Chapter 11
Structural Analysis of Petri Nets

Maria Paola Cabasino, Alessandro Giua, and Carla Seatzu

11.1 Introduction

This chapter is devoted, as the previous one, to the presentation of background material on P/T nets. In particular, here the main focus is on *structural analysis* that consists in a set of algebraic tools that do not require the enumeration of the reachability set of a marked net but are based on the analysis of the state equation, on the incidence matrix, etc.

It is shown how the existence of some vectors, called P- and T-vectors, characterize the behavior of the net. In particular P-vectors enable to express some important constraints on the number of places in certain subsets of places, e.g., they can be constant, increasing or decreasing during the net evolution. On the contrary, T-vectors are related to transitions and express the effect of some sequences of transitions on the marking of the net, e.g., they can keep it unaltered, or make it increase or decrease.

The structural counterpart of several behavioral properties, that we had defined in the previous chapter, will also be defined. The properties we will consider are boundedness, conservativeness, repetitiveness, reversibility and liveness: in their structural form they are related to the net structure regardless of the initial marking. These properties will be characterized in terms of P- and T-vectors.

Subclasses of Petri nets are finally defined. Some of these classes pose some restrictions on the nature of physical systems they can model. However, this restricted modeling power often leads to simplified analysis criteria and this motivates the interest in these subclasses.

Maria Paola Cabasino · Alessandro Giua · Carla Seatzu
Department of Electrical and Electronic Engineering, University of Cagliari, Italy
e-mail: {cabasino,giua,seatzu}@diee.unica.it

C. Seatzu et al. (Eds.): Control of Discrete-Event Systems, LNCIS 433, pp. 213–233.
springerlink.com

11.2 Analysis via State Equation

In this section we present an approach to characterize marking reachability by means of the state equation of a net, that can be solved using integer programming techniques. The main limitation of this approach consists in the fact that in general it only provides necessary (but not sufficient) conditions for reachability. However, as discussed at the end of this chapter, there exist classes of nets (such as acyclic nets, state machines and marked graphs) for which the analysis based on the state equation provides necessary and sufficient conditions for reachability. Unfortunately, both marked graphs and state machines are very restricted classes of models.

Definition 10.5 introduced the notion of transition firing and allows one to compute the marking reached after the firing of an enabled transition through a simple matrix equation. Such a condition can be generalized to a sequence of transitions σ.

Definition 11.1. *Given a net N with set of transitions $T = \{t_1, t_2, \cdots, t_n\}$ and a sequence of transitions $\sigma \in T^*$, we call* firing vector *(or* firing count vector*) of σ the vector*

$$\sigma = [\sigma[t_1]\ \sigma[t_2]\ \cdots\ \sigma[t_n]]^T \in \mathbb{N}^n$$

whose entry $\sigma[t]$ denotes how many times transition t appears in sequence σ.

Consider the sequence $\sigma = t_1 t_2 t_2$ in the net in Fig. 11.1(*a*). Since t_1 appears once in σ, while t_2 appears twice, the firing vector of σ is $\sigma = [1\ 2]^T$.

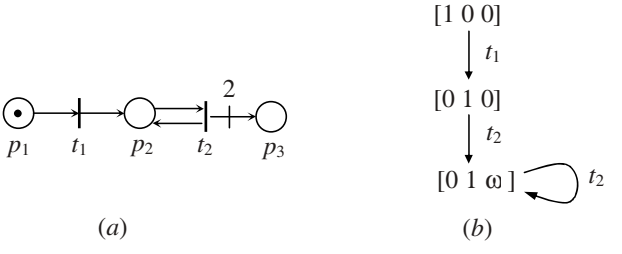


Fig. 11.1 (a) A marked PN; (b) its coverability graph

Proposition 11.1 (State equation). *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a marked net and $\boldsymbol{C}$ its incidence matrix. If $\boldsymbol{m}$ is reachable from $\boldsymbol{m}_0$ firing σ it holds that*

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \sigma. \tag{11.1}$$

Proof. Assume that $\boldsymbol{m}_0[\sigma\rangle \boldsymbol{m}$ with $\sigma = t_{j_1} t_{j_2} \cdots t_{j_r}$, i.e., $\boldsymbol{m}_0[t_{j_1}\rangle \boldsymbol{m}_1[t_{j_2}\rangle \boldsymbol{m}_2 \cdots [t_{j_r}\rangle \boldsymbol{m}_r$, with $\boldsymbol{m}_r = \boldsymbol{m}$. It holds that:

$$\boldsymbol{m} = \boldsymbol{m}_r = \boldsymbol{m}_{r-1} + \boldsymbol{C}[\cdot, t_{j_r}] = \cdots = \boldsymbol{m}_0 + \sum_{k=1}^{r} \boldsymbol{C}[\cdot, t_{j_k}] = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \sigma$$

and the state equation is satisfied. □

Consider the net system in Fig. 11.1(*a*). It is easy to verify that $\boldsymbol{m}' = [0\ 1\ 4]^T$, reachable from the initial marking $\boldsymbol{m}_0 = [1\ 0\ 0]^T$ firing $\sigma = t_1 t_2 t_2$, satisfies the equation $\boldsymbol{m}' = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \sigma$.

Definition 11.2. *Given a marked net $\langle N, \boldsymbol{m}_0 \rangle$ with m places and n transitions, let $\boldsymbol{C}$ be its incidence matrix. The* potentially reachable set *of $\langle N, \boldsymbol{m}_0 \rangle$ is the set*

$$PR(N, \boldsymbol{m}_0) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \exists \boldsymbol{y} \in \mathbb{N}^n : \ \boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{y}\},$$

i.e., the set of vectors $\boldsymbol{m} \in \mathbb{N}^m$ such that there exists a vector $\boldsymbol{y} \in \mathbb{N}^n$ that satisfies the state equation.

Proposition 11.2. [1] *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a marked net. It is: $R(N, \boldsymbol{m}_0) \subseteq PR(N, \boldsymbol{m}_0)$.*

Proof. We simply need to prove that if $\boldsymbol{m}$ is reachable, then $\boldsymbol{m}$ is also potentially reachable. Indeed, if $\boldsymbol{m}$ is reachable, there exists a sequence σ such that $\boldsymbol{m}_0[\sigma\rangle \boldsymbol{m}$. Thus $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{y}$ with $\boldsymbol{y} = \sigma$, i.e., $\boldsymbol{m} \in PR(N, \boldsymbol{m}_0)$. □

We show by means of an example that the converse of Proposition 11.2 does not hold, i.e., it may happen that $R(N, \boldsymbol{m}_0) \subsetneq PR(N, \boldsymbol{m}_0)$.

Consider the marked net in Fig. 11.1(*a*) whose incidence matrix is $\boldsymbol{C} = [\ -1\ 0;\ 1\ 0;\ 0\ 2\ ]$. The initial marking is $\boldsymbol{m}_0 = [1\ 0\ 0]^T$. Let $\boldsymbol{m} = [1\ 0\ 2]^T$. Equation $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{y}$ is verified by $\boldsymbol{y} = [0\ 1]^T$. However, $\sigma = t_2$ is not enabled at the initial marking and $\boldsymbol{m}$ is not reachable.

Definition 11.3. *Given a marked net $\langle N, \boldsymbol{m}_0 \rangle$, potentially reachable but not reachable markings are said to be* spurious markings.

The presence of spurious markings implies that in general the state equation analysis provides necessary, but not sufficient, conditions for reachability. Note however, that the necessary condition in Proposition 11.2 often allows to verify that a marking is not reachable. In the net in Fig. 11.1(*a*), consider the marking $\boldsymbol{m} = [0\ 2\ 0]^T$ and a generic vector $\boldsymbol{y} = [y_1\ y_2]^T \in \mathbb{N}^2$. The equation $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{y}$ implies

$$\begin{array}{rcl} \boldsymbol{m} - \boldsymbol{m}_0 = & \boldsymbol{C} \cdot \boldsymbol{y} \\ \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} = & \begin{bmatrix} -y_1 \\ y_1 \\ 2y_2 \end{bmatrix} . \end{array}$$

The first and second of such equalities require that: $y_1 = 1$ and $y_1 = 2$. Thus the equation does not admit solution and $\boldsymbol{m}$ is not reachable because it does not satisfy Proposition 11.2.

In the previous Chapter 10 we introduced the notion of *covering set* (cfr. Definition 10.12) that provides an approximation of the reachability set. In particular, by Proposition 10.3, it is $R(N, \boldsymbol{m}_0) \subseteq CS(N, \boldsymbol{m}_0)$. It is natural to wonder if an inclusion relationship exists between $PR(N, \boldsymbol{m}_0)$ and $CS(N, \boldsymbol{m}_0)$. The PN system in Fig. 11.1(*a*) shows that no relationship exists. Indeed, it is

$$CS(N, \boldsymbol{m}_0) = \{[1\ 0\ 0]^T\} \cup \{[0\ 1\ k]^T,\ k \in \mathbb{N}\}$$

as it can be easily verified looking at the coverability graph in Fig. 11.1(*b*). Moreover, it is

$$PR(N,\boldsymbol{m}_0) = \{[1\ 0\ 2k]^T,\ k \in \mathbb{N}\} \cup \{[0\ 1\ 2k]^T,\ k \in \mathbb{N}\}.$$

Therefore there exist markings that belong to $PR(N,\boldsymbol{m}_0)$, but do not belong to $CS(N,\boldsymbol{m}_0)$, and viz. As an example, given $\boldsymbol{m} = [0\ 1\ 3]^T$, it is $\boldsymbol{m} \in CS(N,\boldsymbol{m}_0)$ but $\boldsymbol{m} \notin PR(N,\boldsymbol{m}_0)$. On the contrary, if we consider $\boldsymbol{m}' = [1\ 0\ 2]^T$, it is $\boldsymbol{m}' \in PR(N,\boldsymbol{m}_0)$, but $\boldsymbol{m}' \notin CS(N,\boldsymbol{m}_0)$.

11.3 Analysis Based on the Incidence Matrix

11.3.1 Invariant Vectors

Definition 11.4. *Given a net N with m places and n transitions, let $\boldsymbol{C}$ be its incidence matrix. A P-vector*[1] *$\boldsymbol{x} \in \mathbb{N}^m$ with $\boldsymbol{x} \neq \boldsymbol{0}$ is called:*

- P-invariant*: if $\boldsymbol{x}^T \cdot \boldsymbol{C} = \boldsymbol{0}^T$;*
- P-increasing*: if $\boldsymbol{x}^T \cdot \boldsymbol{C} \gneq \boldsymbol{0}^T$;*
- P-decreasing*: if $\boldsymbol{x}^T \cdot \boldsymbol{C} \lneq \boldsymbol{0}^T$.*

A T-vector[2] *$\boldsymbol{y} \in \mathbb{N}^n$ with $\boldsymbol{y} \neq \boldsymbol{0}$ is called:*

- T-invariant*: if $\boldsymbol{C} \cdot \boldsymbol{y} = \boldsymbol{0}$;*
- T-increasing*: if $\boldsymbol{C} \cdot \boldsymbol{y} \gneq \boldsymbol{0}$;*
- T-decreasing*: if $\boldsymbol{C} \cdot \boldsymbol{y} \lneq \boldsymbol{0}$.*

Consider the nets in Fig. 11.2 (*a*) and (*b*) whose incidence matrices are respectively

$$\boldsymbol{C}_a = \begin{bmatrix} -1 & 2 \\ 1 & -1 \\ 0 & -1 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{C}_b = \boldsymbol{C}_a^T = \begin{bmatrix} -1 & 1 & 0 \\ 2 & -1 & -1 \end{bmatrix}.$$

For the net in figure (*a*) one readily verifies that vector $\boldsymbol{x}_I = [1\ 1\ 1]^T$ is a P-invariant, vector $\boldsymbol{x}_C = [1\ 1\ 0]^T$ is a P-increasing and vector $\boldsymbol{x}_D = [0\ 0\ 1]^T$ is a P-decreasing. For the net in figure (*b*) one readily verifies that vector $\boldsymbol{y}_I = [1\ 1\ 1]^T$ is a T-invariant, vector $\boldsymbol{y}_C = [1\ 1\ 0]^T$ is a T-increasing and vector $\boldsymbol{y}_D = [0\ 0\ 1]^T$ is a T-decreasing.

The following definition holds.

Definition 11.5. *The* support of a P-vector $\boldsymbol{x} \in \mathbb{N}^m$, *denoted* $\|\boldsymbol{x}\|$, *is the set of places $p \in P$ such that $x[p] > 0$. The* support of a T-vector $\boldsymbol{y} \in \mathbb{N}^n$, *denoted* $\|\boldsymbol{y}\|$, *is the set of transitions $t \in T$ such that $y[t] > 0$.*

As an example, for the net in Fig. 11.2(*a*) it is $\|\boldsymbol{x}_I\| = \{p_1, p_2, p_3\}$, $\|\boldsymbol{x}_C\| = \{p_1, p_2\}$ and $\|\boldsymbol{x}_D\| = \{p_3\}$.

[1] A P-vector can also be represented by a function $\boldsymbol{x} : P \to \mathbb{N}$.

[2] A T-vector can also be represented by a function $\boldsymbol{y} : T \to \mathbb{N}$.

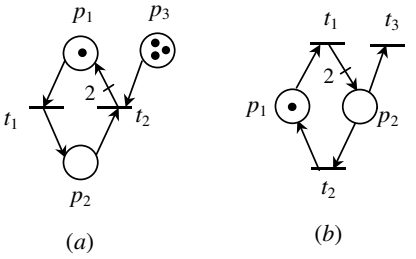


Fig. 11.2 Two PNs for the analysis via invariants

The following proposition allows to give a physical interpretation of P-vectors.

Proposition 11.3. *Given a net N, let $\boldsymbol{x}_I$ be a P-invariant, $\boldsymbol{x}_C$ a P-increasing, $\boldsymbol{x}_D$ a P-decreasing. For all markings $\boldsymbol{m} \in \mathbb{N}^m$ and for all enabled sequences $\sigma \in L(N,\boldsymbol{m})$ such that $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}'$ it holds that*

$$\boldsymbol{x}_I^T \cdot \boldsymbol{m}' = \boldsymbol{x}_I^T \cdot \boldsymbol{m}; \qquad \boldsymbol{x}_C^T \cdot \boldsymbol{m}' \gtrsim \boldsymbol{x}_C^T \cdot \boldsymbol{m}; \qquad \boldsymbol{x}_D^T \cdot \boldsymbol{m}' \lesssim \boldsymbol{x}_D^T \cdot \boldsymbol{m}. \tag{11.2}$$

Proof. If $\boldsymbol{m}'$ is reachable from $\boldsymbol{m}$ with the firing of sequence σ it holds that: $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C} \cdot \sigma$ and for any P-vector $\boldsymbol{x}$ it holds that $\boldsymbol{x}^T \cdot \boldsymbol{m}' = \boldsymbol{x}^T \cdot \boldsymbol{m} + \boldsymbol{x}^T \cdot \boldsymbol{C} \cdot \sigma$.

