN languages are closed under intersection.

IV. Existence of Petri net supervisors

In this section we present some theorems that give necessary and sufficient conditions for the existence of PN supervisors.

The first two theorems regard the case in which we want to restrict the closed behavior of G within the limits of a legal behavior L.
Theorem 4.1. Let G be a PN and let $L \subseteq L(G)$ be a non empty language. There exists a PN supervisor S such that $L(S/G) = L$ iff $L \in \mathcal{P}_d \cap \mathcal{C}(G)$.

Proof. (If) Since $L \in \mathcal{P}_d$ then there exists a PN S such that $L(S) = L$. S is the desired supervisor. In fact, since $L \in \mathcal{C}(G)$ then S, running in parallel with G, will never block an uncontrollable transition of G and $L(S/G) = L(S) \cap L(G) = L$. (Only if) Since $L(S/G) = L$ then L is a P-type language for the PN representing the closed loop system and $L \in \mathcal{P}_d$. Also L is controllable by Theorem 2.1. $\diamond$

It may be interesting to consider the case in which the legal behavior L is not a subset of $L(G)$. In this case we are interested in restricting the system's behavior to $L \cap L(G)$, and we simply need to check whether $L \cap L(G)$ satisfies the conditions of the previous theorem. It may well be the case that $L \notin \mathcal{P}_d \cap \mathcal{C}(G)$ but $L \cap L(G) \in \mathcal{P}_d \cap \mathcal{C}(G)$, i.e., a PN supervisor for this problem may exist even if L is not a PN language or if L is not closed and controllable. However if $L \in \mathcal{P}_d$ we have the following theorem.
Theorem 4.2. Let G be a PN and let $L \subseteq \Sigma^*$, with $L(G) \cap L \neq \emptyset$, and $L \in \mathcal{P}_d$. There exists a PN supervisor S such that $L(S/G) = L \cap L(G)$ iff $L \in \mathcal{C}(G)$.

Proof. (If) The sets $\mathcal{P}_d$ and $\mathcal{C}(G)$ are closed under intersection, as proved, respectively, in [6, 11]. Hence $L \cap L(G) \in \mathcal{P}_d \cap \mathcal{C}(G)$ and by Theorem 4.1 there exists a PN supervisor S such that $L(S/G) = L \cap L(G)$. (Only if) Since $L(S/G) = L \cap L(G)$ then $L \cap L(G) \in \mathcal{P}_d$. Also $L \cap L(G)$ is controllable by Theorem 2.1, i.e., $[\overline{L \cap L(G)}]\Sigma_u \cap L(G) \subseteq \overline{L \cap L(G)} \subseteq \overline{L} \Longrightarrow [L \cap L(G)]\Sigma_u \cap L(G) \subseteq \overline{L}$ (since languages in $\mathcal{P}_d$ are prefix closed) $\Longrightarrow L\Sigma_u \cap L(G) \subseteq \overline{L}, \Longrightarrow \overline{L}\Sigma_u \cap L(G) \subseteq \overline{L}$ (since $L \in \mathcal{P}_d$), hence $L \in \mathcal{C}(G)$. $\diamond$

Let us consider now the case in which we want 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 2.2) are not sufficient to insure the existence of a nonblocking PN supervisor even if $L \in \mathcal{L}_d$. The following example discusses this case.

Example 4.1. Let G be the PN generator in Figure 2, with $\Sigma_u = \emptyset$, $M_0 = (100)^T$, and set of final markings $F = \{(001)^T\}$. The marked behavior of this net is: $L_m(G) = \{a^m b a^n b \mid m \geq 0, n \geq 0\}$. Assume now that we want to restrict the marked behavior of G to $L = \{a^m b a^m b \mid m \geq 0\}$ that is clearly controllable and $L_m(G)-$closed. However since $L \notin \mathcal{L}_{DP}$ there exists no PN whose P-type language is $\overline{L}$. L is the L-type language of the net E in Figure 1.(a) with set of final marking $F = \{(0001)^T\}$, that however does not qualify as a supervisor for this problem, since $L_m(E/G) = L(E) \cap L_m(G) \supset L$. As an example, abb is in $L(E) \cap L_m(G)$, but it is not in $L = L_m(E)$.

