A Systems Theory View of Petri nets*

Alessandro Giua and Carla Seatzu

Dip. Ingegneria Elettrica ed Elettronica, Università di Cagliari, Italy,
{giua,seatzu}@diee.unica.it

Abstract. Petri nets are a family of powerful discrete event models whose interest has grown, within the automatic control community, in parallel with the development of the theory of discrete event systems. In this tutorial paper our goal is that of giving a flavor, by means of simple examples, of the features that make Petri nets a good model for systems theory and of pointing out at a few open areas for research. We focus on Place/Transitions nets, the simplest Petri net model. In particular we compare Petri nets with automata, and show that the former model has several advantages over the latter, not only because it is more general but also because it offers a better structure that has been used for developing computationally efficient algorithms for analysis and synthesis.

1 Introduction

The object of the study of traditional control theory have been *time-driven systems*, i.e., systems of continuous and synchronous discrete variables, modeled by differential or difference equations. However, as the scope of control theory is being extended into the domains of manufacturing, robotics, computer and communication networks, and so on, there is an increasing need for different models, capable of describing systems that evolve in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of physical events. Such systems, whose states have logical or symbolic, rather than numerical, values that change in response to events which may also be described in nonnumerical terms, are called *discrete event systems* and the corresponding models are called *discrete event models* [5].

These systems require control and coordination to ensure the orderly flow of events. As controlled (or potentially controllable) dynamic systems, discrete event systems qualify as a proper subject for control theory. Hence a fundamental issue arises: we need classes of formal models that are capable of capturing the essential

* Published as: A. Giua, C. Seatzu, "A System Theory View of Petri Nets," in *Advances in Control Theory and Applications Series: Lecture Notes in Control and Information Sciences*, Vol. 353 C. Bonivento, A. Isidori, L. Marconi, C. Rossi (Eds.), 2007.

features of discrete, asynchronous and possibly nondeterministic systems and that are endowed with efficient mathematical tools for analysis and control.

Petri nets are a family of models developed from the original model presented in 1962 by Carl Adam Petri in his doctoral dissertation: "Kommunikation mit Automaten" (Communication with Automata). The theory of Petri nets is now well established and many different Petri net models have been defined, capable of describing: logical (i.e., untimed) systems; timed systems, both deterministic and stochastic; hybrid systems.

We claim that Petri nets are a powerful discrete event model and, in fact, the interest for this model has grown, within the automatic control community, in parallel with the development of the theory of discrete event systems. In this tutorial paper the goal is not that of providing a comprehensive survey of the research in this area, but rather that of giving a flavor, by means of simple examples, of the features that make Petri nets a good model for systems theory and of pointing out at a few open areas for research.

We compare Petri nets with automata, and show that the former model has several advantages over the latter, not only because it is more general but also because it offers a better structure that has been used for developing computationally efficient algorithms for analysis and synthesis. This gives credit to our belief that the study of automata — that is an integral part of the introductory courses on discrete event systems — should always be complemented with the presentation of Petri nets.

The paper is structured as follows. In Section 2 the definition of Place/Transition net (the most well-known Petri net model) is given and its dynamic behavior is described. Section 3 deals with the modeling of physical systems with Petri nets, with an example taken from the manufacturing domain. In Section 4 the main analysis techniques pertaining to this model are discussed, with a particular focus on the techniques based on the state equation and on the reachability graph. In Section 5 we look at Petri nets as language generators and characterize the classes of languages accepted and generated by this model. In Section 6 we show that Petri nets are a generalization of automata and point out some of advantages the first model has with respect to the latter. In Section 7 we discuss how many classical control properties may be extended to the context of discrete event systems and, as an example, discuss controllability in the framework of Petri nets. Finally, in Section 8 a few areas of research that are still opened in the Petri net domain are presented.

2 Petri nets: main definitions

In this paper we consider the basic Petri net model called *Place/Transition net* (*P/T net* for short). It is a purely *logic* model that takes into account the order of occurrence of events, without associating time to them. For a comprehensive introduction to Petri nets see also the paper by Murata [27], and the books by Peterson [32] and by David and Alla [9].

2.1 Net structure

Definition 1. *A* Place/Transition net *is a structure* $N = (P, T, Pre, Post)$ *where:*

- $P = \{p_1, p_2, \ldots, p_m\}$ *is a set of* places *represented by circles;*
- $T = \{t_1, t_2, \ldots, t_n\}$ *is a set of* transitions *represented by bars;*
- $Pre : P \times T \to \mathbb{N}$ *is the pre-incidence function that specifies the weight of the arcs directed from places to transitions;*
- $Post : P \times T \to \mathbb{N}$ *is the* post-incidence function *that specifies the weight of the arcs directed from transitions to places.*

▲

Example 1. Fig. 1 shows a net $N = (P, T, Pre, Post)$ with set of places $P = \{p_1, p_2, p_3\}$, and set of transitions $T = \{t_1, t_2, t_3, t_4\}$. Here

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad Post = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix} \end{array}$$

■

The information contained in the two matrices Pre and $Post$ is often summarized in a single matrix, defined as

$$C = Post - Pre : P \times T \to \mathbb{Z} \tag{1}$$

and called *incidence matrix*. Note however that the incidence matrix does not contain the same information of Pre and $Post$, namely the structure of the net cannot be univocally determined starting from C. This is clearly illustrated in the following example.

Example 2. The incidence matrix of the net in Fig. 1 is

$$C = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & 2 & 0 \end{bmatrix} \end{array}$$

In this matrix a negative element corresponds to a pre arc, and a positive element to a post arc. Note, however, that when a transition and a place form a loop, the weight of the pre and post arc may cancel out. In this net such is the case for the loop formed by p_3 and t_4: since $C(p_3, t_4) = 0$ no information on this loop is contained in C. ■

In the following we denote as ${}^\bullet t$ the set of *input places* of transition t, namely the set of places $p \in P$ that have an arc going from p to t, and $t^\bullet$ the set of *output places* of transition t, namely the set of places $p \in P$ that have an arc going from t to p.

Analogously, ${}^\bullet p$ and $p^\bullet$ denote respectively, the set of *input* and *output transitions* of place p, namely the set of transitions $t \in T$ that have an arc going from t to p, and from p to t, respectively.

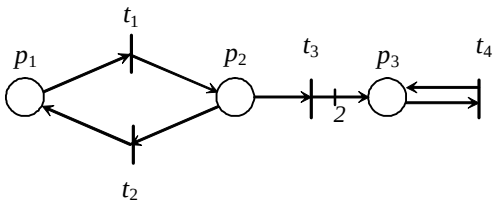

Fig. 1. A Place/Transition net.

Example 3. Let consider the net in Fig. 1. It holds ${}^\bullet t_1 = \{p_1\}$, $t_1^\bullet = \{p_2\}$, ${}^\bullet p_3 = \{t_3, t_4\}$ and $p_3^\bullet = \{t_4\}$. ■

2.2 Dynamic behavior

Definition 1 only refers to the structure of the net. To associate a dynamic behavior to it, we need to introduce the notion of *state* and to definite the *rules* that govern the occurrence of the discrete events. In particular, in the P/T framework, the state corresponds to the *marking* of the net, and the evolution corresponds to the *firing* of transitions that may occur provided that appropriate enabling conditions are verified.

Definition 2. *A* marking *is a function $M : P \to \mathbb{N}$ that associates to each place a non negative number of tokens. The initial marking is denoted M_0.* ▲

Definition 3. *A net N with initial marking M_0 is a dynamical system. It is called* net system *and is denoted as $\langle N, M_0 \rangle$.* ▲

Graphically, tokens are represented as black dots within places.

Example 4. Let us consider the net in Fig. 1. A possible initial marking is

$$M_0 = [M_0(p_1)\ M_0(p_2)\ M_0(p_3)]^T = [1\ 0\ 0]^T$$

that is shown in Fig. 2.(a). Here the only marked place is p_1, that contains one token. Another possible initial marking is $M_0 = [0\ 1\ 0]^T$ that is shown in Fig. 2.(b). Here the only marked place is p_2 that contains one token. ■

Definition 4. *A transition t is* enabled *at marking M if*

$$M \geq Pre(\cdot, t)$$

where $Pre(\cdot, t)$ denotes the column of matrix Pre relative to transition t. We write $M[t\rangle$ to denote this condition . ▲

In simple words, the enabling condition of a transition only depends on the marking of its input places. In particular, t is enabled at M if each place $p \in {}^\bullet t$ contains at least $Pre(p, t)$ tokens, i.e., place p contains a number of tokens greater or equal to the weight of the arc going from p to t.

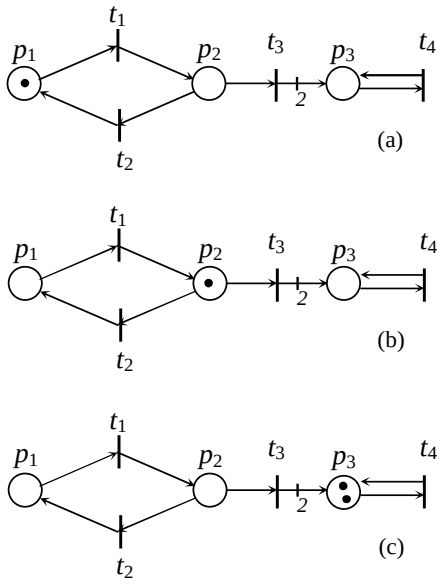

Fig. 2. Place/Transition net systems.