By definition of P-invariant $\boldsymbol{x}_I^T \cdot \boldsymbol{C} = \boldsymbol{0}^T$, i.e., $\boldsymbol{x}^T \cdot \boldsymbol{C} \cdot \sigma = 0$ hence the first result in (11.2) follows. By definition of P-increasing $\boldsymbol{x}_I^T \cdot \boldsymbol{C} \gtrsim \boldsymbol{0}^T$ and since $\sigma \gtrsim \boldsymbol{0}$ it holds that $\boldsymbol{x}^T \cdot \boldsymbol{C} \cdot \sigma \geq 0$, hence the second result in (11.2) follows. The last result can be proved in a similar fashion. □

Applying the result of the previous proposition to the net in Fig. 11.2 (*a*), where $\boldsymbol{m}_0 = [1\ 0\ 3]^T$, one concludes that for all reachable markings $\boldsymbol{m} \in R(N,\boldsymbol{m}_0)$:

- the sum of the tokens in the net remains constant and equal to 4 because the P-invariant $\boldsymbol{x}_I = [1\ 1\ 1]^T$ ensures that $m[p_1] + m[p_2] + m[p_3] = \boldsymbol{x}_I^T \cdot \boldsymbol{m} = \boldsymbol{x}_I^T \cdot \boldsymbol{m}_0 = 4$;
- the sum of the tokens in places p_1 and p_2 may increase starting from the initial value 1 but never decreases, because the P-increasing $\boldsymbol{x}_C = [1\ 1\ 0]^T$ ensures that $m[p_1] + m[p_2] = \boldsymbol{x}_C^T \cdot \boldsymbol{m} \geq \boldsymbol{x}_C^T \cdot \boldsymbol{m}_0 = 1$;
- the number of tokens in place p_3 may decrease starting from the initial value 3 but never increases, because the P-decreasing $\boldsymbol{x}_D = [0\ 0\ 1]^T$ ensures that $m[p_3] = \boldsymbol{x}_D^T \cdot \boldsymbol{m} \leq \boldsymbol{x}_D^T \cdot \boldsymbol{m}_0 = 3$.

The following proposition provides a physical interpretation of T-vectors.

Proposition 11.4. *Given a net N, let $\boldsymbol{m} \in \mathbb{N}^m$ be a marking and $\sigma \in L(N,\boldsymbol{m})$ be a firing sequence such that $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}'$. The following properties hold:*

- *the firing vector σ is a T-invariant $\iff \boldsymbol{m}' = \boldsymbol{m}$, i.e., sequence σ is repetitive and stationary;*
- *the firing vector σ is a T-increasing $\iff \boldsymbol{m}' \gtrsim \boldsymbol{m}$, i.e., sequence σ is repetitive increasing;*
- *the firing vector σ is a T-decreasing $\iff \boldsymbol{m}' \lesssim \boldsymbol{m}$.*

Proof. The three properties can be easily proved considering the state equation $\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{C} \cdot \sigma$ and recalling the properties of T-vectors given in Definition 11.4. □

As an example, consider the net in Fig. 11.2 (*b*). Given the marking $\boldsymbol{m} = [1\ 0]^T$ shown in the figure, sequence $\sigma' = t_1 t_2$ is enabled and its firing yields marking $\boldsymbol{m}' = [1\ 1]^T \gneq \boldsymbol{m}$: the firing vector of σ' is the T-increasing $\boldsymbol{y}_C = [1\ 1\ 0]^T$. From the same marking $\boldsymbol{m}$, the firing of sequence $\sigma'' = t_1 t_2 t_3$ yields $\boldsymbol{m}'' = \boldsymbol{m}$: the firing vector of σ'' is the T-invariant $\boldsymbol{y}_I = [1\ 1\ 1]^T$.

Remark 11.1. *Many authors use a different terminology for* P-invariants *and* T-invariants*, and call them, respectively,* P-semiflow *and* T-semiflow*. In particular, this is the terminology used in the following Chapters 18 and 20. Moreover, in these chapters the term* invariant *is used to denote the token conservation law* $\boldsymbol{x}_I^T \cdot \boldsymbol{m}' = \boldsymbol{x}_I^T \cdot \boldsymbol{m}$ *in Proposition 11.3.* ■

11.3.2 P-Invariants Computation

Definition 11.6. *A P-invariant* $\boldsymbol{x} \in \mathbb{N}^m$ *is called:*

- minimal *if there does not exist a P-invariant* $\boldsymbol{x}'$ *such that* $\boldsymbol{x}' \lneq \boldsymbol{x}$*;*
- of minimal support *if there does not exist a P-invariant* $\boldsymbol{x}'$ *such that* $\|\boldsymbol{x}'\| \subsetneq \|\boldsymbol{x}\|$.

Analogous definitions hold for T-invariants.

As an example, vector $\boldsymbol{x}_I = [1\ 1\ 1]^T$ for the net in Fig. 11.2(*a*) is a minimal P-invariant that has also minimal support. Note however, that there may exist minimal invariants that do not have minimal support. As an example, in the net in Fig. 11.3, P-invariants $\boldsymbol{x}' = [1\ 2\ 0]^T$ and $\boldsymbol{x}'' = [1\ 0\ 2]^T$ are minimal and have minimal support; P-invariant $\boldsymbol{x}''' = 0.5(\boldsymbol{x}' + \boldsymbol{x}'') = [1\ 1\ 1]^T$ is minimal but does not have minimal support.

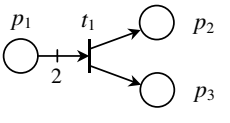


Fig. 11.3 A net with a minimal P-invariant that is not of minimal support

A non minimal or non minimal support P-invariant can always be obtained as the linear combination, with positive coefficients, of one or more minimal and minimal support P-invariants. As an example, given the net in Fig. 11.2(*a*) it holds that $\boldsymbol{x}''' = 0.5(\boldsymbol{x}' + \boldsymbol{x}'')$.

The following algorithm determines a set of P-invariants of a net. In particular, it computes all minimal P-invariants, but also many others that are not minimal (in general an exponential number of non minimal). Solutions for this issue can be found in [12].

Algorithm 11.2. (P-invariants computation). *Consider a net N with m places and n transitions and let $\boldsymbol{C}$ be its incidence matrix.*

1. *Compute the table $\boldsymbol{A} := | \boldsymbol{C} | \boldsymbol{I}_{m\times m} |$, where $\boldsymbol{I}_{m\times m}$ is the $m \times m$ identity matrix.*
2. *For $j := 1, \ldots, n$ (column index associated with transitions):*
   a. *let $J_+ := \{i \mid A[i,j] > 0\}$ be the set of row indices that correspond to positive entries in column j;*
   b. *let $J_- := \{i \mid A[i,j] < 0\}$ be the set of row indices that correspond to negative entries in column j;*
   c. *for each pair $(i_+, i_-) \in J_+ \times J_-$:*
      i. *let $d := lcm\{A[i_+,j], -A[i_-,j]\}$ be the least common multiplier of entries $A[i_+,j]$ and $-A[i_-,j]$;*
      ii. *let $d_+ := d/A[i_+,j]$ and $d_- := -d/A[i_-,j]$;*
      iii. *add the new row $d_+ \boldsymbol{A}[i_+,\cdot] + d_- \boldsymbol{A}[i_-,\cdot]$ to the table (the new row has the $j-$th entry equal to zero by construction);*
   d. *remove from $\boldsymbol{A}$ all rows with index $J_+ \cup J_-$, corresponding to non-null elements along the $j-$th column.*
3. *The resulting table $\boldsymbol{A}$ is in the form $\boldsymbol{A} = | \boldsymbol{0}_{r\times m} | \boldsymbol{X}^T |$, where $\boldsymbol{0}_{r\times m}$ is a null $r \times n$ matrix, while $\boldsymbol{X}$ is a matrix with m rows and r columns. Each column of $\boldsymbol{X}$ is a P-invariant.*

Note that in the previous algorithm if any of the two sets J_+ or J_- is empty, at Step 2(c) no row is added. Moreover, the resulting table will be empty, i.e., $r = 0$, if the net N has no P-invariant.

Finally, note that it could be necessary to divide a column of $\boldsymbol{X}$ for the largest common divisor of its entries to obtain a P-invariant that is minimal.

Let us now present a simple example of application of such algorithm. Consider the net in Fig. 11.4.

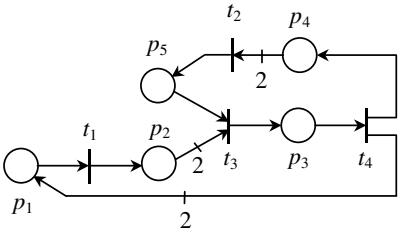


Fig. 11.4 A net for the computation of P-invariants

Initially construct the table

$$\left|\begin{array}{rrrr|ccccc|l} -1 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & p_1 \\ 1 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & p_2 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & p_3 \\ 0 & -2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & p_4 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & p_5 \end{array}\right|$$

where for a better understanding each row has been labeled with the corresponding place. At Step $j = 1$ the sum of rows p_1 and p_2 is computed and added to the table, while the two rows are removed, thus obtaining the table

$$\left|\begin{array}{rrrr|ccccc|l} 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & p_3 \\ 0 & -2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & p_4 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & p_5 \\ 0 & 0 & -2 & 2 & 1 & 1 & 0 & 0 & 0 & p_1 + p_2 \end{array}\right|$$

At step $j = 2$ the linear combination of row p_4 with row p_5 multiplied by 2 is executed, and the two rows are removed, obtaining the table

$$\left|\begin{array}{rrrr|ccccc|l} 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & p_3 \\ 0 & 0 & -2 & 2 & 1 & 1 & 0 & 0 & 0 & p_1 + p_2 \\ 0 & 0 & -2 & 1 & 0 & 0 & 0 & 1 & 2 & p_4 + 2p_5 \end{array}\right|$$

At step $j = 3$ two combinations are computed and added to the table: the sum of row p_3 multiplied by 2 and either row $p_1 + p_2$ or row $p_4 + 2p_5$. Removing the three rows, we get the table

$$\left|\begin{array}{rrrr|ccccc|l} 0 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & p_1 + p_2 + 2p_3 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & 1 & 2 & 2p_3 + p_4 + 2p_5 \end{array}\right|$$

At Step $j = 4$ there are no possible combinations and we simply remove row $2p_3 + p_4 + 2p_5$ that has a non-null entry in the fourth column. The resulting table is

$$\left|\begin{array}{rrrr|ccccc|l} 0 & 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & p_1 + p_2 + 2p_3 \end{array}\right|$$

Thus the net has a single minimal and minimum support P-invariant $\boldsymbol{x} = [1\ 1\ 2\ 0\ 0]^T$.

From the definition of P-invariants and T-invariants, one can readily see that the T-invariants of a net N with incidence matrix $\boldsymbol{C}$ are the P-invariants of the *dual net*[3] with incidence matrix $\boldsymbol{C}^T$, and viz. Thus Algorithm 11.2 can also be used to compute the T-invariants initializing the table as $\boldsymbol{A} := |\ \boldsymbol{C}^T\ |\ \boldsymbol{I}_{n\times n}\ |$. The $i-$th row in such a case should be labeled by transition t_i.

It is also important to observe that the previous algorithm can be modified to determine increasing (or decreasing) P-vectors: at Step 2(d) rather than eliminating all the rows in $\mathscr{I}_+ \cup \mathscr{I}_-$ only the rows with index $\mathscr{I}_-$ (or $\mathscr{I}_+$) should be removed

[3] See Definition 11.19 for a formal definition of duality.

since if $\boldsymbol{x}$ is an increasing (decreasing) vector positive (negative) entries in the product $\boldsymbol{z}^T = \boldsymbol{x} \cdot \boldsymbol{C}$ are allowed. However, in general, when the algorithm is applied for the computation of increasing or decreasing vectors, the resulting vectors are not only the minimal ones or those of minimum support, but many others as well that can be obtained as a linear combination of them.

11.3.3 Reachability Analysis Using P-Invariants

In this section we discuss how P-invariants can be used to approximate the reachability set of a marked net.

Definition 11.7. [1] *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a net with m places and $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_k]$ a matrix $m \times k$, whose generic column $\boldsymbol{x}_i$ is a P-invariant of N.*

The $X-$invariant set *of $\langle N, \boldsymbol{m}_0 \rangle$ is the set*

$$I_X(N, \boldsymbol{m}_0) = \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{X}^T \cdot \boldsymbol{m} = \boldsymbol{X}^T \cdot \boldsymbol{m}_0\},$$

i.e., the set of vectors $\boldsymbol{m} \in \mathbb{N}^m$ such that $\boldsymbol{x}_i^T \cdot \boldsymbol{m} = \boldsymbol{x}_i^T \cdot \boldsymbol{m}_0$ for all $i \in \{1, \ldots, k\}$.

It is easy to prove that the potentially reachable set of a marked net is contained in the $X-$invariant set for any matrix of P-invariants $\boldsymbol{X}$. The following proposition extends the results of Proposition 11.2.