The example shows the need of introducing the further requirement that the legal behavior L be DP-closed for insuring the existence of a PN supervisor.

Theorem 4.3. 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 $L \in \mathcal{L}_{DP} \cap \mathcal{C}(G)$ and L is $L_m(G)-$closed.

Proof. (If) Since $L \in \mathcal{L}_{DP}$ then there exists a PN S such that $L(S) = \overline{L} \subseteq L(G)$. S is the desired supervisor. In fact: since $L \in \mathcal{C}(G)$, then clearly $\overline{L} \in \mathcal{C}(G)$; since L is $L_m(G)-$closed, then $L_m(S/G) = L(S) \cap L_m(G) = \overline{L} \cap L_m(G) = L$; S is nonblocking, since $L(S/G) = L(S) \cap L(G) = \overline{L} \cap L(G) = \overline{L} = \overline{L_m(S/G)}$. (Only if) Since S is a nonblocking supervisor, then $L \in \mathcal{C}(G)$ and L is $L_m(G)-$closed, by Theorem 2.2. We need to prove that $L \in \mathcal{L}_{DP}$. First note that $L = L_m(S/G) = L(S) \cap L_m(G)$. Since $L(S) \in \mathcal{P}_d$, hence $L(S) \in \mathcal{L} \supset \mathcal{P} \supset \mathcal{P}_d$ [6]. (Note that $L(S)$ although a deterministic P-type language may be a nondeterministic L-type language.) Since $\mathcal{L}$ is closed under intersection, then $L \in \mathcal{L}$. Also $\overline{L} = L(S/G)$ by the non blocking hypothesis, hence $\overline{L} \in \mathcal{P}_d$ (since it is the deterministic P-type language closed loop PN generator). It follows that $L \in \mathcal{L}_{DP}$. $\diamond$

Theorem 4.3 does not require L to be deterministic. In fact we want to restrict the marked behavior of G to L by means of a *non marking* deterministic supervisor, that effectively constrains solely the closed behavior of G. Hence it is sufficient that $\overline{L}$ be a deterministic P-type language in order to construct a deterministic supervisor. The following example may clarify this point.

Example 4.2. Let G be the PN generator in Figure 3, with $\Sigma_u = \{b\}$, $M_0 = (10)^T$, and set of final markings $F = \{(01)^T\}$. The marked behavior of this net is: $L_m(G) = \{a^m b a^n \mid m \geq 0; n \geq 0\}$. Assume now that we want to restrict the marked behavior of G to $L = \{a^m b a^n \mid m \geq n \geq 0\}$. L is a L-type PN language and it can be proved that it is not deterministic, i.e., it cannot be accepted by a deterministic generator with a finite set of final states F. However $L \in \mathcal{L}_{DP}$, since $\overline{L}$ is the deterministic P-type language of the net S in Figure 4. Since L is controllable and $L_m(G)-$closed, S is a proper supervisor and is such that $L_m(S/G) = L$.

We would like to extend Theorem 4.3 to the case in which the legal behavior L is not a subset of $L_m(G)$. Unfortunately in this case the hypotheses of Theorem 4.3 are not sufficient to insure the existence of a PN supervisor. In fact it may be possible that, although $L \in \mathcal{L}_{DP}$, $L \cap L_m(G) \notin \mathcal{L}_{DP}$.
Proposition 4.1. The class of DP-closed PN languages $\mathcal{L}_{DP}$ is not closed under intersection.

The proof of the proposition follows from the following example.
Example 4.3. Let G be the PN generator in Figure 2, $M_0 = (100)^T$, and set of final markings $F = \{(001)^T\}$. The marked behavior of this net is: $L_m(G) = \{a^m b a^n b \mid m \geq 0; n \geq 0\} \in \mathcal{L}_{DP}$. Consider the language $L = \{a^m b(bc)^n (a(bc)^p)^m b \mid m \geq 0; n \geq 0; p \geq 0\}$. Clearly $L \in \mathcal{L}_{DP}$ since it is the marked behavior of the nonblocking generator E in Figure 5, with set of final markings $F = \{(0001)^T\}$. Unfortunately $L \cap L_m(G) = \{a^m b a^m b \mid m \geq 0\} \notin \mathcal{L}_{DP}$, as we have shown in Section III.