Example 5. Let us consider the net system in Fig. 2.(a). The only enabled transition is t_1. ■

Definition 5. *A transition t that is enabled at M may* fire*. The firing of t removes $Pre(p,t)$ tokens from each place $p \in P$ and adds $Post(p,t)$ tokens to each place $p \in P$. Thus the firing of t at M determines a new marking*

$$M' = M - Pre(\cdot,t) + Post(\cdot,t) = M + C(\cdot,t). \tag{2}$$

To denote this we write $M[t\rangle M'$. ▲

Note that, since $Pre(p,t) \neq 0$ only if $p \in^\bullet t$, and $Post(p,t) \neq 0$ only if $p \in t^\bullet$, then the firing of t at M removes $Pre(p,t)$ tokens from each input place p to t, and adds $Post(p,t)$ tokens to each output place p to t.

Moreover, by looking at Definition 5 it is immediate to observe that the enabling condition given by Definition 4 guarantees the non-negativity of the marking.

Example 6. Let us consider the net system $\langle N, M_0\rangle$ in Fig. 2.(a). If transition t_1 fires the net reaches the new marking in Fig. 2.(b) because one token is removed from p_1 and added to p_2.

Now, both transitions t_2 and t_3 are enabled. If t_3 fires, the net reaches the new marking in Fig. 2.(c) because one token is removed from p_2 and two tokens are added to p_3, being 2 the weight of the arc going from t_3 to p_3.

Now, the only enabled transition is t_4, but its firing does not change the marking being $C(p_3, t_4) = 0$. ■

Definition 6. *A sequence $\sigma = t_{j_1} t_{j_2} \dots t_{j_k} \in T^*$ is enabled at M if: t_{j_1} is enabled at M and its firing brings to a new marking M_1 that enables t_{j_2}; the firing of t_{j_2} at M_1 brings to a new marking M_2 that enables t_{j_3}, and so on.*

In such a case we write

$$M[t_{j_1}\rangle M_1[t_{j_2}\rangle \dots M_{k-1}[t_{j_k}\rangle M_k$$

or simply $M[\sigma\rangle M_k$. An enabled sequence σ is called a firing sequence. ▲

Definition 7. *A marking M is* reachable *in $\langle N, M_0\rangle$ if there exists a firing sequence σ such that $M_0[\sigma\rangle M$.*

The reachability set *of $\langle N, M_0\rangle$, denoted as $R(N, M_0)$, is the set of markings that are reachable from M_0, i.e.,*

$$R(N, M_0) = \{M \in \mathbb{N}^m \mid \exists \sigma \in T^* : M_0[\sigma\rangle M\}.$$

▲

The reachability set may never be an empty set because it always includes at least the initial marking. Moreover, it may either be finite or infinite.

Example 7. In the case of the P/T net system in Fig. 2.(a) it is easy to verify that

$$R(N, M_0) = \{[1\ 0\ 0]^T, [0\ 1\ 0]^T, [0\ 0\ 2]^T\}.$$

Consider now the P/T net system in Fig. 3. In this case the initial marking is $M_0 = [0]$ but transition t_1 has no input arcs (it is a *source* transition) hence it is always enabled and can fire as many times as desired, adding each time a token to place p. On the contrary transition t_2 is only enabled if place p is marked: its firing removes one token from p. This simple net thus describes an unbounded queueing system: the initial marking in the figure corresponds to a queue initially empty. The reachability set is thus $R(N, M_0) = \mathbb{N}$. ■

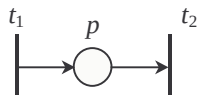

Fig. 3. The P/T net of an unbounded queueing system.

The fact that the reachability set of a P/T system may be infinite is one of the main advantages of Petri nets with respect to other discrete event models, such as automata. In fact, using Petri nets we are able to represent with a finite structure a discrete event system with an infinite number of states.

3 Modeling with Petri nets

Petri nets have been applied in a large variety of application domains, such as operational research, manufacturing systems, flexible production systems, transportation systems, and so on. The book by DiCesare *et al.* [10] provides a nice survey of Petri net approaches for the modeling and control of manufacturing systems.

In this section we first discuss the main primitives of concurrent systems that can be modeled using Petri nets. If one is interested in the order of event occurrences, the basic structures are *sequency*, *choice*, and *concurrency*. On the contrary, if one is interested in describing the use of available resources, the three most common structures are *disassembly*, *assembly*, *mutual exclusion*. Finally, we present in detail an example taken from the manufacturing domain, representing an assembly system.

3.1 Main structures

Let us consider the Petri net systems in Fig. 4. Figure 4.(a) models *sequency*. Given the initial marking, only event e_1 may occur. Then, event e_2 may only occur after the occurrence of event e_1, and event e_3 may only occur after the firing of e_2. Note that here we are talking indifferently of events and transition firings.

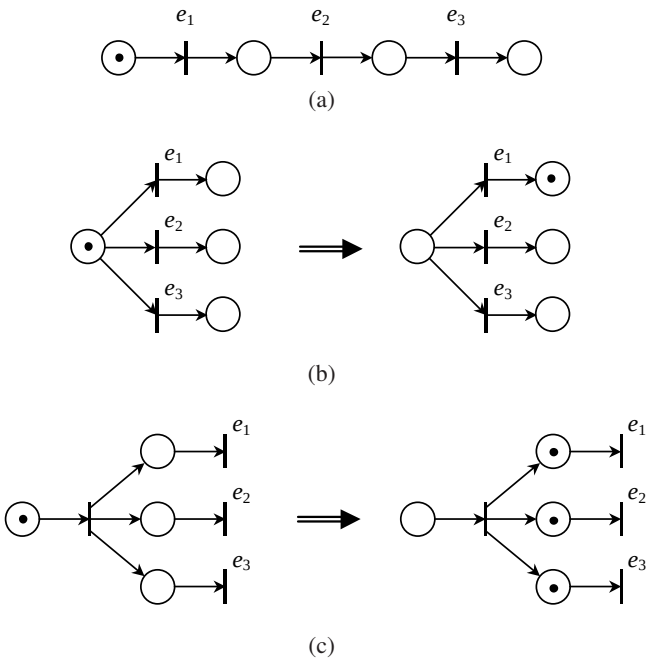

Fig. 4. Three main Petri net structures: (a) sequency, (b) choice, (c) concurrency.

Fig. 4.(b) models the *choice* among events. Given the actual marking, all the events e_1, e_2, and e_3 are enabled. However, if any of such events occurs, then the others are disabled. We also say that these events are in *conflict* among them.

Finally, Fig. 4.(c) models *concurrency*. After the firing of the only enabled transition at the initial marking, all the events e_1, e_2 and e_3 are independently enabled and may occur in any order, even simultaneously.

If tokens represent available resources, three other main structures can be defined, as summarized in Fig. 5.

Fig. 5.(a) provides an example of a *disassembly* operation. If a vehicle is disassembled, then we get 4 wheels and one chassis.

Fig. 5.(b) provides an example of an *assembly* operation. If milk, espresso and cocoa are appropriately combined, then a cappuccino is obtained.

Fig. 5.(c) models *mutual exclusion*. Assume that two machines, M_1 and M_2, share a resource, namely a robot, whose task is that of loading them. At the initial marking the robot may either load M_1 or M_2. However, if it starts loading M_1, then it is not available for M_2. It is ready to load M_2 only after it has finished to process M_1. Analogously, if it is working on M_2, it cannot load M_1 until the loading of M_2 is finished.

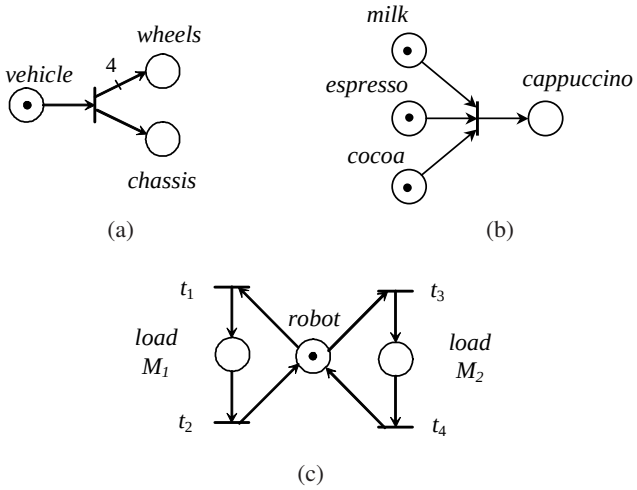

Fig. 5. Three main Petri net structures: (a) disassembly, (b) assembly, (c) mutual exclusion.

3.2 An assembly system

Let us consider the Petri net model in Figure 6, that models an assembly system [15, 10].