Proposition 11.5. [1] *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a marked net and $\boldsymbol{X}$ a matrix whose columns are P-invariants of N. It holds: $R(N, \boldsymbol{m}_0) \subseteq PR(N, \boldsymbol{m}_0) \subseteq I_X(N, \boldsymbol{m}_0)$.*

Proof. The first inequality derives from Proposition 11.2. It is sufficient to prove that if $\boldsymbol{m}$ is potentially reachable, then $\boldsymbol{m}$ is also $X-$invariant. In fact, if $\boldsymbol{m}$ is potentially reachable, there exists a vector $\boldsymbol{y} \in \mathbb{N}^n$ such that $\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{y}$. Thus $\boldsymbol{X}^T \cdot \boldsymbol{m} = \boldsymbol{X}^T \cdot \boldsymbol{m}_0 + \boldsymbol{X}^T \cdot \boldsymbol{C} \cdot \boldsymbol{y}$. Being $\boldsymbol{X}$ a matrix whose columns are P-invariants of N it holds that $\boldsymbol{X}^T \cdot \boldsymbol{C} \cdot \boldsymbol{y} = \boldsymbol{0}$, thus $\boldsymbol{m} \in I_X(N, \boldsymbol{m}_0)$. □

Let us now present an example where it is $R(N, \boldsymbol{m}_0) \subset PR(N, \boldsymbol{m}_0) \subset I_X(N, \boldsymbol{m}_0)$, i.e., where strict inclusion holds. Consider again the net in Fig. 11.1(*a*). There exists only a minimal P-invariant $\boldsymbol{x} = [1\ 1\ 0]^T$, thus let $\boldsymbol{X} = \boldsymbol{x}$. Moreover,

$$R(N, \boldsymbol{m}_0) = \{\ [1\ 0\ 0]^T\} \cup \{[0\ 1\ 2k]^T,\ k \in \mathbb{N}\ \},$$

and, as already discussed in Section 11.2,

$$PR(N, \boldsymbol{m}_0) = \{\ [1\ 0\ 2k]^T,\ k \in \mathbb{N}\ \} \cup \{\ [0\ 1\ 2k]^T,\ k \in \mathbb{N}\ \} \subset R(N, \boldsymbol{m}_0).$$

A generic $\boldsymbol{m}$ is an $X-$invariant only if:

$$\begin{array}{ccc} \boldsymbol{X}^T & \cdot & \boldsymbol{m} \\ [1\ 1\ 0] & \cdot & \begin{bmatrix} m[p_1] \\ m[p_2] \\ m[p_3] \end{bmatrix} \end{array} = \begin{array}{ccc} \boldsymbol{X}^T & \cdot & \boldsymbol{m}_0 \\ [1\ 1\ 0] & \cdot & \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

or equivalently $m[p_1] + m[p_2] = 1$. Moreover, since each $m[p_i]$ should be a non-negative integer, it holds that:

$$I_X(N, \boldsymbol{m}_0) = \{[1\ 0\ k]^T,\ k \in \mathbb{N}\} \cup \{[0\ 1\ k]^T,\ k \in \mathbb{N}\},$$

i.e., it is $PR(N, \boldsymbol{m}_0) \subset I_X(N, \boldsymbol{m}_0)$.

We conclude this section extending the result in Proposition 10.3.

Proposition 11.6. *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a marked net and $\boldsymbol{X}$ a matrix whose columns are P-invariants of N. It holds that: $R(N, \boldsymbol{m}_0) \subseteq CS(N, \boldsymbol{m}_0) \subseteq I_X(N, \boldsymbol{m}_0)$.*

Proof. The first inequality derives from Proposition 10.3. It is sufficient to prove that $\boldsymbol{m} \in CS(N, \boldsymbol{m}_0)$ implies that $\boldsymbol{m}$ is also $X-$invariant. To this aim we first observe that the support of no P-invariant contains unbounded places. Therefore, if p is an unbounded place, for all marking $\boldsymbol{m} \in I_x(N, \boldsymbol{m}_0)$, $m[p]$ may take any value in $\mathbb{N}$. This implies that the statement $\boldsymbol{m} \in CS(N, \boldsymbol{m}_0) \Rightarrow \boldsymbol{m} \in I_X(N, \boldsymbol{m}_0)$ may be at most violated by bounded places. However, either of this cannot occur since bounded places in the coverability graph only contain reachable markings being by Proposition 11.5, $R(N, \boldsymbol{m}_0) \subseteq I_X(N, \boldsymbol{m}_0)$. □

11.4 Structural Properties

In this section we define meaningful *structural properties* of a P/T net. In Section 10.4.3 of the previous chapter we defined several behavioral properties of a marked net $\langle N, \boldsymbol{m}_0 \rangle$, such as boundedness, conservativeness, repetitiveness, reversibility and liveness. In this section we define the structural counterpart of these properties, relating them to the net structure independent of a particular initial marking. We also show how such properties can be verified using P-vectors and T-vectors.

Most of the proofs of the results presented in this section are omitted; the interested reader is addressed to [13].

11.4.1 Structural Boundedness

This property implies boundedness for all initial markings.

Definition 11.8. *Consider a P/T net N and one of its places $p \in P$.*

- *Place p is* structurally bounded *if it is bounded in $\langle N, \boldsymbol{m}_0 \rangle$ for all initial markings $\boldsymbol{m}_0$.*
- *Net N is* structurally bounded *if the marked net $\langle N, \boldsymbol{m}_0 \rangle$ is bounded for all initial markings $\boldsymbol{m}_0$.*

This property can be characterized in terms of P-vectors .

Proposition 11.7. *Consider a P/T net N and a place $p \in P$.*

1. *Place p is structurally bounded if and only if there exists a P-invariant or a P-decreasing $\boldsymbol{x}$ with $x[p] > 0$, i.e., $p \in \|\boldsymbol{x}\|$.*
2. *The net N is structurally bounded if and only if there exists a P-invariant or a P-decreasing of positive integers $\boldsymbol{x} \in \mathbb{N}_+^m$, i.e., $P = \|\boldsymbol{x}\|$.*

As an example, the net in Fig. 11.1(a) has P-invariant $[1\ 1\ 0]^T$. Thus places p_1 and p_2 are structurally bounded while place p_3 is not structurally bounded. Therefore the net is not structurally bounded.

Structural boundedness implies (behavioral) boundedness, since it holds for all initial markings. The opposite is not true: consider the net in Fig. 11.1(a) with initial marking $\boldsymbol{m}_0 = [0\ 0\ r]^T$ (for all $r \in \mathbb{N}$). The resulting marked net is bounded, even if it is not structurally bounded: indeed in such a net only places p_1 and p_2 are structurally bounded.

11.4.2 Structural Conservativeness

Definition 11.9. *Consider a P/T net N and one of its places $p \in P$.*

- *Net N is* structurally strictly conservative *if the marked net $\langle N, \boldsymbol{m}_0 \rangle$ is strictly conservative for all initial markings $\boldsymbol{m}_0$.*
- *Net N is* structurally conservative *if the marked net $\langle N, \boldsymbol{m}_0 \rangle$ is conservative for all initial markings $\boldsymbol{m}_0$.*

This property can be characterized in terms of P-invariants.

Proposition 11.8. *Consider a P/T net N.*

- *N is structurally strictly conservative if and only if vector $\mathbf{1} = \{1\}^m$ is a P-invariant.*
- *N is structurally conservative if and only if there exists a P-invariant of positive integers $\boldsymbol{x} \in \mathbb{N}_+^m$, i.e., a P-invariant whose support contains all places.*

Structural conservativeness implies (behavioral) conservativeness, since it holds for all initial markings. The opposite is not true. As an example, the net in Fig. 10.12(*a*) in Chapter 10 is conservative with respect to the vector $[1\ 1]^T$ for the given initial marking, but such a net admits no P-invariant, thus it is not structurally conservative. It is easy to see that the net system $\langle N, \boldsymbol{m}_0 \rangle$ is not conservative if $m_0[p_1] \geq 1$ and $m_0[p_1] + m_0[p_2] \geq 2$, since in such a case transition t_4 would be almost-live and its firing would decrease the number of tokens in the net.

Let us finally observe, as already discussed for the corresponding behavioral properties, that structural conservativeness implies structural boundedness while the opposite is not true. The net in Fig. 10.10 (Chapter 10) is an example of a structurally bounded net being vector $[1]$ a P-decreasing, while it is not structurally conservative since it admits no P-invariant.

11.4.3 Structural Repetitiveness and Consistency

These properties are the structural counterpart of repetitiveness and stationarity introduced for sequences of transitions.

Proposition 11.9. *Consider a P/T net N.*

- *N is* repetitive *if there exists an initial marking* $\boldsymbol{m}_0$ *such that* $\langle N, \boldsymbol{m}_0 \rangle$ *admits a repetitive sequence containing all transitions.*
- *N is* consistent *if there exists an initial marking* $\boldsymbol{m}_0$ *such that* $\langle N, \boldsymbol{m}_0 \rangle$ *admits a repetitive stationary sequence containing all transitions.*

These properties can be characterized in terms of T-vectors.

Proposition 11.10. *Let N be a P/T net.*

- *N is repetitive if and only if it admits either a T-invariant or a T-increasing of strictly positive integers* $\boldsymbol{y} \in \mathbb{N}_+^n$*, i.e., a T-increasing whose support contains all transitions.*
- *N is consistent if and only if it admits a T-invariant of strictly positive integers* $\boldsymbol{y} \in \mathbb{N}_+^n$*, i.e., a T-invariant whose support contains all transitions.*

As an example the net in Fig. 11.8(b) is repetitive and consistent. If such a net is modified assuming that the multiplicity of the arc from t' to p' is equal to 2, the resulting net is repetitive but not consistent.

11.4.4 Structural Liveness

Definition 11.10. *A P/T net N is* structurally live *if there exists an initial marking* $\boldsymbol{m}_0$ *such that the marked net* $\langle N, \boldsymbol{m}_0 \rangle$ *is live.*

It is possible to give a necessary condition for such property.

Proposition 11.11. *A P/T net N with incidence matrix* $\boldsymbol{C}$ *is structurally live only if it does not admit a P-decreasing .*

As an example, the net in Fig. 11.2(*a*) has a P-decreasing $\boldsymbol{x}_D = [0\ 0\ 1]^T$, i.e., the number of tokens in p_3 can never increase. This net is structurally dead: regardless of the initial marking, each time t_2 fires the number of tokens in p_3 decreases and when the place gets empty, t_2 becomes dead.

11.5 Implicit Places

We now introduce the notion of *implicit place* that is useful in a large variety of problems, such as simulation, performance evaluation and deadlock analysis.

Definition 11.11. [21] *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a PN system with $N = (P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post})$. Place p is* implicit *if and only if $L(N, \boldsymbol{m}_0) = L(N', \boldsymbol{m}'_0)$ where $N' = (P, T, \boldsymbol{Pre}[P,T], \boldsymbol{Post}[P,T])$ is the restriction of N to P, i.e., the net obtained from N removing p and its input and output arcs, and $\boldsymbol{m}'_0 = \boldsymbol{m}_0[P]$ is the projection of $\boldsymbol{m}_0$ on P.*

In simple words, a place p of a marked PN is said to be *implicit* if deleting it does not change the "behavior" of the marked net, i.e., the language it can generate. Therefore a place p is implicit if there does not exist a marking $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$ and a transition $t \in T$ such that $\boldsymbol{m}[P] \geq \boldsymbol{Pre}[P,t]$ and $m[p] < Pre[p,t]$.

As an example, both places p_1 and p_2 in Fig. 11.5(a) are implicit places.

The following theorem provides a sufficient condition for a place p to be implicit.

Theorem 11.1. [21] *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a PN system with $N = (P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post})$. Let*

$$\gamma^* = \min \left\{ \boldsymbol{y}^T \cdot \boldsymbol{m}_0[P] + \mu \;\middle|\; \begin{array}{l} \boldsymbol{y}^T \cdot \boldsymbol{C}[P,T] \leq \boldsymbol{C}[p,T] \\ \boldsymbol{y}^T \cdot \boldsymbol{Pre}[P,p^\bullet] + \mu \cdot \boldsymbol{1}^T \geq \boldsymbol{Pre}[p,p^\bullet] \\ \boldsymbol{y} \geq \boldsymbol{0},\ \mu \geq 0 \end{array} \right\}. \tag{11.3}$$

If $m_0[p] \geq \gamma^*$, then p is implicit.

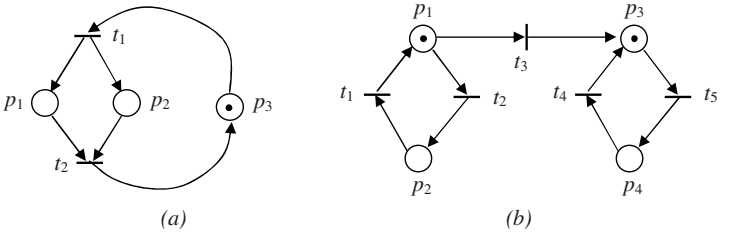


Fig. 11.5 (a) A PN with two implicit places; (b) a PN with a siphon and a trap

If a place p can be made implicit for every possible initial marking then it is called *structurally implicit*.

Definition 11.12. [21] *Let $N = (P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post})$ be a PN. Place p is* structurally implicit *if for every $\boldsymbol{m}_0[P]$, there exists a $m_0[p]$ such that p is implicit in $\langle N, \boldsymbol{m}_0[P \cup \{p\}] \rangle$.*

Both places p_1 and p_2 in Fig. 11.5(a) are also structurally implicit. An example of a place that is implicit but not structurally implicit is given in [21].

A characterization of structurally implicit places is the following:

Theorem 11.2. [21] *Let $N = \langle P \cup \{p\}, T, \boldsymbol{Pre}, \boldsymbol{Post} \rangle$. Place p is* structurally implicit *iff (equivalently):*

- $\exists\, \boldsymbol{y} > \boldsymbol{1}$ *such that* $\boldsymbol{C}[p,T] \geq \boldsymbol{y}^T \cdot \boldsymbol{C}[P,T]$.
- $\nexists\, \boldsymbol{x} \geq \boldsymbol{1}$ *such that* $\boldsymbol{C}[P,T] \cdot \boldsymbol{x} \geq \boldsymbol{0}$ *and* $\boldsymbol{C}[p,T] < \boldsymbol{0}$.

Structurally implicit places are very important when performing structural analysis. Many approaches in this framework are based on either the addition or the removal of structurally implicit places. Indeed, the *addition* of structurally implicit places can: (i) increase the Hamming distance useful for error/fault detecting and error/fault correcting codes [19]; (ii) decompose the system for computing performance evaluation (divide and conquer techniques) [22]; (iii) decompose the system for decentralized control [25]; (iv) cut spurious (deadlock) solutions improving the characterization of the state equation [1]. On the contrary, the *removal* of structurally implicit places can: (i) simplify the implementation having less nodes and (ii) improve the simulation of continuous PNs under infinite server semantics [16] .

Note that other restrictions of the concept of implicit place have been proposed in the literature [21], e.g., that of *concurrent implicit place* that is particularly useful in performance evaluation or control of timed models. For a detailed study on this, we address to [1].

11.6 Siphons and Traps

Let us now introduce two structural complementary objects, namely *siphons* and *traps*. Siphons and traps are usually introduced for ordinary nets and most of the literature dealing with them refers to such a restricted class of nets. However, some authors extended their definitions to non ordinary nets. See e.g. [23] and the references therein.