The problem in the previous example is that L and $L_m(G)$ are *conflicting*, i.e., $\overline{L} \cap \overline{L_m(G)} \supset \overline{L \cap L_m(G)}$. Hence, while $\overline{L} \cap \overline{L_m(G)} \in \mathcal{P}_d$ (since $\mathcal{P}_d$ is closed under intersection) it may be the case that $\overline{L \cap L_m(G)} \notin \mathcal{P}_d$. If however the two languages are *not conflicting*, we have that $\overline{L \cap L_m(G)} = \overline{L} \cap \overline{L_m(G)} \in \mathcal{P}_d$.
Theorem 4.4. Let G be a nonblocking PN and let $L \subseteq \Sigma^*$, with $L_m(G) \cap L \neq \emptyset$. There exists a nonblocking PN supervisor S such that $L_m(S/G) = L \cap L_m(G)$ if $L \in \mathcal{L}_{DP} \cap \mathcal{C}(G)$, L is $L_m(G)-$closed, L and $L_m(G)$ are not conflicting.

Proof. The class $\mathcal{L}$ and the set $\mathcal{C}(G)$ are closed under intersection [6, 12]. Also, by the non-conflicting hypothesis, $\overline{L \cap L_m(G)} \in \mathcal{P}_d$. Hence $L \cap L_m(G) \in \mathcal{L}_{DP} \cap \mathcal{C}(G)$. Finally since L is also $L_m(G)-$closed we can write: $\overline{L \cap L_m(G)} \cap L_m(G) = \overline{L} \cap \overline{L_m(G)} \cap L_m(G) = \overline{L} \cap L_m(G) = L \cap L_m(G)$, i.e., $L \cap L_m(G)$ is $L_m(G)-$closed. By Theorem 4.3 there exists a nonblocking supervisor S such that $L_m(S/G) = L \cap L_m(G)$. $\diamond$

The last theorem gives a sufficient, but not necessary, condition for the existence of a PN supervisor. In fact while: $L \in \mathcal{C}(G) \Longrightarrow L \cap L_m(G) \in \mathcal{C}(G)$ and L is $L_m(G)-$closed $\Longrightarrow L \cap L_m(G)$ is $L_m(G)-$closed, the converse is not true. A necessary and sufficient condition for the existence of a PN supervisor in the case the legal behavior L is not a subset of $L_m(G)$ may be given, as a trivial corollary of Theorem 4.3, requiring that $L \cap L_m(G) \in \mathcal{L}_{DP} \cap \mathcal{C}(G)$ and that $L \cap L_m(G)$ be $L_m(G)-$closed.

V. Supremal controllable sublanguage

Our final result regards the closure of PN languages under the *supremal controllable sublanguage operator* $^\uparrow$ [18].

Proposition 5.1. The classes $\mathcal{P}_d$ 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 5.1. Let G be the PN generator in Figure 6, with $\Sigma_u = \{a\}$, $M_0 = (1000)^T$, and set of final markings $F = \{(0001)^T\}$. Consider now the net E in Figure 5 with set of final markings $F = \{(0001)^T\}$. The two languages $L(E) \in \mathcal{P}_d$ and $L_m(E) \in \mathcal{L}_{DP}$ are not controllable. To show this we have drawn, in Figure 7, the reachability tree of the two nets; since E refines G, we have represented the arcs that belong to the reachability tree of both nets with continuous lines, while the arcs that only belong to the reachability tree of G have been represented by dotted lines. $L(E)$ and $L_m(E)$ are not controllable because of the presence of the dotted arcs associated to the uncontrollable transition a. If we apply the $^\uparrow$ operator, all the states from which a dotted arc is outputting will be removed. Thus we obtain the generator in Figure 8, whose closed and marked behavior are the two supremal controllable sublanguages: $L(E)^\uparrow = \{\overline{a^m b a^m (bc)^n b} \mid m \geq 0; n \geq 0\}$ and $L_m(E)^\uparrow = \{a^m b a^m (bc)^n b \mid m \geq 0; n \geq 0\}$. $L(E)^\uparrow \notin \mathcal{P}_d$, as can be proved using the pumping lemma in the same way we have done in Section III (the string $\omega = a^k b a^k b$ may be used to prove that no pumping is possible). $L_m(E)^\uparrow \notin \mathcal{L}_{DP}$, since $\overline{L_m(E)^\uparrow} = L(E)^\uparrow \notin \mathcal{P}_d$.