It consists of five machines, $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$ and $\mathcal{M}_5$ whose operational process is modeled by the firing of transitions t_1, t_2, t_3, t_4 and t_5, respectively. Two

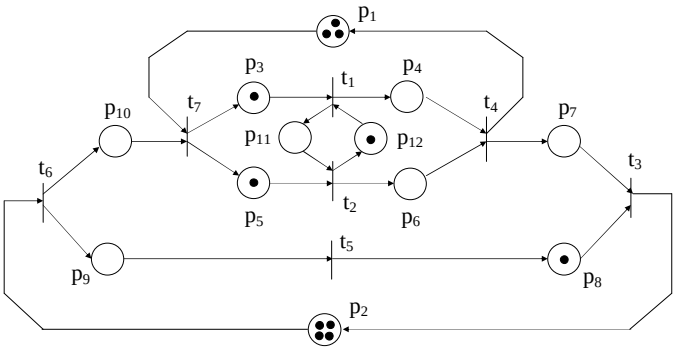

Fig. 6. The Petri net model of the assembly system in Subsection 3.2.

principal types of operations are involved in this manufacturing system: *regular operations* and *assembly operations*. Regular operations (modeled by transitions t_1, t_2 and t_5) just transform a component of the intermediate product. Assembly operations (modeled by transitions t_3 and t_4) put components together to obtain a more complex component of a final product or the final product itself.

Note that this model uses transitions (t_6 and t_7) which do not represent operations but the beginning of the manufacturing of components which are required to assemble a more complex component or the final product.

In this example there are two manufacturing levels, the primary one, performed by $\mathcal{M}_3$, leads to finite product, the secondary one, performed by $\mathcal{M}_4$, leads to semi–finished (in–working) product.

The markings of places p_1 and p_2 represent the number of assembly servers for t_4 and t_3 respectively. The marking of places p_3, p_5, and p_9 represent the availability of parts to be processed (raw materials), while the marking of places p_4, p_6, p_7 and p_8 represent the availability of semi–finished products. Places p_{11} and p_{12} ensure that machines $\mathcal{M}_1$ and $\mathcal{M}_2$ work alternatively.

4 Analysis techniques

As discussed in the previous section, P/T nets are a formal model that allows one to describe many interesting features of concurrent systems. Once a physical system has been modeled by a P/T net, the properties of interest of the system map fairly well into properties of the corresponding model. The formal definition of these properties, such as *reachability*, *boundedness*, *reversibility*, *liveness*, *deadlock-freeness*, *fairness*, etc., goes beyond the scope of this paper, but we address to [5] for a comprehensive discussion of this topic.

Many algorithms, with a well developed mathematical and practical foundation, have been developed to study these properties. The analysis techniques for Petri nets may be divided into the following groups.

- *Structural analysis*. It permits the demonstration of several properties almost independently of the initial marking. Structural analysis may be based on the study of the state equation of the net or on the study of the net graph.
- *Analysis by enumeration*. It requires the construction of the *reachability graph* representing the set of reachable markings and transition firings. If this set is not finite, a finite *coverability graph* may be constructed.
- *Analysis by transformation*. A net N_1 is transformed, according to particular rules, into a net N_2 while maintaining the properties of interest. The analysis of the net N_2 is assumed to be simpler than the analysis of the net N_1. Examples of this analysis technique are *reduction methods*, that permit the simplification of the structure of a net.
- *Simulation analysis*. It is useful to study the behavior of nets that interact with an external environment.

An extensive literature on these topics has appeared in last decades. In particular, we address to [9, 32, 34, 36] for more details. In the rest of this section, only the first two techniques will be partially described. Furthermore, we will limit our analysis to the basic *reachability problem*, that consists in establishing if a given marking is reachable starting from the initial marking.

4.1 State equation

A linear algebraic equation can be written to describe the evolution of the net system after the firing of a sequence $\sigma \in T^*$. Such equation is based on Definition 5 and on the definition of *firing vector*.

Definition 8. *Given a net N with set of transitions $T = \{t_1, t_2, \ldots, t_n\}$ and a firing sequence $\sigma \in T^*$, we call* firing vector *relative to σ, the vector $\boldsymbol{\sigma} \in \mathbb{N}^n$ whose i-th component is equal to the number of times t_i appears in σ.* ▲

Next result follows immediately from Definition 5.

Proposition 1. *Let us consider a net system $\langle N, M_0 \rangle$ with incidence matrix C. If M is reachable from M_0 firing σ, then*

$$M = M_0 + C \cdot \boldsymbol{\sigma}. \tag{3}$$

▲

Eq. (2), or sometimes its transitive closure given by Eq. (3), is called the *state equation* of $\langle N, M_0 \rangle$.

Example 8. Let us consider the net system in Fig. 2.(a) and the firing sequence $\sigma = t_1 t_2 t_1 t_2 t_1 t_3$. The firing vector associated to σ is $\boldsymbol{\sigma} = [3\ 2\ 1\ 0]^T$ and we can easily verify that the marking $M = [0\ 2\ 0]^T$ obtained from M_0 firing σ satisfies eq. (3) where C is given in Example 2. ■

It is important to stress that the state equation only provides a *necessary* (but *not sufficient*) condition for reachability. Indeed, the existence of a vector $\boldsymbol{\sigma} \in \mathbb{N}^n$ such that $M = M_0 + C \cdot \sigma \in \mathbb{N}^m$ does not imply the existence of a firing sequence σ whose firing vector is $\boldsymbol{\sigma}$, and that is enabled at M_0.

Example 9. Let us consider the net system in Fig. 7 where $M_0 = [1\ 0\ 0\ 0]^T$, and $\boldsymbol{\sigma} = [1\ 1]^T$. The marking $M = M_0 + C \cdot \boldsymbol{\sigma} = [0\ 0\ 0\ 1]^T$ is a non-negative marking however it is not reachable from M_0. In fact, no transition is enabled at the initial marking. Hence neither $\sigma' = t_1 t_2$ nor $\sigma'' = t_2 t_1$, i.e., no sequence whose firing sequence is $\boldsymbol{\sigma}$, may fire from M_0. ■

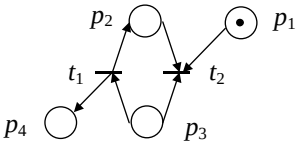

Fig. 7. The P/T system in Example 9.

4.2 Reachability graph

In this section we focus on a particular class of P/T nets, namely *bounded Petri nets*, for which the reachability problem can be solved constructing the so-called *reachability graph*.

Definition 9. *A Petri net system* $\langle N, M_0 \rangle$ *is* bounded *if and only if there exists a* finite *constant* K *such that* $\forall p \in P$ *and* $\forall M \in R(N, M_0)$, $M(p) \leq K$. ▲

Thus a Petri net system is bounded if and only if the marking of each place is bounded for any reachable marking. An obvious result is the following.

Proposition 2. *A Petri net system is bounded if and only if its reachability set is* finite. ▲

For bounded Petri net systems, it is possible to enumerate in a systematic way the reachability set by means of the *reachability graph*. Here each node corresponds to a reachable marking, and each arc corresponds to a transition. The reachability graph may be constructed using the following algorithm that terminates in a finite number of steps if the reachability set is finite.

Algorithm 1 (Reachability graph) *Let* $\langle N, M_0 \rangle$ *be a marked net with incidence matrix* C.

1. *The root node is* M_0. *This node has initially no label.*
2. *Let us consider a node* M *with no label.*

(a) For each transition t enabled at M:
i. Let $M' = M + C(\cdot, t)$.
ii. If there does not exists a node M' in the graph, add it.
iii. Add an arc t from M to M'.
(b) Label the node M "old".

3. *It there are nodes with no label, goto step 2.*
4. *Remove all labels from nodes.*

▲

Example 10. Let us consider the P/T system in Fig. 2.(a). Using Algorithm 1 we obtain the reachability graph in Fig. 8. ■

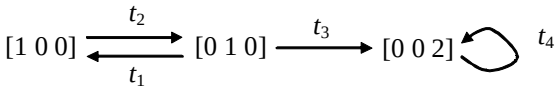

Fig. 8. The reachability graph of the P/T net system in Fig. 2.(a).

Looking at the reachability graph of a P/T system $\langle N, M_0 \rangle$, one can immediately determine which markings are reachable, because a node M is reachable from M_0 if and only if it belongs to the graph. Furthermore:

(1) a marking M' is reachable from a reachable marking M *iff* there exist two nodes M and M' in the graph and there exists an oriented path that goes from M to M';
(2) a sequence σ is firable from a reachable marking M *iff* there exists an oriented path that starts from M whose sequence of arc labels is σ.

If the reachability set is infinite, then obviously the reachability graph is infinite as well. In such a case a different algorithm can be used to compute a finite graph, called the *coverability graph*, where each arc still corresponds to a transition, while each node either corresponds to a single reachable marking, or it represents an infinite set of reachable markings. Note, however, that in such a case there is a price to pay for representing with a finite graph an infinite set: the coverability graph usually provides only necessary (but not sufficient) conditions for determining if a marking is reachable or if a sequence is firable. See [9, 32] for details.

5 Petri net languages

In the previous section we introduced the notion of reachability and highlighted the importance of characterizing the reachability set of a net system. However, the modeling power of a discrete event system is also strictly related to the sequences of

events it can generate, i.e., in the Petri net framework, to the sequences of transitions that can fire. A sequence of transitions is a string, and a set of strings is a *language*. In this section we focus on the classes of languages defined by Petri nets. In particular, we first recall the notion of *generated* and *accepted* languages, and define *labeled* Petri nets. Then, we provide the definition of L-type, G-type and P-type Petri net languages. Finally, we provide some important relationships among these classes and the class of *regular languages*.