Definition 11.13. *A* siphon *of an ordinary PN is a set of places $S \subseteq P$ such that the set of input transitions of S is included in the set of output transitions of S, i.e.,*

$$\bigcup_{p \in S} {}^{\bullet}p \subseteq \bigcup_{p \in S} p^{\bullet}.$$

A siphon is minimal *if it is not the superset of any other siphon.*

Definition 11.14. *A* trap *of an ordinary PN is a set of places $S \subseteq P$ such that the set of output transitions of S is included in the set of input transitions of S, i.e.,*

$$\bigcup_{p \in S} p^{\bullet} \subseteq \bigcup_{p \in S} {}^{\bullet}p.$$

A trap is minimal *if it is not the superset of any other trap.*

The main interest on siphons and traps derives from the following two considerations. Once a siphon becomes empty, it remains empty during all the future evolutions of the net. Once a trap becomes marked, it remains marked during all the future evolutions of the net.

Consider the net in Fig. 11.5(b). The set $S = \{p_1, p_2\}$ is a siphon. The token initially in p_2 may move to p_1 and again to p_2, through the firing of t_1 and t_2,

respectively. However, once t_3 fires, S becomes empty and remains empty during all future evolutions of the net. On the contrary, $S' = \{p_3, p_4\}$ is a trap. Once a token enters in p_3, it can only move to p_4 and to p_3 again, but it will never leave S'.

Traps and siphons have been extensively used for the structural analysis of PNs. Following [21], here we limit to mention three of the most significant results in this context.

- In an ordinary deadlocked system, the subset of unmarked places is a siphon, otherwise one of its input transitions would be enabled.
- Taking into account that traps remain marked, if every siphon of an ordinary net contains an initially marked trap, then the system is deadlock-free.
- If $\boldsymbol{m}$ is a home state of a live system, then every trap must be marked, otherwise once the trap becomes marked — and it will eventually do by liveness — $\boldsymbol{m}$ cannot be reached any more.

Linear algebraic characterizations of siphons and traps have been given since the nineties [21] and can be considered as the starting point for extensive and fruitful theories on liveness analysis and deadlock prevention, particularly in the case of some net subclasses [4, 5, 6, 11, 24].

11.7 Classes of P/T Nets

The P/T net definition given in the previous chapter corresponds to a model sometimes called *general P/T net*. In this section we present some classes of P/T nets that satisfy particular structural conditions. In particular, the considered classes are summarized in the Venn diagram in Fig. 11.6.

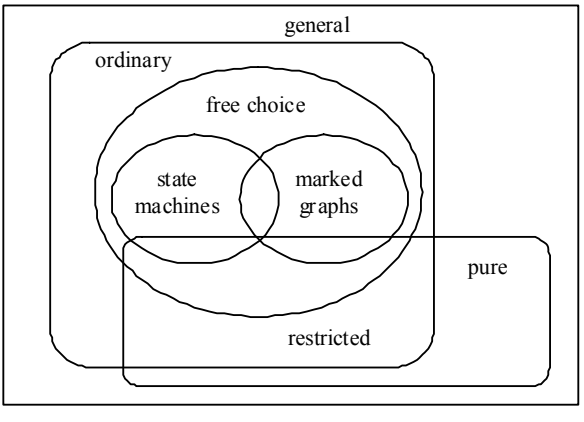


Fig. 11.6 Venn diagram of the different classes of P/T nets

11.7.1 Ordinary and Pure Nets

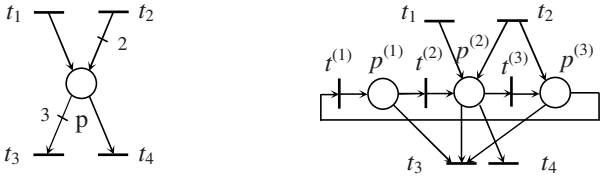
Definition 11.15. *A P/T net $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ is called:*

- ordinary *if* $\boldsymbol{Pre} : P \times T \longrightarrow \{0,1\}$ *and* $\boldsymbol{Post} : P \times T \longrightarrow \{0,1\}$*, i.e., if all arcs have unitary multiplicity;*
- pure *if for each place p and transition t it holds that $Pre[p,t] \cdot Post[p,t] = 0$, i.e., if the net has no self-loop;*
- restricted *if it is ordinary and pure.*

Even if the definition of restricted net seems more restrictive than that of a general P/T net, it is possible to prove that the two formalisms have the same modeling power, in the sense that they can describe the same class of systems.

In the following we present a rather intuitive construction to convert a general net into an equivalent[4] restricted one, by removing arcs with multiplicity greater than one and self-loops.

The first construction, shown in Fig. 11.7, removes arcs with a multiplicity greater than one. Let r be the maximum multiplicity among all "pre" and "post" arcs incident on place p. We replace p with a cycle of r places $p^{(i)}$ and transitions $t^{(i)}$, $i = 1, \ldots, r$, as in the figure. Each arc "post" ("pre") with multiplicity $k \leq r$ is replaced by k arcs, each one directed (coming from) k different places. Disregarding the firing of transitions $t^{(i)}$ and keeping into account that for each marking $\boldsymbol{m}$ of the original net and a corresponding marking $\boldsymbol{m}'$ of the transformed net it holds that $m[p] = \sum_{i=1}^{r} m'[p^{(i)}]$, the two nets have the same behavior.



(*a*) (*b*)

Fig. 11.7 (*a*) A place of a non-ordinary net; (*b*) transformation to eliminate the arcs of non-unitary multiplicity

The second construction, shown in Fig. 11.8, removes a self-loop from t to p: the firing of t in the original net corresponds to the firing of t' and t'' in the modified net.

[4] Here the term *equivalent* is used in a purely qualitative fashion: a formal discussion of model equivalence (e.g., in terms of languages, bisimulation, etc.) goes beyond the scope of this book.

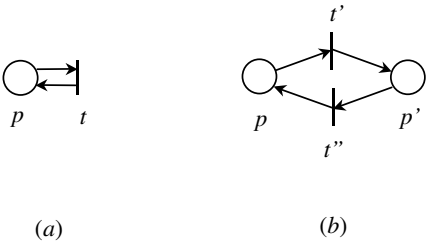


Fig. 11.8 (*a*) A self-loop; (*b*) transformation to remove the self-loop

11.7.2 Acyclic Nets

Definition 11.16. *A P/T net is* acyclic *if its underlying graph is acyclic, i.e., it does not contain directed cycles.*

The net in Fig. 11.3 is acyclic. On the contrary, the net in Fig. 11.1 is not acyclic because it contains the self-loop $p_2t_2p_2$. Analogously, the nets in Fig. 11.2 are not acyclic because they both contain cycle $p_1t_1p_2t_2p_1$.

The main feature of acyclic nets is that their state equation has no spurious solutions.

Proposition 11.12. [13] *Let N be an acyclic net. For all initial markings $\boldsymbol{m}_0$, it holds that $R(N,\boldsymbol{m}_0) = PR(N,\boldsymbol{m}_0)$.*

Therefore for this class of nets, the analysis based on the state equation provides necessary and sufficient conditions to solve the reachability problem.

11.7.3 State Machines

Definition 11.17. *A* state machine *is an ordinary net whose transitions have exactly one input and one output arc, i.e., it holds that $\sum_{p\in P} Pre[p,t] = \sum_{p\in P} Post[p,t] = 1$ for all transitions $t \in T$.*

The net in Fig. 11.9(*a*) is a state machine, while all the other nets in the same figure are not.

A state machine with a single token is analogous to a finite state automaton: each place of the net corresponds to a state of the automaton and the position of the token denotes the current state of the automaton. Since each place can have more than one output transition, as place p_1 in Fig. 11.9(*a*), state machines can model a "choice".

The initial marking can also assign to a state machine a number of tokens greater than one. In such a case it is possible to represent a limited form of "parallelism", that originates from the firing of transitions enabled by different tokens. On the

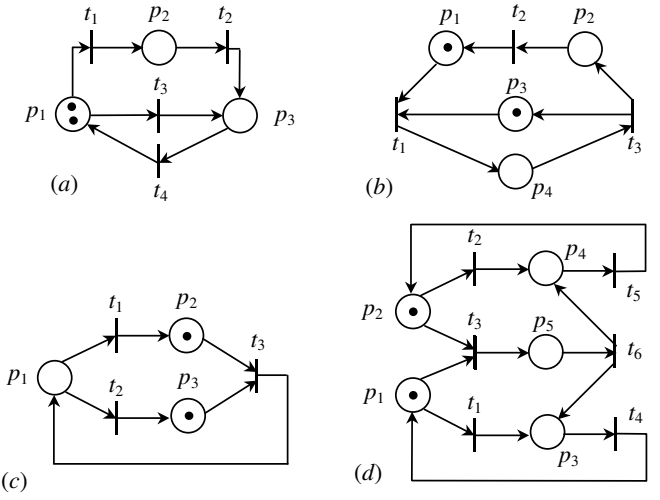


Fig. 11.9 Ordinary PNs: (*a*) a state machine; (*b*) a marked graph; (*c*) a free-choice net; (*d*) a non free-choice net

contrary, it is not possible to model a "synchronization" since the enabling of a transition depends on a single place.

The particular structure of state machines leads to a more restrictive model than ordinary nets, in the sense that it is not possible in general to find a state machine that is equivalent to an arbitrary ordinary net. Nevertheless, such restriction allows to significantly simplify the study of their properties [14, 17]. In particular, the following two important results can be proved.

- A state machine is always bounded.
- If a state machine is connected (but not necessarily strictly connected), then for all initial markings $\boldsymbol{m}_0$, it holds that $R(N,\boldsymbol{m}_0) = PR(N,\boldsymbol{m}_0)$, i.e., a marking is reachable if and only if it is potentially reachable.

11.7.4 Marked Graphs

Definition 11.18. *A* marked graph*, also called* marked event *(or* synchronization*)* graph*, is an ordinary net whose places have exactly one input and one output transition, i.e.,* $\sum_{t\in T} Pre[p,t] = \sum_{t\in T} Post[p,t] = 1$ *for all places* $p \in P$.

The net in Fig. 11.9(*b*) is a marked graph, while all the other nets in the same figure are not.

Since each place of a marked graph has a single output transition, this structure cannot model a "choice". However, it can model "parallelism" because a transition may have more than one output place. Moreover, it can model a "synchronization"

since the enabling state of a transition can depend on several places; such is the case of transition t_1 in Fig. 11.9(b).

There exists a dual relation between state machines and marked graphs.

Definition 11.19. *Given a net* $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ *with m places and n transitions, the* dual *net of N is the net* $N^T = (T, P, \boldsymbol{Pre}^T, \boldsymbol{Post}^T)$ *with n places and m transitions.*

The dual net can be obtained from the original one simply replacing each node "place" with a node "transition" and viz., and inverting the directions of arcs. If the original net has incidence matrix $\boldsymbol{C}$, the dual net has incidence matrix $\boldsymbol{C}^T$. Moreover, if N' is the dual net of N, then N is the dual net of N'.

The two nets in Fig. 11.9(a)-(b) are one the dual net of the other.

Proposition 11.13. *If N is a state machine (resp., marked graph) its dual net* N^T *is a marked graph (resp., state machine).*

Proof. The construction of the dual net transforms each node "place" into a node "transition" and viz., and does not change the multiplicity of the arcs but only their orientation. Thus, if N is a state machine each transition has a single input place and a single output place and in N^T each place has a single input transition and a single output transition, thus the resulting net is a marked graph. A similar reasoning applies if N is a marked graph. □

As in the case of state machines, also for marked graphs some important properties can be proved. For a detailed discussion on this we address to [14].

11.7.5 Choice-Free Nets

A generalization of state machines and marked graphs is the following.

Definition 11.20. *A* free-choice net *is an ordinary net such that from* $p \in P$ *to* $t \in T$ *is either the single output arc from p or the single input arc to t, i.e., for all places* $p \in P$ *if* $Pre[p,t] = 1$ *it holds that:*

$$[\forall t' \neq t : Pre[p,t'] = 0] \ \vee \ [\forall p' \neq p : Pre[p',t] = 0].$$

The admissible structures in a free-choice net are shown in Fig. 11.10(a)-(b) while a non-admissible structure is shown in Fig. 11.10(c).

A free-choice net thus can model "choice", "parallelism" and "synchronization". As an example, in the free-choice net in Fig. 11.9(c) transitions t_1 and t_2 are in structural conflict and model a choice in place p_1; t_3 models a synchronization.

However, a free-choice net cannot represent "choice" and "synchronization" relatively to the same transition. As an example, the net in Fig. 11.9(d) is not a free-choice net since transition t_3 models a synchronization and at the same time one of the admissible choices for place p_1.

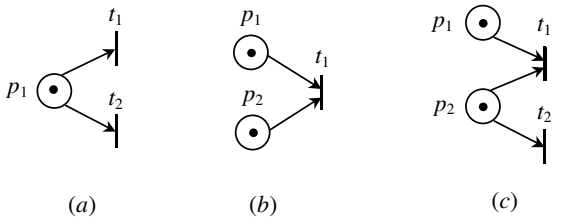


Fig. 11.10 Elementary structures: (*a*)-(*b*) free-choice; (*c*) not free-choice

Note that each state machine and each marked graph are also free-choice net. However, the class of free-choice net is larger than the union of these two classes. The net in Fig. 11.9(*c*) is a free-choice net even if it is neither a state machine nor a marked graph. Also, free-choice nets satisfy particular conditions that allow to reduce the computational complexity of the analysis of their properties with respect to the case of ordinary nets. A rich literature exists on this topic. See e.g. [3].

11.8 Further Reading

As in the case of Chapter 10, further details on the proposed topics can be found in the survey paper by Murata [13] and on the books of Peterson [14] and David and Alla [2]. Moreover, for more details on methods to improve the state equation based on implicit places, we address to the paper by Silva *et al.* [21]. Significant results related to the analysis of structural boundedness and structural liveness can be found in [20, 21], most of which are based on rank theorems. Finally, very interesting results on deadlock analysis and prevention are summarized in the book of Li and Zhou [10] and in the book edited by Zhou and Fanti [7].