VI. Conclusion

The results of this note show that although unbounded PN do not have all desirable properties from the point of view of supervisory control, there are cases in which PN supervisors may be constructed, as discussed in Section IV.

The main difficulties when using Petri nets as discrete event models for supervisory control stems out from the fact that not all L-type PN languages are DP-closed and from the fact that the class $\mathcal{L}_{DP}$ of DP-closed languages is not closed under intersection. Another undesirable property is given by the fact that the classes $\mathcal{P}_d$ and $\mathcal{L}_{DP}$ are not closed under the $^\uparrow$ operator. It may be possible to give a characterization of the subclasses of $\mathcal{P}_d$ and $\mathcal{L}_{DP}$ that are closed under the $^\uparrow$ operator, and this will be the object of further research.

References

[1] A. Giua, F. DiCesare, "Supervisory Design Using Petri Nets", *Proc. 30th IEEE Int. Conf. Decision and Control* (Brighton, England), pp. 92–97, December, 1991.

[2] A. Giua, F. DiCesare, "On the Existence of Petri Net Supervisors", *Proc. 31th IEEE Int. Conf. Decision and Control* (Tucson, Arizona), December, 1992.

[3] A. Giua, F. DiCesare, M. Silva, "Petri Nets Supervisors for Generalized Mutual Exclusion Constraints," *Proc. 1993 IFAC World Congress* (Sidney, Australia), July, 1993.

[4] L.E. Holloway, B.H. Krogh, "Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets", *IEEE Trans. on Automatic Control*, Vol. AC-35, No. 5, pp. 514–523, May, 1990.

[5] N.Q Huang, Y. Li, W.M. Wonham, "Supervisory Control of Vector Discrete-Event Systems", *Proc. 27th Allerton Conf. on Communication, Control and Computing* (Urbana-Champaign, Illinois), pp. 925–934, September, 1989.

[6] M. Jantzen, "Language Theory of Petri Nets", *Advances in Petri Nets 1986*, Lecture Notes in Computer Sciences, Vol. 254-I, pp. 397–412, Springer-Verlag, 1987.

[7] R. Kumar, L.E. Holloway, "Supervisory Control of Petri Net Languages," *Proc. 31th IEEE Int. Conf. Decision and Control* (Tucson, Arizona), pp. 1190–1195, December, 1992.

[8] S. Lafortune, H. Yoo, "Some Results on Petri net Languages", *IEEE Trans. on Automatic Control*, Vol. AC-35, No. 4, pp. 482–485, April, 1990.

[9] Y. Li, W.M. Wonham, "Strict Concurrency and Nondeterministic Control of Discrete-Event Systems", *Proc. 28th IEEE Int. Conf. on Decision and Control* (Tampa, Florida), pp. 2731–2736, December, 1989.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings IEEE*, Vol. PROC-77, No. 4, pp. 541–580, April, 1989.

[11] P.J. Ramadge, W.M. Wonham, "Supervisory Control of a Class of Discrete-Event Processes", *SIAM Jour. Control and Optimization*, Vol. 25, No. 1, pp. 206–230, January, 1987.

[12] P.J. Ramadge, W.M. Wonham, "The Control of Discrete Event Systems", *Proceedings IEEE*, Vol. PROC-77, No. 1, pp. 81–98, January, 1989.

[13] R.S. Sreenivas, "A Note on Deciding the Controllability of a Language K with respect to a Language L," *IEEE Trans. on Automatic Control*, to appear.

[14] R.S. Sreenivas, B.H. Krogh, "On Petri Net Models of Infinite State Supervisors", *IEEE Trans. on Automatic Control*, Vol. AC-37, No. 2, pp. 274–277, February, 1992.

[15] T. Ushio, "On the Controllability of Controlled Petri Nets", *Control-Theory and Advanced Technology*, Vol. 5, No. 3, pp. 265–275, September, 1989.