A good introduction to Petri net languages can be found in the classic book of Peterson [32], while some generalizations and more recent results can be found in the paper by Gaubert and Giua [12]. All the material presented in this section is taken from these two references.

5.1 Generated and accepted languages

Definition 10. *The* language generated *by* $\langle N, M_0 \rangle$ *is the set of sequences that are enabled at the initial marking* M_0*, i.e.,*

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

▲

The language generated by a P/T net system is thus a *prefix-closed* language. Note that it always includes the empty word (usually denoted as ε) because for any $M \in \mathbb{N}^m$, it holds $M[\varepsilon\rangle M$.

Example 11. Let us consider the net system in Fig. 2.(a). The language of this net can be easily described with a regular expression as

$$L(N, M_0) = (t_1 t_2)^*[\varepsilon + t_1 + t_1 t_3 t_4^*].$$

This means that the sequence $t_1 t_2$ may fire indefinitely from the initial marking. Then, either no other sequence fires, or it fires t_1, or it fires the sequence $t_1 t_3$: at this point the only enabled transition is t_4 that can fire indefinitely. ■

Definition 11. *Let us consider a P/T system* $\langle N, M_0 \rangle$*. Let* F *be a set of* final *(or* accepting*) markings. The* language accepted *by* $\langle N, M_0 \rangle$ *is the set of sequences that are enabled at the initial marking* M_0 *and that lead to a marking* $M \in F$*, i.e.,*

$$L_F(N, M_0) = \{\sigma \in T^* \mid (\exists M \in F)\ M_0[\sigma\rangle M\}.$$

▲

Depending on the final set F, the language accepted by a P/T net system may not be prefix-closed. Moreover, it includes the empty word if and only if $M_0 \in F$.

Example 12. Let us consider the net system in Fig. 2.(a). Assume $F = \{[0\ 1\ 0]^T\}$. The language accepted by $\langle N, M_0 \rangle$ is $L_F(N, M_0) = (t_1 t_2)^* t_1$. ■

5.2 Labeled P/T nets

When observing the evolution of a net, it is common to assume that each transition t is assigned a label $\ell(t)$ and that the occurrence of t generates an observable output $\ell(t)$. This leads to the definition of labeled nets.

Definition 12. *Given a Petri net N with set of transitions T, a* labeling function *$\ell : T \to \Sigma$ assigns to each transition $t \in T$ a symbol from a given set of labels Σ, that may also include the empty string ε.*

*A Σ-*labeled *Petri net system is a 3-tuple $G = \langle N, M_0, \ell \rangle$ where $N = (P, T, Pre, Post)$, M_0 is the initial marking, and $\ell : T \to \Sigma$ is the labeling function.* ▲

Also in the case of labeled P/T nets we can distinguish among generated and accepted language. In particular, the following definitions hold.

Definition 13. *The* language generated *by a Σ-labeled P/T net system $\langle N, M_0, \ell \rangle$ is the ℓ-image of the set of firing sequences that are enabled at M_0, i.e.,*

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

▲

Definition 14. *Let us consider a Σ-labeled P/T net system $\langle N, M_0, \ell \rangle$. Let F be a set of final markings. The* language accepted *by $\langle N, M_0, \ell \rangle$ is the ℓ-image of the set of firing sequences leading to a final marking, i.e.,*

$$L_F(N, M_0, \ell) = \{\ell(\sigma) \mid \sigma \in T^*, (\exists M \in F)\ M_0[\sigma\rangle M\}.$$

▲

Example 13. Let us consider again the net system in Fig. 2.(a). Assume $\ell(t_1) = \ell(t_4) = a$, $\ell(t_2) = \ell(t_3) = b$. Then $L(N, M_0, \ell) = (ab)^*[\varepsilon + a + aba^*]$. Moreover, if $F = \{[0\ 1\ 0]^T$, the accepted language is $L_F(N, M_0, \ell) = (ab)^* a$. ■

5.3 Classes of languages

Different classes of *accepted* Petri net languages may be defined depending on the set of final markings F and on the labeling function ℓ [32].

Definition 15. *The accepted language of a Petri net system $\langle N, M_0 \rangle$ with set of accepting markings F, can be classified as follows.*

- *L-*type*: $L_F(N, M_0)$ is an L-*type Petri net language *if the set of final markings F is* finite*.*
- *G-*type*: $L_F(N, M_0)$ is a G-*type Petri net language *if the set of final markings F is the covering set of a given* finite *set $\bar{F}$. This means that a marking M is final if and only if $M \geq \bar{M}$ for a given $\bar{M} \in \bar{F}$. Languages in this class are usually called* weak *languages.*

- P-type*: $L_F(N, M_0)$ is a* P-type Petri net language *if the set of final markings F coincides with the reachability set $R(N, M_0)$. In such a case the accepted language is equal to the generated language and it is obviously prefix-closed.* ▲

Moreover, four classes of labeling functions may be defined.

Definition 16. *The labeling function of a labeled Petri net system $\langle N, M_0, \ell\rangle$ can be classified as follows.*

- free*: if all transitions are labeled distinctly, namely a different label is associated to each transition, and no transition is labeled with the empty string.*
- deterministic*: if no transition is labeled with the empty string, and the following condition*[2] *holds: for all $t, t' \in T$, with $t \neq t'$, and for all $M \in R(N, M_0)$: $M[t\rangle \wedge M[t'\rangle \Rightarrow [\ell(t) \neq \ell(t')]$ i.e., two transitions simultaneously enabled may not share the same label. This ensures that the knowledge of the firing labels $\ell(\sigma)$ is sufficient to reconstruct the marking M that the firing of σ yields.*
- λ-free*: if no transition is labeled with the empty string*[3].
- arbitrary*: if no restriction is posed on the labeling function ℓ.* ▲

Each of these type of labeling is a generalization of the previous one. Furthermore all types of labeling only depend on the structure of the net, but for the deterministic labeling, that depends both on the structure and on the behavior of the net.

Example 14. Let us consider the nets in Fig. 9. If we only look at the net structure — that is the same in both nets — we can say that the labeling is λ-free. However, in the first net the labeling is also deterministic because the two transitions labeled a can never be simultaneously enabled from any reachable marking. The second net is nondeterministic, because the two transitions labeled a can be simultaneously enabled.

Assume that the string aa is observed in the second net. The first a is certainly due to the occurrence of transition t_1, the only one enabled at M_0, whose firing yields the new marking $M = [1\ 1\ 0]^T$. From this marking, however, both t_1 and t_2 are enabled and one cannot determine if the second a yield $M = [0\ 2\ 0]^T$ or $M = [1\ 0\ 1]^T$. ■

Twelve different classes of Petri net languages result from the cross product of the three types of final marking sets in Definition 15 and the four types of labeling in Definition 16, as summarized in Table 1.

Here the classes of L-type, G-type, and P-type λ-free languages are denoted, respectively, $\mathcal{L}$, $\mathcal{G}$, and $\mathcal{P}$. An additional superscript f, det or λ denotes, respectively, the corresponding classes of free, deterministic, and arbitrary languages.

[2] A looser condition is sometimes given: for all $t, t' \in T$, with $t \neq t'$, and for all $M \in R(N, M_0)$: $M[t\rangle \wedge M[t'\rangle \Rightarrow [\ell(t) \neq \ell(t')] \vee [Post(\cdot, t) - Pre(\cdot, t) = Post(\cdot, t') - Pre(\cdot, t')]$. Thus two transitions with the same label may be simultaneously enabled at a marking M, if the two markings reached from M by firing t and t' are the same.

[3] In the Petri net literature the empty string is denoted λ, while in the formal language literature it is denoted ε. In this paper we denote the empty string ε but, for consistency with the Petri net literature, we still use the term *λ-free* for a non erasing labeling function $\ell : T \to \Sigma$.

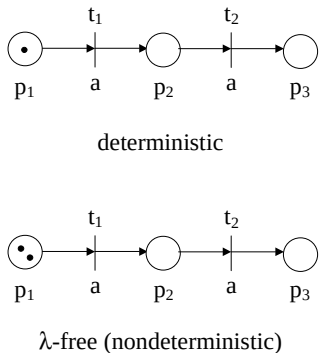

Fig. 9. A deterministic labeled net (a) and a nondeterministic one.

	free	deterministic	λ-free	arbitrary
$\mathcal{L}$-type	$\mathcal{L}^f$	$\mathcal{L}^{det}$	$\mathcal{L}$	$\mathcal{L}^\lambda$
$\mathcal{G}$-type	$\mathcal{G}^f$	$\mathcal{G}^{det}$	$\mathcal{G}$	$\mathcal{G}^\lambda$
$\mathcal{P}$-type	$\mathcal{P}^f$	$\mathcal{P}^{det}$	$\mathcal{P}$	$\mathcal{P}^\lambda$

Table 1. The 12 classes of Petri net languages.