References

1. Colom, J.M., Silva, M.: Improving the Linearly Based Characterization of P/T Nets. In: Rozenberg, G. (ed.) APN 1990. LNCS, vol. 483, pp. 113–145. Springer, Heidelberg (1991)
2. David, R., Alla, H.: Discrete, Continuous and Hybrid Petri Nets. Springer (2005)
3. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995)
4. Ezpeleta, J., Colom, J.M., Martinez, J.: A Petri net based deadlock prevention policy for flexible manufacturing systems. IEEE Transactions on Robotics and Automation 11(2), 173–184 (1995)
5. Ezpeleta, J., Recalde, L.: A deadlock avoidance approach for nonsequential resource allocation systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 34(1), 93–101 (2004)

6. Ezpeleta, J., Valk, R.: A polynomial deadlock avoidance method for a class of nonsequential resource allocation systems. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 36(6), 1234–1243 (2006)
7. Fanti, M.P., Zhou, M.: Deadlock Resolution in Computer-Integrated Systems. Marcel Dekker/CRC Press (2005)
8. Finkel, A., Johnen, C.: The home state problem in transition systems. In: Rapport de Recherche, vol. 471. Univ. de Paris-Sud., Centre d'Orsay (1989)
9. Karp, R., Miller, C.: Parallel program schemata. Journal of Computer and System Sciences 3(2), 147–195 (1969)
10. Li, Z., Zhou, M.: Deadlock Resolution in Automated Manufacturing Systems. Springer (2009)
11. Liao, H., Lafortune, S., Reveliotis, S., Wang, Y., Mahlke, S.: Synthesis of maximally-permissive liveness-enforcing control policies for Gadara Petri nets. In: 49th IEEE Conference on Decision and Control, Atlanta, USA (2010)
12. Martinez, J., Silva, M.: A simple and fast algorithm to obtain all invariants of a generalized Petri net. In: Informatik-Fachberichte: Application and Theory of Petri Nets, vol. 52. Springer (1982)
13. Murata, T.: Petri nets: properties, analysis and applications. Proceedings IEEE 77(4), 541–580 (1989)
14. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall (1981)
15. Petri, C.A.: Kommunication Mit Automaten. Institut für Instrumentelle, Mathematik (Bonn, Germany) Schriften des IIM 3 (1962)
16. Recalde, L., Mahulea, C., Silva, M.: Improving analysis and simulation of continuous Petri nets. In: 2nd IEEE Conf. on Automation Science and Engineering, Shanghai, China (2006)
17. Reisig, W.: Petri Nets: an Introduction. EATCS Monographs on Theoretical Computer Science (1985)
18. Reutenauer, C.: Aspects Mathématiques des Réseaux de Petri. Prentice-Hall International (1990)
19. Silva, M., Velilla, S.: Error detection and correction on Petri net models of discrete control systems. In: IEEE Int. Symp. on Circuits and Systems, Kyoto, Japan (1985)
20. Silva, M., Recalde, L., Teruel, E.: On linear algebraic techniques for liveness analysis of P/T systems. Journal of Circuits, Systems and Computers 8(11), 223–265 (1998)
21. Silva, M., Teruel, E., Colom, J.M.: Linear Algebraic and Linear Programming Techniques for the Analysis of Net Systems. In: Reisig, W., Rozenberg, G. (eds.) APN 1998. LNCS, vol. 1491, pp. 309–373. Springer, Heidelberg (1998)
22. Silva, M., Campos, J.: Performance evaluation of DEDS with conflicts and synchronizations: net-driven decomposition techniques. In: 4th Int. Workshop on Discrete Event Systems, Cagliari, Italy (1998)
23. Tricas, F., Ezpeleta, J.: Some results on siphon computation for deadlock prevention in resource allocation systems modeled with Petri nets. In: IEEE Conf. on Emerging Technologies and Factory Automation, Lisbon, Portugal (2003)
24. Tricas, F., Ezpeleta, J.: Computing minimal siphons in Petri net models of resource allocation systems: a parallel solution. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 36(3), 532–539 (2006)
25. Wang, L., Mahulea, C., Julvez, J., Silva, M.: Decentralized control of large scale systems modeled with continuous marked graphs. In: 18th IFAC World Congress, Milano, Italy (2011)

Chapter 12
Supervisory Control of Petri Nets with Language Specifications

Alessandro Giua

12.1 Introduction

In this chapter, we study Petri nets (PNs) as language generators and we show how PNs can be used for supervisory control of discrete event systems under language specifications.

Supervisory control, originated by the work of Ramadge and Wonham [13], is a system theory approach that has been gaining increasing importance because it provides a unifying framework for the control of Discrete Event Systems (DESs). A general overview of Supervisory Control has been presented in Chapter 3.

In the original work of Ramadge and Wonham finite state machines (FSMs) were used to model plants and specifications. FSMs provide a general framework for establishing fundamental properties of DES control problems. They are not convenient models to describe complex systems, however, because of the large number of states that have to be introduced to represent several interacting subsystems, and because of the lack of structure. More efficient models have been proposed in the DES literature. Here the attention will be drawn to Petri net models.

PNs have several advantages over FSMs. First, PNs have a higher language complexity than FSM, since Petri net languages are a proper superset of regular languages. Second, the states of a PN are represented by the possible markings and not by the places: thus they give a compact description, i.e., the structure of the net may be maintained small in size even if the number of the markings grows[1]. Third, PNs can be used in modular synthesis, i.e., the net can be considered as composed

Alessandro Giua
Department of Electrical and Electronic Engineering, University of Cagliari,
Piazza d'Armi, 09123 Cagliari, Italy
e-mail: giua@diee.unica.it

[1] However, we should point out that many analysis techniques for Petri nets are based on the construction of the reachability graph, that suffers from the same state explosion problem typical of automata. To take advantage of the compact PN representation, other analysis techniques (e.g. structural) should be used.

C. Seatzu et al. (Eds.): Control of Discrete-Event Systems, LNCIS 433, pp. 235–255.
springerlink.com

of interrelated subnets, in the same way as a complex system can be regarded as composed of interacting subsystems.

Although PNs have a greater modeling power than FSMs, computability theory shows that the increase of modeling power often leads to an increase in the computation required to solve problems. This is why a section of this paper focuses on the decidability properties of Petri nets by studying the corresponding languages: note that some of these results are original and will be presented with formal proofs. It will be shown that Petri nets represent a good tradeoff between modeling power and analysis capabilities

The chapter is structured as follows. In Section 1 Petri net generators and languages are defined. In Section 2 the concurrent composition operator on languages is defined and extended to an operator on generators. In Section 3 it is shown how the classical monolithic supervisory design can be carried out using Petri net models. Finally, in Section 4 some issues arising from the use of unbounded PNs in supervisory control are discussed.

12.2 Petri Nets and Formal Languages

This section provides a short but self-standing introduction to Petri net languages. PN languages represent an interesting topic within the broader domain of formal language theory but there are few books devoted to this topic and the relevant material is scattered in several journal publications. In this section and in the following we focus on the definition of Petri net generators and operators that will later be used to solve a supervisory control problem.

12.2.1 Petri Net Generators

Definition 12.1. *A* labeled Petri net system *(or* Petri net generator*) [7, 12] is a quadruple* $G = (N, \ell, \boldsymbol{m}_0, F)$ *where:*

- $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ *is a* Petri net structure *with* $|P| = m$ *and* $|T| = n$*;*
- $\ell : T \to E \cup \{\lambda\}$ *is a* labeling function *that assigns to each transition a label from the alphabet of events* E *or assigns the empty word*[2] λ *as a label;*
- $\boldsymbol{m}_0 \in \mathbb{N}^n$ *is an* initial marking*;*
- $F \subset \mathbb{N}^n$ *is a finite set of* final markings*.*

Three different types of *labeling functions* are usually considered.

- *Free labeling:* all transitions are labeled distinctly and none is labeled λ, i.e., $(\forall t, t' \in T)\ [t \neq t' \Longrightarrow \ell(t) \neq \ell(t')]$ and $(\forall t \in T)\ [\ell(t) \neq \lambda]$.

[2] While in other parts of this book the empty string is denoted ε, in this section we have chosen to use the symbol λ for consistency with the literature on PN languages.

- λ-*free labeling:* no transition is labeled λ.
- *Arbitrary labeling:* no restriction is posed on ℓ.

The labeling function may be extended to a function $\ell : T^* \to E^*$ defining: $\ell(\lambda) = \lambda$ and $(\forall t \in T, \forall \sigma \in T^*)\ \ell(\sigma t) = \ell(\sigma)\ell(t)$.

Example 12.1. Consider the nets in Fig. 12.1 where the label of each transition is shown below the transition itself. Net (a) is a free-labeled generator on alphabet $E = \{a, b\}$. Nets (b) and (c) are λ-free generators on alphabet $E = \{a\}$. Net (d) is an arbitrary labeled generator on alphabet $E = \{a\}$. ■

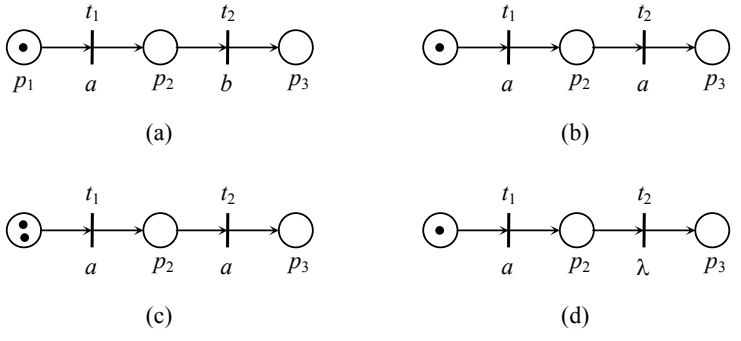


Fig. 12.1 PN generators of Example 12.1

Three languages are associated with a generator G depending on the different notions of terminal strings.

- *L-type* or *terminal language*:[3] the set of strings generated by firing sequences that reach a final marking, i.e.,

$$L_L(G) = \{\ell(\sigma) \mid \boldsymbol{m}_0\ [\sigma\rangle\ \boldsymbol{m}_f \in F\}.$$

- *G-type* or *covering language* or *weak language:* the set of strings generated by firing sequences that reach a marking $\boldsymbol{m}$ covering a final marking, i.e.,

$$L_G(G) = \{\ell(\sigma) \mid \boldsymbol{m}_0\ [\sigma\rangle\ \boldsymbol{m} \geq \boldsymbol{m}_f \in F\}.$$

- *P-type* or *prefix language*:[4] the set of strings generated by any firing sequence, i.e.,

$$L_P(G) = \{\ell(\sigma) \mid \boldsymbol{m}_0\ [\sigma\rangle\ \}.$$

[3] This language is called marked behavior in the framework of Supervisory Control and is denoted $L_m(G)$.

[4] This language is called closed behavior in the framework of Supervisory Control and is denoted $L(G)$.

Example 12.2. Consider the free-labeled generator G in Fig. 12.2. The initial marking, also shown in the figure, is $\boldsymbol{m}_0 = [1\ 0\ 0]^T$. Assume the set of final markings is $F = \{[0\ 0\ 1]^T\}$. The languages of this generator are:

$$L_L(G) = \{a^m c b^m \mid m \geq 0\};$$

$$L_G(G) = \{a^m c b^n \mid m \geq n \geq 0\};$$

$$L_P(G) = \{a^m \mid m \geq 0\} \cup \{a^m c b^n \mid m \geq n \geq 0\}.$$ ■

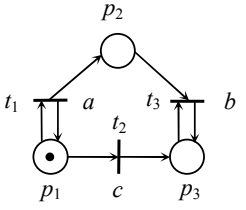


Fig. 12.2 Free-labeled generator G of Example 12.2

12.2.2 Deterministic Generators

A *deterministic* PN generator [7] is such that the word of events generated from the initial marking uniquely determines the marking reached.

Definition 12.2. *A λ-free generator G is* deterministic *iff for all $t, t' \in T$, with $t \neq t'$, and for all $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$: $\boldsymbol{m}\,[t\rangle \wedge \boldsymbol{m}\,[t'\rangle \Longrightarrow \ell(t) \neq \ell(t')$.*

According to the previous definition, in a deterministic generator two transitions sharing the same label may never be simultaneously enabled and no transition may be labeled by the empty string. Note that a free-labeled generator is also deterministic. On the contrary, a λ-free (but not free labeled) generator may be deterministic or not depending on its structure and also on its initial marking.

Example 12.3. Consider generators (b) and (c) in Fig. 12.1: they have the same net structure and the same λ-free labeling, but different initial marking. The first one is deterministic, because transitions t_1 and t_2, sharing label a can never be simultaneously enabled. On the contrary, the second one is not deterministic, because reachable marking $[1\ 1\ 0]^T$ enables both transitions t_1 and t_2: as an example, the observed word aa may be produced by two different sequences yielding two different markings

$$\begin{bmatrix}2\\0\\0\end{bmatrix}\ [t_1\rangle\ \begin{bmatrix}1\\1\\0\end{bmatrix}\ [t_1\rangle\ \begin{bmatrix}0\\2\\0\end{bmatrix} \qquad or \qquad \begin{bmatrix}2\\0\\0\end{bmatrix}\ [t_1\rangle\ \begin{bmatrix}1\\1\\0\end{bmatrix}\ [t_2\rangle\ \begin{bmatrix}1\\0\\1\end{bmatrix}.$$ ■

The previous definition of determinism was introduced in [18] and used in [7, 12]. It may be possible to extend it as follows.

Definition 12.3. *A λ-free generator G is* deterministic *iff for all $t,t' \in T$, with $t \neq t'$, and for all $\boldsymbol{m} \in R(N,\boldsymbol{m}_0)$: $\boldsymbol{m}\,[t\rangle \wedge \boldsymbol{m}\,[t'\rangle \Longrightarrow [\ell(t) \neq \ell(t')] \vee [\boldsymbol{Post}[\cdot,t] - \boldsymbol{Pre}[\cdot,t] = \boldsymbol{Post}[\cdot,t'] - \boldsymbol{Pre}[\cdot,t']]$.*

With this extended definition, we accept as deterministic a generator in which two transitions with the same label may be simultaneously enabled at a marking $\boldsymbol{m}$, provided that the two markings reached from $\boldsymbol{m}$ by firing t and t' are the same. Note that with this extended definition, while the word of events generated from the initial marking uniquely determines the marking reached it does not necessarily uniquely determine the sequences that have fired.

12.2.3 Classes of Petri Net Languages

The classes of Petri net languages are denoted as follows:

- $\mathscr{L}^f$ (resp. $\mathscr{G}^f$, $\mathscr{P}^f$) denotes the class of terminal (resp. covering, prefix) languages generated by free-labeled PN generators.
- $\mathscr{L}_d$ (resp. $\mathscr{G}_d$, $\mathscr{P}_d$) denotes the class of terminal (resp. covering, prefix) languages generated by deterministic PN generators.
- $\mathscr{L}$ (resp. $\mathscr{G}$, $\mathscr{P}$) denotes the class of terminal (resp. covering, prefix) languages generated by λ-free PN generators.
- $\mathscr{L}^\lambda$ (resp. $\mathscr{G}^\lambda$, $\mathscr{D}^\lambda$, $\mathscr{P}^\lambda$) denotes the class of terminal (resp. covering, prefix) languages generated by arbitrary labeled PN generators.

Table 12.1 shows the relationship among these classes. Here $A \to B$ represents a strict set inclusion $A \subsetneq B$.