[16] T. Ushio, "On The Existence of Finite State Supervisors in Discrete-Event Systems", *Proc. 29th IEEE Int. Conf. Decision and Control* (Honolulu, Hawaii), pp. 2857–2860, December, 1990.

[17] F-Y. Wang, "Supervisory Control for Concurrent Discrete Event Dynamic Systems Based on Petri Nets," *Proc. 31th IEEE Int. Conf. Decision and Control* (Tucson, Arizona), pp. 1196–1197, December, 1992.

[18] W.M. Wonham, P.J. Ramadge, "On the Supremal Controllable Sublanguage of a Given Language", *SIAM Jour. Control and Optimization*, Vol. 25, No. 3, pp. 637–659, May, 1987.

[19] W.M. Wonham, P.J. Ramadge, "Modular Supervisory Control of Discrete-Event Systems", *Math. Control Signals Systems*, Vol. 1, No. 1, pp. 13–30, 1988.

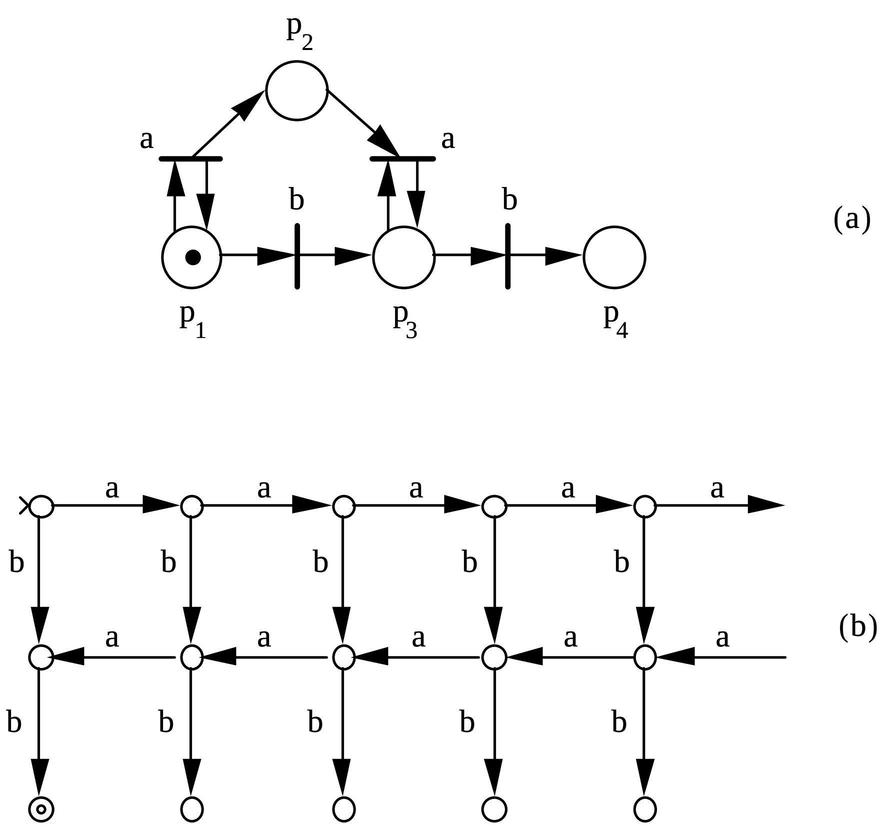

Figure 1: Blocking net in Example 3.1.

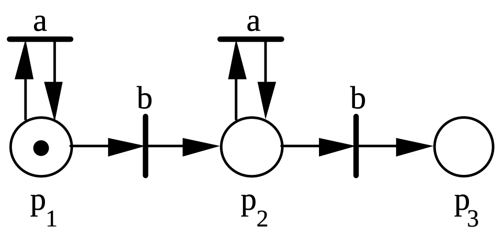

Figure 2: System net G in Example 4.1 and Example 4.3.

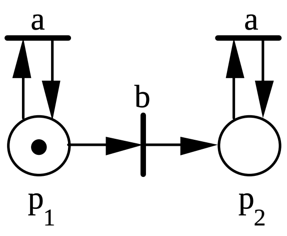

Figure 3: System net G in Example 4.2.