5.4 Relationships among classes of Petri net languages

The above classes of Petri net languages are closely related. In particular, some intuitive relationships hold:

$$\begin{array}{c} \mathcal{L}^f \subsetneq \mathcal{L}^{det} \subsetneq \mathcal{L} \subsetneq \mathcal{L}^\lambda, \\ \mathcal{G}^f \subsetneq \mathcal{G}^{det} \subsetneq \mathcal{G} \subsetneq \mathcal{G}^\lambda, \\ \mathcal{P}^f \subsetneq \mathcal{P}^{det} \subsetneq \mathcal{P} \subsetneq \mathcal{P}^\lambda, \end{array} \tag{4}$$

where the symbol $\subsetneq$ denotes strict inclusion.

Note that as a consequence of the strict inclusions (4), it is not possible to provide determinization procedures to convert a nondeterministic Petri net (namely a Petri net with an arbitrary labeling function) into an equivalent deterministic Petri net. On the contrary, this is possible with finite state automata where a systematic approach exists to convert a nondeterministic finite state automaton into an equivalent deterministic one [5].

Another quite intuitive relationship is the following

$$\mathcal{P}^f \subsetneq \mathcal{G}^f, \qquad \mathcal{P}^{det} \subsetneq \mathcal{G}^{det}, \qquad \mathcal{P} \subsetneq \mathcal{G}, \qquad \mathcal{P}^\lambda \subsetneq \mathcal{G}^\lambda. \tag{5}$$

In fact, every P-type language is a G-type language if F is a singleton containing the null marking.

Other less intuitive relationships have also been proved and can be summarized graphically as in Fig. 10. Here for sake of simplicity we use $\rightarrow$ to denote $\subsetneq$, i.e.,

$$\begin{array}{ccccccc}
\mathcal{L}^f & \rightarrow & \mathcal{L}^{det} & \rightarrow & \mathcal{L} & \rightarrow & \mathcal{L}^\lambda \\
 & & & & & & \uparrow \\
\mathcal{G}^f & \rightarrow & \mathcal{G}^{det} & \rightarrow & \mathcal{G} & \rightarrow & \mathcal{G}^\lambda \\
\uparrow & & \uparrow & & \uparrow & & \uparrow \\
\mathcal{P}^f & \rightarrow & \mathcal{P}^{det} & \rightarrow & \mathcal{P} & \rightarrow & \mathcal{P}^\lambda
\end{array}$$

Fig. 10. Relationships among classes of Petri net languages.

$A \rightarrow B$ is equivalent to $A \subsetneq B$. Note that classes that are unrelated in the table (such as $\mathcal{L}^{det}$ and $\mathcal{G}^{det}$, or such as $\mathcal{L}^{det}$ and $\mathcal{P}^{det}$) are not comparable.

This plethora of Petri net languages may generate some confusion, even more considering the fact that additional classes can be defined as mentioned in [12]. Note, however, that not all these classes are useful in practice. In fact, the classes of free languages are very restricted, in the sense they do not contain all regular languages. On the contrary, for the largest classes of λ-free or arbitrary languages the problems of language equivalence or inclusion is not *decidable*. Thus we may conclude that the only interesting classes of Petri net languages are the deterministic ones [12], and we will consider them as representative of Petri net languages.

5.5 Relationships among Petri net languages and regular languages

One of the classes of formal languages that has received most attention in the literature, is the class of *regular languages* [23] that we denote as $\mathcal{R}$. Regular languages are characterized by regular expressions and are generated by regular grammars. Moreover, it has been proved that the class of regular languages is coincident with the class of languages accepted by finite state automata.

The following important result expresses the most important relationship among Petri net languages and regular languages.

Theorem 2. [12] *The intersection of the classes of L-type and G-type regular Petri net is the class of regular languages, i.e.,* $\mathcal{R} = \mathcal{G}^{det} \cap \mathcal{L}^{det}$. ▲

Therefore, $\mathcal{L}^{det}$ and $\mathcal{G}^{det}$ provide proper and distinct extensions of regular languages.

Other interesting relationships among Petri net languages and other classes of languages, such as *contex-free* languages, *bounded contex-free* languages, *context-sensitive* languages, have been proved and are reported in Fig. 11. Here we can see that Petri net languages are a subclass of context-sensitive languages, and a superclass of regular languages. Petri net languages are not comparable with context-free languages.

6 Comparison with automata

The language analysis in the previous session shows that Petri nets are a generalization of automata. In this section we want to focus on the relationship between P/T

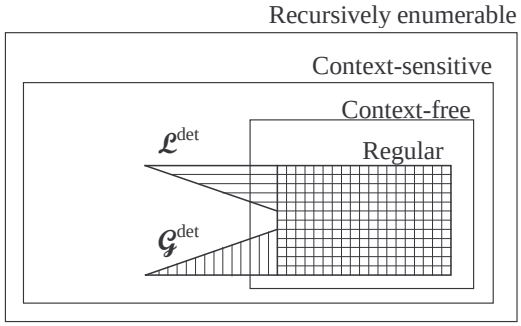

Fig. 11. Relationships among classes of formal languages.

nets and automata and show what are the main advantages the former model offers with respect to the latter. Five different aspects will be considered: the *state representation power*, the *language power*, the *modularity*, the *structural representation of primitives*, and the *linear algebraic structure*.

6.1 State representation power

A Petri net is a finite state automaton additionally equipped with *weak counters*, i.e., with the possibility of testing if a counter has reached a fixed value:

$$M(p) \geq k?$$

Example 15. Let us consider the net in Fig. 12. Place p is the counter, whose value is increased by the firing of t_1, and decreased by the firing of t_2. If there are k or more tokens in p, transition t_3 is also enabled and may fire (test of the counter) without changing the value of the counter. ■

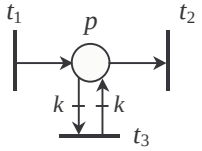

Fig. 12. A weak counter.

It is important to stress that places in a P/T net are weak counters, i.e., may be tested only for inequalities of the type $\geq$ while a test for $\leq$ is not allowed. In fact, from the enabling rule in Definition 4 follows this obvious result.

Monotonicity property. *If something can happen from M it can also happen from any marking greater than M, i.e., for any sequence $\sigma \in T^*$:*

$$M[\sigma\rangle \quad \textit{and} \quad M' \geq M \quad \Longrightarrow \quad M'[\sigma\rangle.$$

This property can be violated adding an *inhibitor arc* that allows a transition to fire only if a place is empty, thus testing a counter for zero [37]. However, this feature increases the modeling power — and the analysis complexity — of Petri nets to that of a Turing machine, making most properties of interest undecidable: we cannot properly consider these models as P/T nets.

6.2 Language power

A Petri net is a generator of regular languages with the additional feature of generating *one-sided Dyck languages*, i.e., of testing if a string of parenthesis

$$(((\,)\,)\,(\,)\,((\cdots$$

is well formed [31].

Example 16. Let us consider the net in Fig. 3. Here, the firing of t_1 corresponds to the opening of a parenthesis "(", while the firing of t_2 corresponds to the closing of a parenthesis ")". All firing sequences generated by this net correspond to well formed strings of parenthesis. ■

6.3 Modularity

With modular synthesis, complex systems may be constructed by aggregation of simpler modules. The most common operator that allows to automatically construct the model of a complex system from the models of the subsystems that compose it, is the *concurrent composition operator*, that can be defined both for automata and Petri nets.

There are, however, two main advantages in using Petri nets rather than automata.

- When applying the concurrent composition operator to Petri nets, the structure of the modules is kept in the composed net.
- The composition of k automata, each with a state space Q_i of cardinality n, yields a composed model with state space $Q \subseteq Q_1 \times \cdots \times Q_k$, i.e., the composed automata has a state space of cardinality up to n^k (exponential growth). On the contrary, the composition of k Petri nets, each with set of place P_i of cardinality m yields a composed model with set of places $P = P_1 \cup \cdots \cup P_k$, i.e., the composed net has a set of places of cardinality $k \cdot m$ (linear growth).

Example 17. Let us consider the automata in Fig. 13.(a) that represent two machines with state space Q' and Q'' respectively. Here event t_2 is shared between the two modules and their concurrent composition is shown in Fig. 13.(b). Note that each state of the new automaton is a pair $(q', q'') \in Q' \times Q''$. The structure of the two modules is lost in the composed system in Fig. 13.(b), in the sense that it is not possible to partition its structure into two parts, each corresponding to one of the two modules.

In Fig. 13.(c) we have represented the P/T net models of the two machines, whose concurrent composition is given by the net in Fig. 13.(d). Note that the composed model is obtained by the modules simply fusing the transitions with the same label. The set of places of the composed net is the union of the set of places of the modules, whose structure can still be clearly identified in Fig. 13.(d). ■

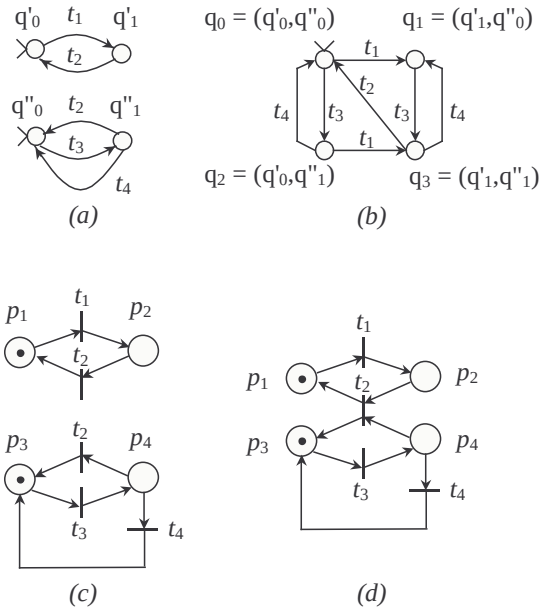

Fig. 13. The automata models of two machines (a) and their concurrent composition (b); the Petri net models of two machines (c) and their concurrent composition (d).