Table 12.1 Known relations among classes of Petri net languages. An arc $\to$ represents the set inclusion

$$\begin{array}{ccccccc}
\mathscr{L}^f & \to & \mathscr{L}_d & \to & \mathscr{L} & \to & \mathscr{L}^\lambda \\
 & & & & \uparrow & & \uparrow \\
\mathscr{G}^f & \to & \mathscr{G}_d & \to & \mathscr{G} & \to & \mathscr{G}^\lambda \\
\uparrow & & \uparrow & & \uparrow & & \uparrow \\
\mathscr{P}^f & \to & \mathscr{P}_d & \to & \mathscr{P} & \to & \mathscr{P}^\lambda
\end{array}$$

While a formal proof of all these relations can be found in [1], we point out that the relations on each line — that compare the same type of languages of nets with different labeling — are rather intuitive. Additionally, one readily understands that any *P-type* language of a generator G may also be obtained as a *G-type* language defining as a set of final markings $F = \{\mathbf{0}\}$.

Parigot and Peltz [10] have defined PN languages as regular languages with the additional capability of determining if a string of parenthesis is well formed.

If we consider the class $\mathscr{L}$ of PN languages, it is possible to prove [12] that $\mathscr{L}$ is a strict superset of regular languages and a strict subset of context-sensitive languages. Furthermore, $\mathscr{L}$ and the class of context-free languages are not comparable. An example of a language in $\mathscr{L}$ that is not context-free is: $L = \{a^m b^m c^m \mid m \geq 0\}$. An example of a language that is context-free but is not in $\mathscr{L}$ is: $L = \{ww^R \mid w \in E^*\}$[5] if $|E| > 1$.

All these results are summarized in Fig. 12.3. Note that the class $\mathscr{L}_d$, although contained in $\mathscr{L}$, occupies the same position as $\mathscr{L}$ in the hierarchy shown in the figure.

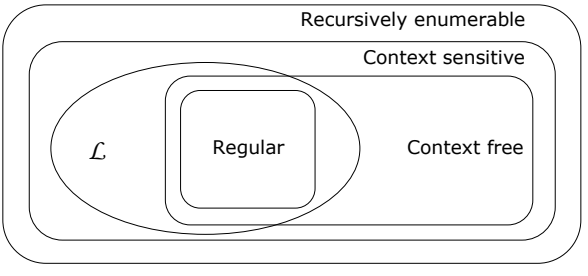


Fig. 12.3 Relations among the class $\mathscr{L}$ and other classes of formal languages

In the framework of Supervisory Control, we will assume that the generators considered are deterministic. In particular, class $\mathscr{L}_d$ (or possibly $\mathscr{G}_d$ for unbounded nets) will be used to describe marked languages, while class $\mathscr{P}_d$ will be used to describe closed languages. There are several reasons for this choice.

- Systems of interest in supervisory control theory are deterministic.
- Although each class of deterministic languages defined here is strictly included in the corresponding class of λ-free languages, it is appropriate to restrict our analysis to deterministic generators. In fact, several properties of interest are decidable for deterministic nets while they are not for λ-free nets [1, 11, 18].
- In [1] it was shown that the classes $\mathscr{G}_d$ and $\mathscr{L}_d$ are incomparable, and furthermore $\mathscr{G}_d \cap \mathscr{L}_d = \mathscr{R}$, where $\mathscr{R}$ is the class of regular languages. Hence taking also into account the G-type language (in addition to the L-type language) one extends the class of control problems that can be modeled by deterministic unbounded PNs.

[5] The string w^R is the reversal of string w.

12.2.4 Other Classes of Petri Net Languages

Gaubert and Giua [1] have explored the use of infinite sets of final markings in the definition of the marked behavior of a net. With each more or less classical subclass of subsets of $\mathbb{N}^m$ — finite, ideal (or upper), semi-cylindrical, star-free, recognizable, rational (or semilinear) subsets — it is possible to associate the class of Petri net languages whose set of accepting states belongs to the class.

When comparing the related Petri net languages, it was shown that for arbitrary or λ-free PN generators, the above hierarchy collapses: one does not increase the generality by considering semilinear accepting sets instead of the usual finite ones. However, for free-labeled and deterministic PN generators, it is shown that one gets new distinct subclasses of Petri net languages, for which several decidability problems become solvable.

12.3 Concurrent Composition and System Structure

In this section we recall the definition of the concurrent composition operator on languages and introduce the corresponding operator on nets.

Definition 12.4 (Concurrent composition of languages). *Given two languages $L_1 \subseteq E_1^*$ and $L_2 \subseteq E_2^*$, their* concurrent composition *is the language L on alphabet $E = E_1 \cup E_2$ defined as follows:*

$$L = L_1 \parallel L_2 = \{\, w \in E^* \mid w\uparrow_{E_1} \in L_1,\ w\uparrow_{E_2} \in L_2 \,\}$$

where $w\uparrow_{E_i}$ denotes the projection of word w on alphabet E_i, for $i = 1,2$.

We now consider the counterpart of this language operator on a net structure.

Definition 12.5 (Concurrent composition of PN generators)
Let $G_1 = (N_1, \ell_1, \boldsymbol{m}_{0,1}, F_1)$ and $G_2 = (N_2, \ell_2, \boldsymbol{m}_{0,2}, F_2)$ be two PN generators. Their concurrent composition*, denoted also $G = G_1 \parallel G_2$, is the generator $G = (N, \ell, \boldsymbol{m}_0, F)$ that generates $L_L(G) = L_L(G_1) \parallel L_L(G_2)$ and $L_P(G) = L_P(G_1) \parallel L_P(G_2)$.*

The structure of G may be determined with the following procedure.

Algorithm 12.4. *Let P_i, T_i and E_i $(i = 1,2)$ be the place set, transition set, and the alphabet of G_i.*

- *The place set P of N is the union of the place sets of N_1 and N_2, i.e., $P = P_1 \cup P_2$.*
- *The transition set T of N and the corresponding labels are computed as follows:*
  - *For each transition $t \in T_1 \cup T_2$ labeled λ, a transition with the same input and output bag of t and labeled λ belongs to T.*

- *For each transition $t \in T_1 \cup T_2$ labeled $e \in (E_1 \setminus E_2) \cup (E_2 \setminus E_1)$, a transition with the same input and output bag of t and labeled e belongs to T.*
- *Consider a symbol $e \in E_1 \cap E_2$ and assume it labels m_1 transitions $T_{e,1} \subseteq T_1$ and m_2 transitions $T_{e,2} \subseteq T_2$. Then $m_1 \times m_2$ transitions labeled e belong to T. The input (output) bag of each of these transitions is the sum of the input (output) bags of one transition in $T_{e,1}$ and of one transition in $T_{e,2}$.*

- $\boldsymbol{m}_0 = [\boldsymbol{m}_{0,1}^T \ \boldsymbol{m}_{0,2}^T]^T$.
- *F is the cartesian product of F_1 and F_2, i.e., $F = \{[\boldsymbol{m}_1^T \ \boldsymbol{m}_2^T]^T \mid \boldsymbol{m}_1 \in F_1, \boldsymbol{m}_2 \in F_2\}$.*

The composition of more than two generators can be computed by repeated application of the procedure. Note that while the set of places grows linearly with the number of composed systems, the set of transitions and of final markings may grow faster.

Example 12.5. Let $G_1 = (N_1, \ell_1, \boldsymbol{m}_{0,1}, F_1)$ and $\mathbf{G_2} = (N_2, \ell_2, \boldsymbol{m}_{0,2}, F_2)$ be the two generators shown in Fig. 12.4. Here $F_1 = \{[1\ 0]^T\}$ and $F_2 = \{[1\ 0]^T, [0\ 1]^T\}$. Their concurrent composition $G = G_1 \parallel G_2$ is also shown in Fig. 12.4. The initial marking of G is $\boldsymbol{m}_0 = [1\ 0\ 1\ 0]^T$ and its set of final markings is $F = \{[1\ 0\ 1\ 0]^T, [1\ 0\ 0\ 1]^T\}$. ■

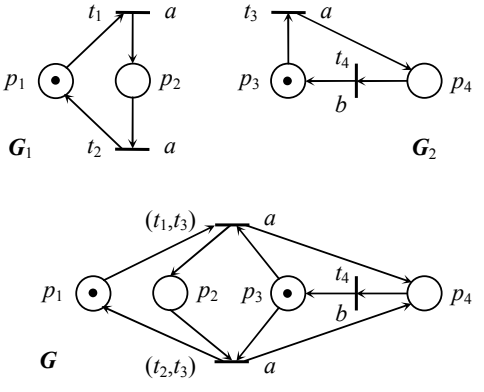


Fig. 12.4 Two generators G_1, G_2 and their concurrent composition G of Example 12.5

12.4 Supervisory Design Using Petri Nets

In this section we discuss how Petri net models may be used to design supervisors for language specifications within the framework of Supervisory Control. The design of a supervisor in the framework of automata was presented in Chapter 3 and we assume the reader is already familiar with this material.

12.4.1 Plant, Specification and Supervisor

Here we comment on some of the assumptions that are peculiar to the PN setting.

- The **plant** is described by a deterministic PN generator G on alphabet E. Its closed language is $L(G) = L_P(G)$ while its marked[6] language is $L_m(G) = L_L(G)$. We assume such a generator is *nonblocking*, i.e., $\overline{L_L(G)} = L_P(G)$.

  The transition set of G is partitioned as follows: $T = T_c \cup T_{uc}$, where T_c are the *controllable transitions* that can be disabled by a control agent, while T_{uc} are the *uncontrollable transitions*. Note that this allows a generalization of the automata settings where the notion of controllability and uncontrollability is associated to the events. In fact, it is possible that two transitions, say t' and t'', have the same event label $\ell(t') = \ell(t'') = e \in E$ but one of them is controllable while the other is not. In the rest of the chapter, however, we will not consider this case and assume that the event alphabet may be partitioned as $E = E_c \cup E_{uc}$ where

$$E_c = \bigcup_{t \in T_c} \ell(t), \quad E_{uc} = \bigcup_{t \in T_{uc}} \ell(t) \quad \text{and} \quad E_c \cap E_{uc} = \emptyset.$$

  It is also common to consider plants composed by m PN generators $G_1, \ldots, G_m$ working concurrently. The alphabets of these generators are $E_1, \ldots, E_m$. The overall plant is a PN generator $G = G_1 \parallel \cdots \parallel G_m$ on alphabet $E = E_1 \cup \cdots \cup E_m$.
- The **specification** is a language $K \subset \hat{E}^*$, where $\hat{E} \subset E$ is a subset of the plant alphabet. Such a specification defines a set of *legal words* on E given by $\{w \in E^* \mid w \uparrow_{\hat{E}} \in \text{prefix}(K)\}$.
  The specification K is represented by a deterministic nonblocking PN generator H on alphabet $\hat{E}$ whose marked language is $L_m(H) = L_L(H) = K$. As for the plant, other choices for the marked language are possible.
- The **supervisor**[7] is described by a nonblocking PN generator S on alphabet E. It runs in parallel with the plant, i.e., each time the plant generates an event e a transition with the same label is executed on the supervisor. The control law computed by S when its marking is $\boldsymbol{m}$ is given by $g(\boldsymbol{m}) = E_{uc} \cup \{e \in E_c \mid (\exists t \in T_c)\boldsymbol{m}[t\rangle, \ell(t) = e\}$.

12.4.2 Monolithic Supervisor Design

The *monolithic supervisory design* requires three steps. In the first step, a coarse structure for a supervisor is synthesized by means of concurrent composition of the plant and specification. In the second step, the structure is analyzed to check

[6] While in the case of bounded nets the L-type language can describe any marked language, in the case of unbounded generators other choices for the marked language are possible considering the G-type language of the generator or even any other type of terminal languages as mentioned in § 12.2.4. This will be discussed in Section 12.5.

[7] See also Definition 3.9 in Chapter 3.9.

if properties of interest (namely, the absence of uncontrollable and blocking states) hold. In the third step, if the properties do not hold, this structure is trimmed to avoid reaching undesirable states.

Algorithm 12.6. (Monolithic supervisory design). *We are given a plant G and a specification H.*

1. *Construct by concurrent composition the generator $J = G \parallel H$.*
2. *Determine if the generator J satisfies the following properties:*
   - nonblockingness*, i.e., it does not contain blocking markings from which a final marking cannot be reached;*
   - controllability*, i.e., it does not contain uncontrollable markings such that when G and H run in parallel an uncontrollable event is enabled in G but is not enabled in H.*

   If J satisfies both properties, then both H and J are suitable supervisors.
3. *If J contains blocking or uncontrollable markings, we have to trim it to obtain a nonblocking and controllable generator S. The generator S obtained through this procedure is at the same time a suitable maximally permissive supervisor and the corresponding closed-loop system.*

In the previous algorithm, the generator J constructed in step 1 represents the largest behavior of the plant that satisfies all the constraints imposed by the specifications. More precisely, its closed language

$$L(J) = \{w \in E \mid w \in L(G), w \uparrow_{\hat{E}} \in L(H)\}$$

represents the behavior of the plant restricted to the set of legal words, while its marked behavior

$$L_m(J) = \{w \in E \mid w \in L_m(G), w \uparrow_{\hat{E}} \in L_m(H)\}$$

represents the marked behavior of the plant restricted to the set of legal words marked by the specification.

In step 2 we have used informally the term "blocking marking" and "uncontrollable marking". We will formally define these notions in the following.

We first define some useful notation. The structure of the generators is $J = (N, \ell, \boldsymbol{m}_0, F)$, $G = (N_1, \ell_1, \boldsymbol{m}_{0,1}, F_1)$, and $H = (N_2, \ell_2, \boldsymbol{m}_{0,2}, F_2)$, where $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ and $N_i = (P_i, T_i, \boldsymbol{Pre}_i, \boldsymbol{Post}_i)$, $(i = 1, 2)$. We define the *projection of a marking $\boldsymbol{m}$ of N on net N_i*, $(i = 1, 2)$, denoted $\boldsymbol{m} \uparrow_i$, is the vector obtained from $\boldsymbol{m}$ by removing all the components associated to places not present in N_i.

We first present the notion of a blocking marking.

Definition 12.6. *A marking $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$ of generator J is a* blocking marking *if no final marking may be reached from it, i.e., $R(N, \boldsymbol{m}) \cap F = \emptyset$. The generator J is* nonblocking *if no blocking marking is reachable.*

We now present the notion of an uncontrollable marking.