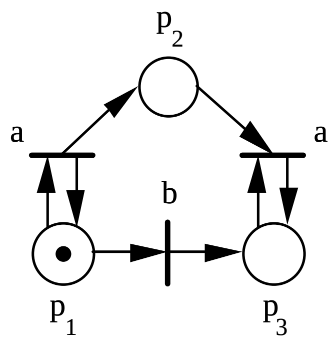

Figure 4: Supervisor S in Example 4.2.

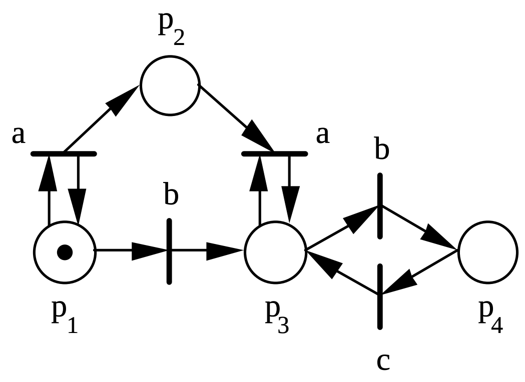

Figure 5: Generator E in Example 4.3 and Example 5.1.

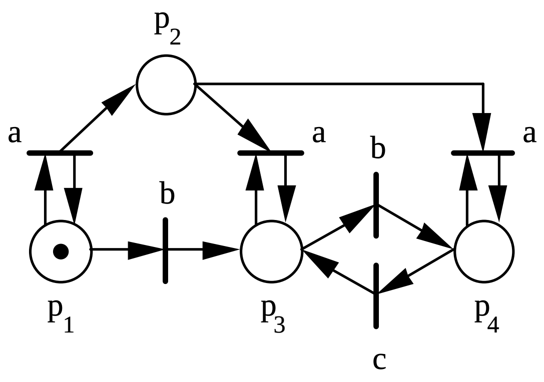

Figure 6: Generator G in Example 5.1.

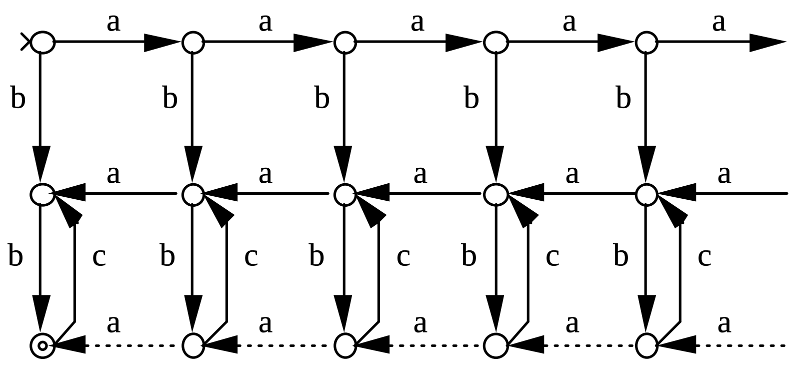

Figure 7: Reachability tree of G and E in Example 5.1.

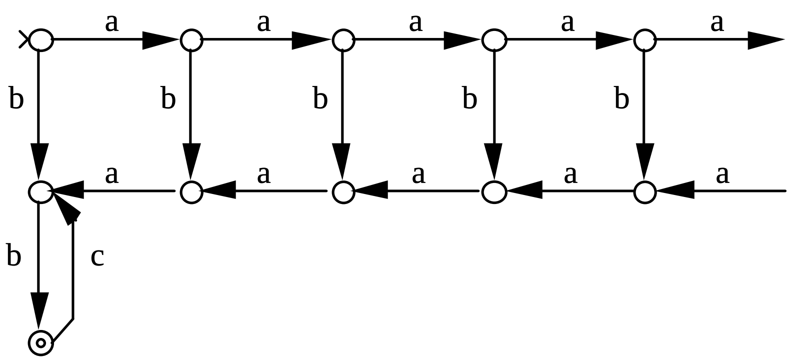

Figure 8: Generator of $L(E)^{\uparrow}$ and $L_m(E)^{\uparrow}$ in Example 5.1.