6.4 Structural representation of primitives

In Section 3.1 we have discussed several primitives that can be represented by Petri nets, such as "sequency", "choice", "concurrency". Each of these primitives correspond to a clear Petri net structure: sequency corresponds to a path in the graph, choice to a place inputting to more than one transition, concurrency to parallel transitions.

In the case of automata concurrency may not be represented, because an automaton can only describe the interleaving of events and not their simultaneous occurrence. However, one may think that the other primitives can be well described by structures similar to those described in Section 3.1. Here we point out that this is not always true with a simple example.

Example 18. Let us consider again the system composed by two machines whose automaton and Petri net model are shown in Fig. 13.(b) and Fig. 13.(d).

In the automaton structure we identify the path $q_0 - t_1 - q_1 - t_3 - q_3$. *Can we conclude that events t_1 and t_3 are in a sequency relation?*

In the automaton structure, from state q_0 both event t_1 and t_3 are enabled. *Can we conclude that events t_1 and t_3 are in a choice relation?*

The answer to both questions is no: transitions t_1 and t_3 are concurrent as can be seen from the Petri net model. In fact, the two transitions belong to different subsystems and can fire concurrently when both are enabled. ■

6.5 Linear algebraic structure

One of the main advantages of Petri nets is that the state is a *vector of non-negative integers*, while it is usually non numerical in other discrete event models, such as automata.

Example 19. Let us consider again the system composed by two machines whose automaton and Petri net model are shown in Fig. 13.(b) and Fig. 13.(d). State q_0' (resp., q_0'') denotes that the first (resp., second) machine is idle; state q_1' (resp., q_1'') denotes that the first (resp., second) machine is working.

In the Petri net a state is represented by a non-negative vector. Marking $[1\ 0\ 1\ 0]^T$ corresponds to the state in which both machines are idle; the marking $[0\ 1\ 1\ 0]^T$ corresponds to the state in which the first machine is working and the second is idle, and so on. Using a Petri net model the state space of this system, that is a series of labels with no algebraic structure, can be described by a set of vectors, i.e., by a highly structured set.

This also allows to describe logical specifications in a numerical form. Assume for instance, that we want to impose that the first machine should never be working if the second machine is idle. Using the notation in Fig. 13 such a constraint can be imposed forcing the constraint $M(p_2) + M(p_3) \leq 1$. ■

The possibility offered by Petri nets to describe the state space of a discrete event system that may have absolutely no algebraic structure, with a set of integers vectors has an important implication. In fact, it is possible to apply algebraic formalisms such as integer programming for the analysis and control of these systems. Within this area of research that, as we mentioned before, is called *structural analysis*, several well-founded formal approaches have been developed. Unfortunately a survey of this area is still missing, and we cannot provide comprehensive references; see however [22, 36] for a few interesting examples.

7 Mapping classical properties into discrete event systems

Classical control theory deals with *time-driven* systems modeled by difference or differential equations. However, many properties of dynamical systems have been defined in very general terms that are *model independent*.

It seems natural to study these properties in the context of *discrete event systems*, and more specifically in the context of Petri nets. This rather standard approach has been used by many researchers, and it has proved to be extremely fruitful, inspiring many of the current research areas in Petri nets.

It is important, however, to point out that the extension from time-driven to discrete event systems must be taken with care. To give a flavor of the problems one may face, in this section we discuss the classical property of *controllability* and the way in which it has been handled in the framework of Petri nets.

Note that much of this discussion is essentially due to Murata [26]. As far as we know, his 1977 work was the first paper dealing with Petri nets published in an IEEE journal. The fact that this paper was published on an Automatic Control journal is emblematic of the appeal that the algebraic structure of Petri nets has to control engineers.

7.1 Controllability

In Subsection 4.1 we introduced the state equation of a Petri net system, that can be rewritten as

$$M_{k+1} = M_k + C \cdot \boldsymbol{\sigma}_k \tag{6}$$

where $\boldsymbol{\sigma}_k \in \{0,1\}^n$ is the firing vector relative to the transition that has fired at the marking M_k, thus leading to the new marking M_{k+1}.

This clearly reminds one of the state equation of a *discrete-time linear stationary time-driven* system, namely

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k \tag{7}$$

where $\mathbf{x}_k \in \mathbb{R}^m$ (resp., $\mathbf{x}_{k+1} \in \mathbb{R}^m$) is the state vector at the sampling time k (resp., $k+1$), and $\mathbf{u}_k \in \mathbb{R}^n$ is the control input vector at sampling time k. More precisely, equation (6) is a particular case of (7) with $A = I$ and $B = C$, where I denotes the identity matrix.

As well known, the following definition of controllability holds.

Definition 17. *A discrete-time linear stationary system is* controllable *if and only if it is possible, by appropriately acting on the input, to transfer the state of the system from* any *initial state* $\mathbf{x}_0$ *to* any *other state* x_f*, called the* target state*, in a finite number of sampling steps* $f \geq 0$*.* ▲

Theorem 3. *Given the discrete-time linear stationary system (7), we call* controllability matrix *the* $(m \times n \cdot m)$ *matrix*

$$\Gamma = \left[B \middle| AB \middle| A^2B \middle| \ldots \middle| A^{m-1}B \right].$$

A necessary and sufficient condition for the controllability of (7) is that

$$n_c \triangleq \textit{rank}\ \Gamma = m.$$

▲

Therefore, using Theorem 3, the controllability matrix of a Petri net is

$$\Gamma = \left[C | C | C | \ldots | C \right],$$

thus

$$\text{rank } \Gamma = \text{rank } C,$$

i.e., the rank of the controllability matrix always coincides with the rank of the incidence matrix.

We now observe that:

- the condition rank $\Gamma = m$ is only a *necessary* condition for controllability if we restrict the control input to $\mathbf{u}_k \in \{0,1\}^m$ and to $M_k + C \cdot \boldsymbol{\sigma}_k \geq \mathbf{0}$;
- moreover, as already discussed above (see Example 9), the state equation of a Petri net system only provides a *necessary* condition for reachability.

Therefore, as a result of this analysis, the following conclusion may be drawn: *in the Petri net framework, rank $\Gamma = m$ only provides a* necessary *(but* not sufficient*) condition for controllability*.

This conclusion is not surprising — discrete event systems are much more difficult to study than linear systems — but, as Murata observes, does not address the real issue. What in fact is totally missing from this analysis is a discussion of how significant for a discrete event system is the property of controllability that derives from Definition 17. In fact, this classical notion does not fit well with discrete event systems, and it is hardly meaningful. As an example, consider a Petri net model of a manufacturing system where the marking of a place denotes the availability of a resource. It is not meaningful to investigate if the marking of such a place may reach *any* value starting from *any* other marking. As a trivial example, in an assembly system described by a Petri net starting from a state in which there are only *two* wheels available, it may be possible to reach state in which *one* bicycle has been assembled, but not a state in which *ten* bicycles have been assembled.

It seems thus natural to introduce different notions of controllability, more suited to describe the desired properties of discrete event systems. Here are some possible examples.

- Given a Petri net with incidence matrix C of dimension $m \times n$, we say that $\mathbf{x} \in \mathbb{Z}^m$ is a *P-flow* if $\mathbf{x}^T \cdot C = \mathbf{0}$.
  A P-flow imposes an invariant law on the reachability set of a net: in fact if marking M is reachable from the initial marking M_0 it must hold

  $$\mathbf{x}^T \cdot M = \mathbf{x}^T \cdot M_0$$

  as can be seen multiplying the state equation (3) by $\mathbf{x}^T$ from the left.
  This condition, however, is necessary but not sufficient for reachability. Given a net system $\langle N, M_0 \rangle$ with incidence matrix C, let X be a matrix whose columns are P-flows forming a basis of the left-null space of matrix C. It holds

  $$R(N, M_0) \subseteq I_X(N, M_0) \stackrel{\text{def}}{=} \{M \in \mathbb{N}^m \mid X^T \cdot M = X^T \cdot M_0\}.$$

Thus a meaningful definition for a Petri net system may be the following: *a Petri net system* $\langle N, M_0 \rangle$ *is controllable if* $R(N, M_0) = I_X(N, M_0)$.

- Yet a different definition of controllability may be given for *timed* Petri nets, namely P/T nets in which a *time interval* is associated to transitions: an enabled transition may fire provided that it has been enabled for a time that belongs to its time interval. Such a model is particularly useful when making performance analysis. Clearly imposing a timing structure over a logical model influences its reachability set. In fact, since a timed model can be seen as a logical model with additional timing constraints, the reachability set of the timed net is usually a subset of that of the underlying untimed one. One may define a timed Petri net system controllable if its reachability set coincides with that of the underlying untimed model.
- Finally, in Supervisory Control — one of the most interesting approaches to the control of discrete event systems — controllability is not defined as a property of a system alone, but is defined with respect to a given *specification*, i.e., with respect to a set of legal states or to a set of legal words. This definition has often been used to define a Petri net system controllable if its evolution can be restricted to a given set of legal markings or to a given set of legal words.