Definition 12.7. *Let $T_u \subseteq T$ be the set of uncontrollable transitions of J. A marking $\boldsymbol{m} \in R(N, \boldsymbol{m}_0)$ of generator J is* uncontrollable *if there exists an uncontrollable transition $t \in T_u$ that is enabled by $\boldsymbol{m}\uparrow_1$ in G but that is not enabled by $\boldsymbol{m}\uparrow_2$ in H. The generator J is* controllable *if no uncontrollable marking is reachable.*

Determining if a generator J is nonblocking and controllable is always possible, as we will show in the next section. We also point out that for bounded nets this test can be done by construction of the reachability graph[8] as in the following example of supervisory design.

Example 12.7. Consider the generators G_1 and G_2, and the specification H in Fig. 12.5 (left). Note that all nets are free-labeled, hence we have an isomorphism between the set of transitions T and the set of events E: in the following each transitions will be denoted by the corresponding event.

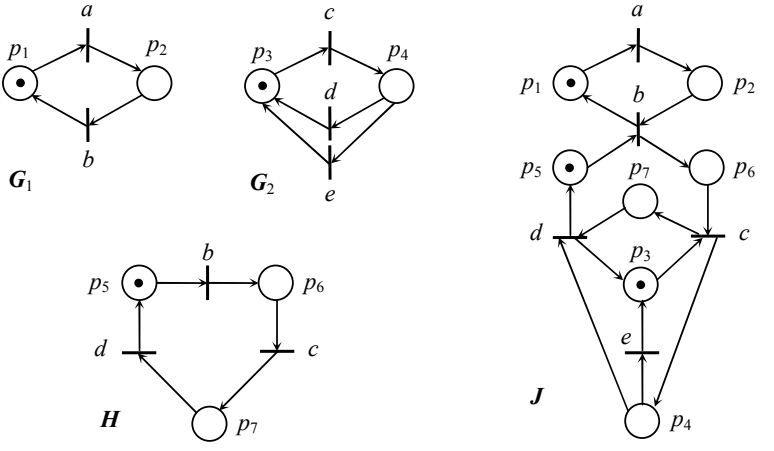


Fig. 12.5 Left: Systems G_1, G_2 and specification H for the control problem of Example 12.7. Right: System $J = G_1 \parallel G_2 \parallel H$

G_1 describes a conveyor that brings in a manufacturing cell a raw part (event a) that is eventually picked-up by a robot (event b) so that a new part can enter. G_2 describes a machine that is loaded with a raw part (event c) and, depending on the operation it performs, may produce parts of type A or type B (events d or e) before returning to the idle state. The set of final states of both generators consists of the initial marking shown in the figure.

The specification we consider, represented by the generator H, describes a cyclic operation process where a robot picks-up a raw part from the conveyor, loads it on

[8] As we have already pointed out, the construction of the reachability graph suffers from the state explosion problem. An open area for future research is the use of more efficient analysis techniques (e.g., structural) to check nonblockingness and controllability for language specification.

the machine and after recognizing that a part of type A has been produced repeats the process. The set of final states consists of the initial marking shown in the figure.

The overall process is $G = G_1 \parallel G_2$ and the generator $J = G \parallel H$, is shown in Fig. 12.5 (right). Its set of final states consists of the initial marking shown in the figure.

Assume now that the controllable transition/event set is $E_c = \{a, c, d, e\}$ and the uncontrollable transition/event set is $E_u = \{b\}$.

It is immediate to show that generator J is blocking and uncontrollable. To show this we have constructed the reachability graph of J in Fig. 12.6. The two markings shown in thick boxes are blocking because from them it is impossible to reach the initial marking (that is also the unique final marking). The three markings shaded in gray are uncontrollable: in fact, in all these markings $m[p_2] = 1$, i.e., uncontrollable transition b is enabled in the plant G, while $m[p_5] = 0$, i.e., b is not enabled in H. ■

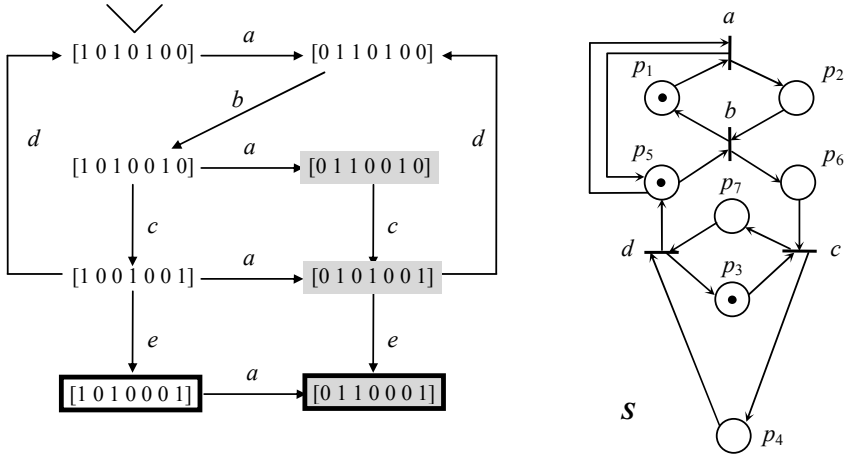


Fig. 12.6 Left: Reachability graph of generator J of Example 12.7. Right: the structure of the trim generator **S** of Example 12.8

12.4.3 Trimming

Once the coarse structure of a candidate supervisor is constructed by means of concurrent composition, we need to trim it to obtain a nonblocking and controllable generator.

The next example shows the problems involved in the trimming of a net.

Example 12.8. Let us consider the generator J constructed in Example 12.7.

Refining the PN to avoid reaching the undesirable markings shown in Fig. 12.6 is complex. First, we could certainly remove the transition labeled by e since its

firing always leads to an undesirable state and it is controllable. After removal of this transition, the transition labeled by a will be enabled by the following reachable markings: $\boldsymbol{m}' = [1\ 0\ 1\ 0\ 1\ 0\ 0]^T, \boldsymbol{m}'' = [1\ 0\ 1\ 0\ 0\ 1\ 0]^T, \boldsymbol{m}''' = [1\ 0\ 0\ 1\ 0\ 0\ 1]^T$. We want to block the transition labeled a when the markings $\boldsymbol{m}''$ and $\boldsymbol{m}'''$ are reached. Since

$$m'[p_5] = 1 > m''[p_5] = m'''[p_5] = 0,$$

we can add an arc from p_5 to a and from a to p_5 as in Fig. 12.6. ■

The following algorithm can be given for the trimming of a net.

Algorithm 12.9. *Let t be a transition to be controlled, i.e., a transition leading from an admissible marking to an undesirable marking. Let e be its label.*

1. *Determine the set of admissible reachable markings that enable t, and partition this set into the disjoint subsets $\mathscr{M}_a$ (the markings from which t should be allowed to fire), and $\mathscr{M}_{na}$ (the markings from which t should not be allowed to fire, to avoid reaching an undesirable marking). If $\mathscr{M}_a = \emptyset$ remove t and stop, else continue.*
2. *Determine a construct in the form:*

$$\begin{aligned}\mathscr{U}(\boldsymbol{m}) = {} & [(m[p_1^1] \geq n_1^1) \wedge \ldots \wedge (m[p_{k1}^1] \geq n_{k1}^1)] \vee \\ & \ldots \\ & \vee [(m[p_1^l] \geq n_1^l) \wedge \ldots \wedge (m[p_{kl}^l] \geq n_{kl}^l)],\end{aligned}$$

   such that $\mathscr{U}(\boldsymbol{m}) = TRUE$ if $\boldsymbol{m} \in \mathscr{M}_a$, and $\mathscr{U}(\boldsymbol{m}) = FALSE$ if $\boldsymbol{m} \in \mathscr{M}_{na}$.
3. *Replace transition t with l transitions $t^1, \ldots, t^l$ labeled a. The input (output) arcs of transition t^j, $j = 1, \ldots, l$, will be those of transition t plus n_i^j arcs inputting from (outputting to) place p_i^j, $i = 1, \ldots, k_j$.*

It is clear that following this construction there is an enabled transition labeled e for any marking in $\mathscr{M}_a$, while none of these transitions are enabled by a marking in $\mathscr{M}_{na}$. We also note that in general several constructs of this form may be determined. The one which requires the minimal number of transitions, i.e., the one with the smallest l, is preferable.

The following theorem gives a sufficient condition for the applicability of the algorithm.

Theorem 12.1. *The construct of Algorithm 12.9 can always be determined if the net is bounded.*

Proof. For sake of brevity, we prove this result for the more restricted class of conservative nets. One should keep in mind, however, that given a bounded non conservative net, one can make the net conservative adding dummy sink places that do not modify its behavior.

A net is conservative if there exists an integer vector $Y > \mathbf{0}$ such that for any two markings $\boldsymbol{m}$ and $\boldsymbol{m}'$ reachable from the initial marking $Y^T\boldsymbol{m} = Y^T\boldsymbol{m}'$. Hence if $\boldsymbol{m} \neq \boldsymbol{m}'$ there exists a place p such that $m[p] > m'[p]$. Also the set of reachable markings is finite.

On a conservative net, consider $\boldsymbol{m}_i \in \mathscr{M}_a$, $\boldsymbol{m}_j \in \mathscr{M}_{na}$. We have that $\mathscr{M}_a$ and $\mathscr{M}_{na}$ are finite sets and also there exists a place p_{ij} such that $m_i[p_{ij}] = n_{ij} > m_j[p_{ij}]$. Hence

$$\mathscr{U}(\boldsymbol{m}) = \bigvee_{i \in \mathscr{M}_a} \left[\bigwedge_{j \in \mathscr{M}_{na}} (m[p_{ij}] \geq n_{ij}) \right]$$

is a construct for Algorithm 12.9. □

Unfortunately, the construct may contain up to $|\mathscr{M}_a|$ *OR* clauses, i.e., up to $|\mathscr{M}_a|$ transitions may be substituted for a single transition to control. Note, however, that it is often possible to determine a simpler construct as in Example 12.8, where the construct for the transition labeled a was $\mathscr{U}(\boldsymbol{m}) = [m[p_5] \geq 1]$.

12.5 Supervisory Control of Unbounded PN Generators

As we have seen in the previous section, the monolithic supervisory design presented in Algorithm 12.6 can always be applied when the plant G and the specification H are bounded PN generators. Here we consider the case of general, possibly unbounded, generators.

In step 1 of the monolithic supervisory design algorithm the unboundedness of the G or H does not require any special consideration, since the procedure to construct the concurrent composition $J = G \parallel H$ is purely structural in the PN setting. Thus we need to focus on the last two steps, and discuss how it is possible to check if an unbounded generator G is nonblocking and controllable, and eventually how it can be trimmed.

We have previously remarked that in the case of bounded nets the L-type language can describe any marked language. In the case of unbounded generators other choices for the marked language are possible considering the G-type language of the generator or even any other type of terminal language mentioned in § 12.2.4.

In the rest of this section we will only consider two types of marked languages for a PN generator G.

- *L-type language*, i.e., $L_m(G) = L_L(G)$. This implies that the set of *marked markings* reached by words in $L_m(G)$ is $\mathscr{F} = F$, i.e., it coincides with the finite set of final markings associated to the generator.
- *G-type marked language*, i.e., $L_m(G) = L_G(G)$. This implies that the set of *marked markings* reached by words in $L_m(G)$ is

$$\mathscr{F} = \bigcup_{\boldsymbol{m}_f \in F} \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{m} \geq \boldsymbol{m}_f\},$$

  i.e., it is the infinite covering set of F.

12.5.1 Checking Nonblockingness

We will show in this subsection that checking a generator for nonblockingness is always possible.

Let us first recall the notion of home space.

Definition 12.8. *A marking $\boldsymbol{m} \in \mathbb{N}^m$ of a Petri net is a* home-marking *if it is reachable from all reachable markings.*

A set of markings $\mathscr{M} \subseteq \mathbb{N}^m$ of a Petri net is a home space *if for all reachable marking $\boldsymbol{m}$ a marking in $\mathscr{M}$ is reachable from $\boldsymbol{m}$.*

The following result is due to Johnen and Frutos Escrig.

Proposition 12.1. [8] *The property of being a home space for finite unions of linear sets*[9] *having the same periods is decidable.*

We can finally state the following original result.

Theorem 12.2. *Given a generator J constructed as in step 1 of Algorithm 12.6 it is decidable if it is nonblocking when its marked language is the L-type or G-type language.*

Proof. Let $\mathscr{F}$ be the set of marked markings of the generator. According to Definition 12.6 generator J is nonblocking iff from every reachable markings $\boldsymbol{m}$ a marked marking in $\mathscr{F}$ is reachable. Thus checking for nonblockingness is equivalent to checking if the set of marked markings $\mathscr{F}$ is a home space.

When the marked language is the L-type language, $\mathscr{F} = F$ and we observe that each marking $\boldsymbol{m}_f$ can be considered as a linear set with base $\boldsymbol{m}_f$ and empty set of generators.

When the marked language is the G-type language,

$$\mathscr{F} = \bigcup_{\boldsymbol{m}_f \in F} \{\boldsymbol{m} \in \mathbb{N}^m \mid \boldsymbol{m} \geq \boldsymbol{m}_f\} = \{\boldsymbol{m}_f + \sum_{i=1}^{m} k_i \boldsymbol{e}_i \mid \ k_i \in \mathbb{N}\}$$

where vectors $\boldsymbol{e}_i$ are the canonical basis vectors, i.e., $\boldsymbol{e}_i \in \{0,1\}^n$, with $e_i[i] = 1$ and $e_i[j] = 0$ if $i \neq j$.

In both cases $\mathscr{F}$ is the finite unions of linear sets having the same periods, hence checking if it is a home space is decidable by Proposition 12.1. □

12.5.2 Checking Controllability

We will show in this subsection that checking a generator for controllability is always possible. The material presented in this subsection is original and proofs of all results will be given.

[9] We say that $\mathscr{E} \subseteq \mathbb{N}^m$ is a *linear set* if there exists some $\boldsymbol{v} \in \mathbb{N}^m$ and a finite set $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\} \subseteq \mathbb{N}^m$ such that $\mathscr{E} = \{\boldsymbol{v}' \in \mathbb{N}^m \mid \boldsymbol{v}' = \boldsymbol{v} + \sum_{i=1}^{n} k_i \boldsymbol{v}_i \text{ with } k_i \in \mathbb{N}\}$. The vector $\boldsymbol{v}$ is called the *base* of $\mathscr{E}$, and $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n$ are called its *periods*.

We first present some intermediate result.