8 Current research areas in Petri nets

In the last two decades a large number of researchers from the automatic control community have devoted their effort to the study of Petri nets. There are, however, a certain number of basic problems that are still open. Here, we mention the following four significant areas of on-going research.

- *Control*: as in classical control, the control problem consists in finding a control law that constraints the controlled system behavior to satisfy a given specification.
- *Deadlock*: a deadlock represents an anomalous state from which no further evolution is possible. This is an issue that appears in most practical applications, and appropriate strategies should be adopted in order to prevent it.
- *Observability*: the problem is that of determining efficient ways of reconstructing the state of a net based on observed events occurrence and/or on partial marking observation.
- *Identification*: this problem consists in determining a Petri net system starting from examples/counterexamples of its language, or from the structure of its reachability (or coverability) graph.

In the following we recall the main results that have been proposed in the above areas.

8.1 Control

The most interesting and original approach to the control of discrete event systems, that has directly or indirectly shaped much of the research in this area, is *Supervisory*

Control Theory (SCT), originated by the work of P.J. Ramadge and W.M. Wonham [33]. According to the paradigm of SCT, a discrete event system G is a language generator whose behavior, i.e., language, is denoted $L(G)$. Given a legal language K, the basic control problem is to design a supervisor that restricts the closed loop behavior of the plant to $K \cap L(G)$, disabling *controllable* events; the events whose occurrence cannot be disabled are called *uncontrollable*. It is also usually required that the closed loop system satisfies additional qualitative specifications, such as absence of blocking, reversibility, etc. Since Petri nets can be seen as language generators, it is also possible to use them as discrete event models for SCT; in this case it is assumed that some transitions, that we call controllable, can be disabled by an external agent. See [5, 20] for a review of this topic.

A similar approach can also be taken when considering the state evolution of a discrete event system, rather than the traces of events it generates. This approach, that we call *state-based*, is particularly attractive when Petri nets are used to represent the plant and was used by several authors, as reviewed in [20]. Let us consider a Petri net system $\langle N, M_0 \rangle$ with m places, whose set of reachable markings is $R(N, M_0)$. Assume we are given a set of legal markings $\mathcal{M} \subseteq \mathbb{N}^m$: the basic control problem consisting in designing a supervisor that restricts the reachability set of plant in closed loop to $\mathcal{M} \cap R(N, M_0)$, while satisfying some qualitative properties of interest.

Of particular interest are those Petri net state-based control problems where the set of legal markings $\mathcal{M}$ is expressed by one — or more — linear inequality constraints called Generalized Mutual Exclusion Constraints (GMEC) [13]. In this case we write $\mathcal{M}(\mathbf{w}, k) = \{M \in \mathbb{N}^m \mid \mathbf{w}^T M \leq k\}$ to denote that $\mathcal{M}$ is expressed by the GMEC $(\mathbf{w}, k)$ with $\mathbf{w} \in \mathbb{Z}^m, k \in \mathbb{Z}$. Problems of this kind have been considered by several authors and this special structure of the legal set has the advantage that the supervisor for this class of problems takes the form of a place, called *monitor*, which has arcs going to and coming from some transitions of the plant net. The plant and the controller are both described by a net in order to have a useful linear algebraic model for control analysis and synthesis. Moreover the synthesis is not computationally demanding since it involves only a matrix multiplication. See [22] for a recent survey.

The use of Petri nets for the control of discrete event systems is still an active research area and we believe that it will continue to remain central during the next decade. In fact, there exist many interesting Petri net analysis tools — as an example, the partial order based techniques, such as *unfolding* — whose applicability to control is still largely unexplored.

8.2 Deadlock

Deadlock is a major issue to be addressed when designing a supervisory controller. A Petri net is said to be *deadlocked* if no transition is enabled. Clearly, this is an undesirable condition in quite all real applications. This is the reason why this problem has been largely investigated in the literature, particularly within the area of flexible manufacturing systems.

Deadlock problems may be seen from two different prospectives: deadlock *prevention* refers to static policies — usually coded in the net structure — for eliminating deadlocks, whereas deadlock *avoidance* refers to dynamic policies applied on-line.

The first significant contribution in this area dates back to 1990 and is due to Viswanadham *et. al* [39]. Here the authors used the reachability graph of the Petri net model to arrive at static resource allocation policies. For deadlock avoidance, they proposed an on-line monitoring and control system.

Many other significant contributions on deadlock prevention are based on a linear algebraic characterization of deadlock in *ordinary*[4] net. In fact, it is well know that a necessary and sufficient condition for an ordinary net to be deadlocked is the following: the set of empty places of the net forms a *siphon*[5] and each transitions has at least one empty input place. An interesting deadlock prevention procedure has been proposed by Iordache, Moody, and Antsaklis in [21]: the approach consists in adding to the Petri net that models the plant a number of additional places that prevent reaching empty siphons, thus ensuring deadlock freeness. Other significant contributions on deadlock prevention based on a linear algebraic characterization of empty siphons are due to Ezpeleta *et al.* [11], to Chu and Xie [6], to Li and Zhou [24], and more recently to Reveliotis [35]. In particular, in [11] the authors consider a particular class of Petri nets and proved that for such a class deadlock prevention also ensures liveness. Finally, in [35] Reveliotis develop a general theory that provides a unifying framework for all the relevant existing results based on siphon analysis, and reveals the key structures and mechanisms that connect the resource allocation systems (RAS) non-liveness to the concept of deadly marked and empty siphon.

The most important contributions in the development of deadlock avoidance strategies are due to Park and Reveliotis. In [28] the authors shown that a significant class of deadlock avoidance policies, known as algebraic polynomial kernel–deadlock avoidance policies, originally developed in the finite-state automata paradigm, can be analysized using results from Petri net structural analysis. Other interesting results in this framework have been given by Park and Reveliotis in [29, 30].

Despite the above important contributions, deadlock prevention and deadlock avoidance are still open research areas because in the case of very large scale problems the computational complexity of most of the existing approaches may be prohibitive and, in the case of deadlock prevention, the number of places that should be added is too large. Moreover, using the above approaches it is not always possible to deal with the case of uncontrollable and unobservable transitions.

8.3 Observability

If the marking of a Petri net system is not measurable, different information can be used to reconstruct it, or at least to estimate it.

[4] A net is ordinary if all the arc weight are unitary.

[5] A *siphon* is a set of places $\mathcal{S} \subseteq P$ such that $\bigcup_{p\in\mathcal{S}} {}^\bullet p \subseteq \bigcup_{p\in\mathcal{S}} p^\bullet$, i.e., all transitions outputting to one place of the set are also inputting from one place of the set.

Benasser in [1] has studied the possibility of defining the set of markings reached firing a "partially specified" set of transitions using logical formulas, without having to enumerate this set.

Meda *et al.* in [25] have discussed the problem of estimating the marking of a Petri net using a mix of transition firings and place observations. Finally, Zhang and Holloway [40] used a Controlled Petri Net model for forbidden state avoidance under partial *event* observation assuming that the initial marking is known.

We have also worked in this area and studied the following cases.

- The initial marking of the net is not known (or only a partial information of it is available) but all events are observable.
- The initial marking of the net is known but the events occurrence is observed though a labeling function, i.e., a mask, that makes some events *undistinguishable* or *silent*.

While the first case can be studied using *unlabeled* Petri nets, the second case requires *labeled* models. In particular, undistinguishable events are modeled with transitions that share the same label; silent events are modeled with transitions whose label is equal to the empty string.

In all cases the goal is that of characterizing the set of markings in which the system may be, given the actual observation, and the information, if any, on the initial marking and on the structure of the net. We denote this set as $\mathcal{C}(w)$, where w is the word of observed events, and we call it the set of *markings consistent with* w.

Some important contributions in this topic have been given in [7, 14, 18] where it has been proved that in all the above cases the set $\mathcal{C}(w)$ may be characterized by a *finite* set of *linear algebraic* constraints whose structure keeps the same regardless of the length of the observed word w, and depends on some parameters that can be easily computed using appropriate recursive algorithms. The main advantage of such an approach is that it does not require the enumeration of the set of consistent markings.

In [15, 16] it has also been shown how such characterizations can be efficiently used when the observer is included in a control loop, and when designing a diagnoser for fault detection.

8.4 Identification

The first partial but interesting approach to identification is due Hiraishi [19] on the synthesis of safe Petri nets. Bourdeaud'huy and Yim [2] have presented an approach based on logic constraints that can deal with positive examples of firing sequences but not with counterexamples.

A different approach is based on the *theory of regions* whose objective is that of deciding whether a given graph is isomorphic to the reachability graph of some free labeled net and then constructing it. An excellent survey of this approach, that also presents some efficient algorithms for net synthesis based on linear algebra, can be found in the paper by Badouel and Darondeau [8].

Recently, in a series of paper [3, 4, 17] we have presented a general approach to identification based on integer programming. In particular, in [17] the problem of identifying a Petri net system, given a finite language that it generates, has been considered. Note that this approach allows one to specify not only examples of the systems behavior (i.e., strings that belong to the language) but also counterexamples (i.e., strings that do not belong to the language). It has been shown that the identification problem can be solved via an integer programming problem, and additional structural constraints can also be easily imposed to the net. The above results have been extended in [3] to the case of *labeled* Petri nets.

In [4] the following identification problem has been dealt with: given an automaton that represents the coverability graph of a net, determine a net system whose coverability graph is isomorph to the automaton. Again the proposed approach requires solving an integer programming problem whose set of unknowns contains the elements of the pre and post incidence matrices and the initial marking of the net.

Finally, in a recent paper Sreenivas [38] dealt with a related topic: the minimization of Petri net models. Given a λ-free labeled Petri net generator and a measure function — that associates to it, say, a non negative integer — the objective is that of finding a Petri net that generates the same language of the original net while minimizing the given measure. Unfortunately, these minimization procedures only exist for restricted families of Petri net languages where language-containment is decidable, and for a restricted class of measures.

9 Conclusion

In this paper we considered Petri nets, an efficient and powerful formalism for the simulation, analysis, and control of discrete event systems. The purpose of this paper is that of presenting the main features of such a model to the automatic control community, whose interest in the area of discrete event systems has been constantly increasing during the last years. In particular, we focus on the basic Petri net model, namely on Place/Transition nets, and discuss via some simple but intuitive examples, its modeling power, and its main advantages with respect to other formalisms, such as automata. A discussion on the main current research area is also presented, together with a survey of the classical results in this framework.

References

1. Benasser, A., "Reachability in Petri nets: an approach based on constraint programming," *Ph.D. Thesis*, Université de Lille, France (in French), 2000.
2. Bourdeaud'huy T., P. Yim, "Synthèse de réseaux de Petri à partir d'exigences," Actes de la 5me conf. francophone de Modélisation et Simulation, (Nantes, France), pp. 413–420, (in French), Sep. 2004.
3. Cabasino M.P., A. Giua, C. Seatzu, "Identification of deterministic Petri nets," *8th Int. Workshop on Discrete Event Systems*, Ann Arbor, Michigan, USA, Jul 2006.

4. Cabasino M.P., A. Giua, C. Seatzu, "Identification of unbounded Petri nets from their coverability graph," *45th IEEE Int. Conf. on Decision and Control*, San Diego, CA, USA, Dec 2006.
5. Cassandras C.G., S. Lafortune, "Introduction to discrete event systems," Kluwer Academic Publishers, 1999.
6. Chu F., X. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming", *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 6, pp. 793-804, 1997.
7. Corona D., A. Giua, C. Seatzu, "Marking estimation of Petri nets with silent transitions," *43rd IEEE Int. Conf. on Decision and Control*, Atlantis, Paradise Island, Bahamas, Dec 2004.
8. Badouel E., P. Darondeau, "Theory of regions", Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models, Springer-Verlag, (eds. Reisig, W. and Rozenberg, G.), Vol. 1491, pp. 529–586, 1998.
9. David R., H. Alla, "Discrete, continous and hybrid Petri nets," Springer, Berlin, Heidelberg, 2005.
10. DiCesare F., G. Harhalakis, J.M. Proth, M. Silva and F.B. Vernadat, "Practice of Petri nets in manufacturing," Chapman and Hall, 1993.
11. Ezpeleta J., J.M. Colom, J. Martinez, "A PN based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. on Robotics and Automation*, Vol. 11, No. 8, pp. 173–184, 1995.
12. Gaubert S., A. Giua, "Petri net languages and infinite subsets of $\mathbb{N}^m$," *Journal of Computer and System Sciences*, Vol. 59, No. 3, pp. 373-391, 1999.
13. Giua A., F. DiCesare, M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," *1992 IEEE Int. Conf. on Systems, Man, and Cybernetics*, 1992 IEEE Int. Conf. on Systems, Man, and Cybernetics, Chigago, October, 1992.
14. Giua A., C. Seatzu, "Observability of Place/Transition nets," *IEEE Trans. on Automatic Control*, Vol. 47, No. 9, pp. 1424-1437, 2002.
15. Giua A., C. Seatzu, F. Basile, "Observer based state-feedback control of timed Petri nets with deadlock recovery," *IEEE Trans. on Automatic Control*, Vol. 49, No. 1, pp. 17-29, 2004.
16. Giua A., C. Seatzu, "Fault detection for discrete event systems using labeled Petri nets," *44th IEEE Int. Conf. on Decision and Control and European Control Conference 2005*, Seville, Spain, Dec 2005.
17. Giua A., C. Seatzu, "Identification of free-labeled Petri nets via integer programming," *44th IEEE Int. Conf. on Decision and Control and European Control Conference 2005*, Seville, Spain, Dec 2005.
18. Giua A., D. Corona, C. Seatzu, "State estimation of λ-free labeled Petri nets with contact-free nondeterministic transitions," *J. of Discrete Event Dynamic Systems*, Vol. 15, No. 1, pp. 85-108, 2005.
19. Hiraishi K., "Construction of a class of safe Petri nets by presenting firing sequences", Lecture Notes in Computer Science; 13th International Conference on Application and Theory of Petri Nets 1992, Sheffield, UK, Springer-Verlag, (ed. K. Jensen), Vol. 616, pp. 244–262, 1992.
20. Holloway L.E., B.H. Krogh, A. Giua, "A survey of Petri nets methods for controlled discrete event systems," *Discrete Event Dynamic Systems*, Vol. 7, No. 2, pp. 151–190, 1997.
21. Iordache M.V., J.O. Moody, P.J. Antsaklis, "Synthesis of deadlock prevention supervisors using Petri nets," *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 1, pp. 59–68, 2002.

22. Iordache M.V., P.J. Antsaklis, "Supervisory control of concurrent systems. A Petri net structural approach," Series: Systems & Control: Foundations & Applications. Birkhäuser, 2006.
23. Lewis H.R., C.H. Papadimitriou, "Elements of the theory of computation," Prentice-Hall, 1981.
24. Li Z., M. Zhou, "Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems", *IEEE Trans. on Systems, Man, Cybernetics, Part A*, Vol. 34, No. 1, pp. 38–51, 2004.
25. Meda M.E., A. Ramírez, A. Malo, "Identification in discrete event systems, *IEEE Int. Conf. on Systems, Man and Cybernetics*, San Diego, California, pp. 740-745, oct 1998.
26. Murata T., "State equation, controllability, and maximal matching of Petri nets," *IEEE Trans. on Automatic Control*, Vol. 22, No. 6, pp. 412-416, 1977.
27. Murata T., "Petri nets: properties, analysis and applications," *Proceedings IEEE*, Vol. 77, No. 4, pp. 541–580, 1989.
28. Park J., S.A. Reveliotis, "Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. on Automatic Control*, Vol. 16, No. 2, pp. 190–195, 2000.
29. Park J., S.A. Reveliotis, "Policy mixtures: a novel approach for enhancing the operational flexibility of resource allocation systems with alternate routings," *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 4, pp. 616–620, 2002.
30. Park J., S.A. Reveliotis, "Liveness-enforcing supervision for resource allocation systems with uncontrollable behavior and forbidden states," *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 2, pp. 234–238, 2002.
31. Parigot M., E. Peltz, "A logical formalism for the study of the finite behavior of Petri nets," Advances in Petri Nets 1985, Lecture Notes in Computer Science 222, G. Rozenberg (ed.), Springer Verlag, pp. 346–361, 1985.
32. Peterson J.L., "Petri net theory and the modeling of systems," Prentice-Hall, 1981.
33. Ramadge P.J., W.M. Wonham, "The control of discrete event systems," *Proceedings IEEE*, Vol. 77, No. 1, pp. 81-98, 1989.
34. Recalde L., "Structural methods for the design and analysis of concurrent systems modeled with Place/Transition nets," PhD Thesis, DIIS. Univ. Zaragoza, 1998.
35. Reveliotis S.A., "Siphon-based characterization of liveness and liveness-enforcing supervision for sequential resource allocation systems," In Deadlock Resolution in Computer-Integrated Systems, M. Zhou and M. P. Fanti, (Eds), Marcel Dekker, Inc., pp. 283-307, 2005.
36. Silva M., J.M. Colom, J. Campos, "Linear algebraic techniques for the analysis of Petri nets," *Int. Symp. on Mathematical Theory of Networks and Systems*, MITA Press (Tokyo, Japan), 1992.
37. Sreenivas R.S., B.H. Krogh, "On Petri net models of infinite state supervisors," *IEEE Trans. on Automatic Control*, Vol. 37, No. 2, pp. 274–277, 1992.
38. Sreenivas R.S., "On minimal representations of Petri net languages," *6th Workshop on Discrete Event Dynamic Systems*, (Zaragoza, Spain), pp. 237–242, oct 2002.
39. Viswanadham N., Y. Narahari, T.L. Johnson, " Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models," *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 6, pp. 713–723, 1990.
40. Zhang L., L.E. Holloway, "Forbidden state avoidance in controlled Petri nets under partial observation," *33rd Allerton Conference*, Monticello, Illinois, pp. 146–155, 1995.