Lemma 12.1. *Let $\langle N, \boldsymbol{m}_0 \rangle$ be a marked net with $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ and $|P| = m$. Given a marking $\bar{\boldsymbol{m}} \in \mathbb{N}^m$ and a place $\bar{p} \in P$, we define the set*

$$\mathscr{S}(\bar{\boldsymbol{m}}, \bar{p}) = \{\boldsymbol{m} \in \mathbb{N}^m \mid m[\bar{p}] = \bar{m}[\bar{p}], \ \ (\forall p \in P \setminus \{\bar{p}\})\ m[p] \geq \bar{m}[p]\}$$

of those markings that are equal to $\bar{\boldsymbol{m}}$ in component $\bar{p}$ and greater than or equal to $\bar{\boldsymbol{m}}$ in all other components.

Checking if a marking in this set is reachable in $\langle N, \boldsymbol{m}_0 \rangle$ is decidable.

Proof. To prove this result, we reduce the problem of determining if a marking in $\mathscr{S}(\bar{\boldsymbol{m}}, \bar{p})$ is reachable to the standard marking reachability problem (see Chapter 10) of a modified net.

Consider in fact net $N' = (P', T', \boldsymbol{Pre}', \boldsymbol{Post}')$ obtained from N as follows. $P' = P \cup \{p_s, p_f\}$; $T' = T \cup \{t_f\} \cup \{t_p \mid \forall p \in P \setminus \{\bar{p}\}\}$. For $p \in P$ and $t \in T$ it holds $\boldsymbol{Pre}'[p,t] = \boldsymbol{Pre}[p,t]$ and $\boldsymbol{Post}'[p,t] = \boldsymbol{Post}[p,t]$, while the arcs incident on the newly added places and transitions are described in the following. Place p_s is self-looped with all transitions in T, i.e., $\boldsymbol{Pre}'[p_s,t] = \boldsymbol{Post}'[p_s,t] = 1$ for all $t \in T$. Place p_f is self-looped with all new transitions t_p, for all $p \in P \setminus \{\bar{p}\}$. Transition t_f has an input arc from place p_s and an output arc to place p_f; furthermore it has $\bar{m}[p]$ input arcs from any place $p \in P \setminus \{\bar{p}\}$. Finally, for all $p \in P \setminus \{\bar{p}\}$ transition t_p is a sink transition with a single input arc from place p.

We associate to N' an initial marking $\boldsymbol{m}'_0$ defined as follows: for all $p \in P$, $m'_0[p] = m_0[p]$, while $m'_0[p_s] = 1$ and $m'_0[p_f] = 0$. Such a construction is shown in Fig. 12.7 where the original net N with set of places $P = \{\bar{p}, p', \ldots, p''\}$ and set of transitions $T = \{t_1, ldots, t_n\}$ is shown in a dashed box. Arcs with starting and ending arrows represent self-loops.

We claim that a marking in the set $\mathscr{S}(\bar{\boldsymbol{m}}, \bar{p})$ is reachable in the original net if and only if marking $\boldsymbol{m}'_f$ is reachable in $\langle N', \boldsymbol{m}'_0 \rangle$, where $m'_f[\bar{p}] = \bar{m}[\bar{p}]$, $m'_f[p_f] = 1$ and $m'_f[\bar{p}] = 0$ for $p \in P' \setminus \{\bar{p}, p_f\}$.

This can be proved by the following reasoning. The evolution of net N' before the firing of t_f mimics that of N. Transition t_f may only fire from a marking greater than or equal to $\bar{\boldsymbol{m}}$ in all components but eventually $\bar{p}$. After the firing of t_f, the transitions of the original net are blocked (p_s is empty) and only the sink transitions t_p, for all $p \in P \setminus \{\bar{p}\}$, may fire thus emptying the corresponding places. The only place whose markings cannot change after the firing of t_f is $\bar{p}$. □

Theorem 12.3. *Given a generator $J = G \parallel H$ constructed as in step 1 of Algorithm 12.6 it is decidable if it is controllable.*

Proof. We will show that the set of uncontrollable markings to be checked can be written as the finite union of sets of the form $\mathscr{S}(\bar{\boldsymbol{m}}, \bar{p})$.

Given an uncontrollable transition $t \in T_{uc}$ let $P_G(t)$ (resp., $P_H(t)$) be the set of input places of t that belong to generator G (resp., H). Consider now a place $p \in P_H(t)$ and an integer $k \in \{0, 1, \ldots, Pre[p,t] - 1\}$ and define the following marking $\boldsymbol{m}_{t,p,k}$ such that $m_{t,p,k}[p] = k$, $m_{t,p,k}[p'] = Pre[p',t]$ if $p' \in P_G(t)$, else $m_{t,p,k}[p'] = 0$. Clearly,

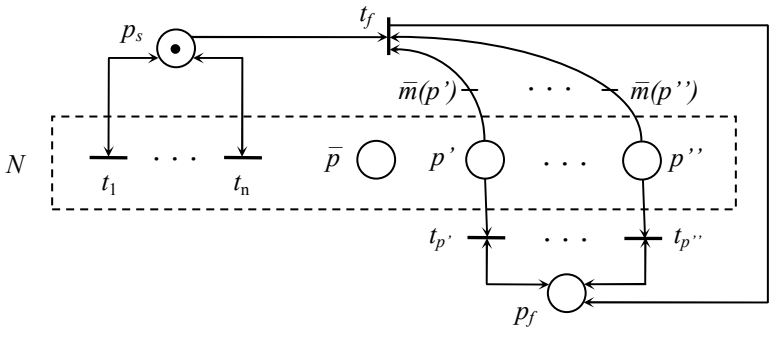


Fig. 12.7 Construction of Lemma 12.1

such a marking is uncontrollable because the places in G contain enough tokens to enable uncontrollable transition t while place p in H does not contain enough tokens to enable it. All other markings in $\mathscr{S}(\boldsymbol{m}_{t,p,k}, p)$ are equally uncontrollable.

Thus the overall set of uncontrollable markings to be checked can be written as the finite union

$$\bigcup_{t\in T_{uc}} \bigcup_{p\in P_H(t)} \bigcup_{k\in\{0,1,\ldots,Pre[p,t]-1\}} \mathscr{S}(\boldsymbol{m}_{t,p,k}, p)$$

and by Lemma 12.1 checking if an uncontrollable marking is reachable is decidable. □

12.5.3 Trimming a Blocking Generator

The problem of trimming a blocking net is the following: given a deterministic PN generator G with languages $L_m(G)$ and $L(G) \supset \overline{L_m(G)}$ one wants to modify the structure of the net to obtain a new DES G' such that $L_m(G') = L_m(G)$ and $L(G') = \overline{L_m(G')} = \overline{L_m(G)}$.

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

On Petri net models the trimming may be more complex. If the Petri net is bounded, it was shown in the previous section how the trimming may be done without major changes of the net structure, in the sense that one has to add new arcs and eventually duplicate transitions without introducing new places. Here we discuss the general case of possibly unbounded nets.

When the marked language of a net is its L-type Petri net language, the trimming of the net is not always possible as will be shown by means of the following example.

Example 12.10. Let G be the deterministic PN generator in Fig. 12.8 (left), with $\boldsymbol{m}_0 = [1\ 0\ 0\ 0]^T$ and set of final markings $F = \{[0\ 0\ 0\ 1]^T\}$. The marked (L-type) and closed behaviors of this net are: $L_m(G) = \{a^m b a^m b \mid m \geq 0\}$ and $L(G) = \{\overline{a^m b a^n b} \mid m \geq n \geq 0\}$. The infinite reachability graph of this net is partially shown in Fig. 12.8 (right): here the unique final marking is shown in a box.

One sees that all markings of the form $[0\ k\ 0\ 1]^T$ with $k \geq 1$ are blocking. To avoid reaching a blocking marking one requires that p_2 be empty before firing the transition inputting into p_4. However, since p_2 is unbounded this cannot be done with a simple place/transition structure. ■

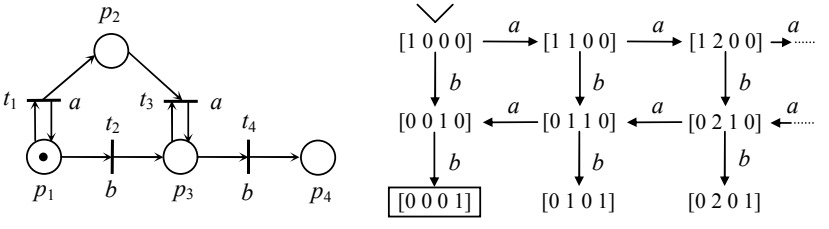


Fig. 12.8 Left: Blocking net of Example 12.10; Right: Its labeled reachability graph

It is possible to prove formally that the prefix closure of the marked language of the net discussed in Example 12.10 is not a P-type Petri net language. The proof is based on the pumping lemma for P-type PN languages, given in [7].

Lemma 12.2. (Pumping lemma). *Consider a PN language $L \in \mathscr{P}$. Then there exist numbers k, l such that any word $w \in L$, with $|\ w\ | \geq k$, has a decomposition $w = xyz$ with $1 \leq |\ y\ | \leq l$ such that $xy^i z \in L, \forall i \geq 1$.*

Proposition 12.2. *Consider the L-type PN language $L' = \{a^m b a^m b \mid m \geq 0\}$. Its prefix closure $L = \overline{L'}$ is not a P-type Petri net language.*

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

When the marked language of a net is its G-type Petri net language, the trimming of the net is always possible because the prefix closure of such a language is a deterministic P-type Petri net language. This follows from the next theorem, that provides an even stronger result.

Theorem 12.4. [4] *Given a deterministic PN generator $G = (N, \ell, \boldsymbol{m}_0, F)$ with $\overline{L_G(G)} \subsetneq L_P(G)$, there exists a finite procedure to construct a new deterministic PN generator G' such that $L_G(G') = L_G(G)$ and $L_P(G') = \overline{L_G(G')}$.*

12.5.4 Trimming an Uncontrollable Generator

In this section we show by means of an example that given a PN generator $J = G \parallel H$ obtained by concurrent composition of a plant and of a specification, it is not always possible to trim it removing the uncontrollable markings.

Example 12.11. Consider a plant G described by the PN generator on the left of Fig. 12.11 (including the dashed transition and arcs). We are interested in the closed language of the net, so we will not specify a set of final markings F: all reachable markings are also final. We assume $T_{uc} = \{t_1, t_3, t_5\}$, i.e., $E_{uc} = \{a\}$.

Consider a specification H described by the PN generator on the left of Fig. 12.9 (excluding the dashed transition and arcs).

On the right of Fig. 12.9 we have represented the labeled reachability graph of G (including the dashed arcs labeled a on the bottom of the graph) and the labeled reachability graph of H (excluding the dashed arcs labeled a on the bottom of the graph). Now if we consider the concurrent composition $J = G \parallel H$ and construct its labeled reachability graph, we obtain a graph isomorphic to the labeled graph of generator H (only the labeling of the nodes changes).

All markings of the form $[0\ k\ 0\ 1]^T$ with $k \geq 1$ are uncontrollable: in fact, when the plant is in such a marking the uncontrollable transition t_5 labeled a is enabled, while no event labeled a is enabled on J. If we remove all uncontrollable markings, we have a generator whose closed language is $L = \{\overline{a^m b a^m b} \mid m \geq 0\}$ that, however, as shown in Proposition 12.2, is not a P-type language. ■

Based on these results in [3] the following result was proven.

Theorem 12.5. The classes $\mathscr{P}_d$, $\mathscr{G}_d$, and $\mathscr{L}_d$ of PN languages are not closed under the supremal controllable sublanguage operator[10].

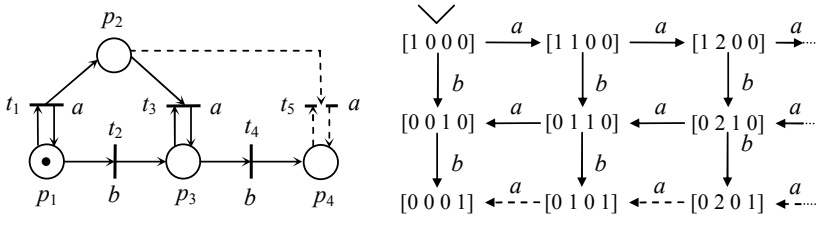


Fig. 12.9 Left: Generators of Example 12.11. Right: Their labeled reachability graphs

[10] See Chapter 3 for a formal definition of this operator.

12.5.5 Final Remarks

The results we have presented in this section showed that in the case of unbounded PN generators a supervisor may not always be represented as a PN. In fact, while it is always possible to check a given specification for nonblockingness and controllability — even in the case of generators with an infinite state space — when these properties are not satisfied the trim behavior of the closed loop system may not be represented as a net. A characterization of those supervisory control problems that admit PN supervisors is an area still open to future research.

12.6 Further Reading

The book by Peterson [12] contains a good introduction to PN languages, while other relevant results can be found in [1, 7, 10, 11, 16, 18].

Many issues related to PNs as discrete event models for supervisory control have been discussed in the survey by Holloway *et al.* [5] and in the works of Giua and DiCesare [3, 4]. The existence of supervisory control policies that enforce liveness have been discussed by Sreenivas in [15, 17].

Finally, an interesting topic that has received much attention in recent years is the supervisory control of PNs under a special class of state specifications called *Generalized Mutual Exclusion Constraints* (GMECs) that can be enforced by controllers called *monitor places* [2]. Several monitor-based techniques have been developed for the control of Petri nets with uncontrollable and unobservable transitions and good surveys can be found in [6, 9].

References

1. Gaubert, S., Giua, A.: Petri net languages and infinite subsets of $\mathbb{N}^m$. Journal of Computer and System Sciences 59, 373–391 (1999)
2. Giua, A., DiCesare, F., Silva, M.: Generalized Mutual Exclusion Constraints for Nets with Uncontrollable Transitions. In: Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, Chicago, USA, pp. 974–799 (1992)
3. Giua, A., DiCesare, F.: Blocking and controllability of Petri nets in supervisory control. IEEE Transactions on Automatic Control 39(4), 818–823 (1994)
4. Giua, A., DiCesare, F.: Decidability and closure properties of weak Petri net languages in supervisory control. IEEE Transactions on Automatic Control 40(5), 906–910 (1995)
5. Holloway, L.E., Krogh, B.H., Giua, A.: A Survey of Petri Net Methods for Controlled Discrete Event Systems. Discrete Event Dynamic Systems 7, 151–190 (1997)
6. Iordache, M.V., Antsaklis, P.J.: Supervision Based on Place Invariants: A Survey. Discrete Event Dynamic Systems 16, 451–492 (2006)
7. Jantzen, M.: Language Theory of Petri Nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 254, pp. 397–412. Springer, Heidelberg (1987)

8. Johnen, C., Frutos Escrig, D.: Decidability of home space property. In: LRI 503. Univ. d'Orsay (1989)
9. Moody, J.O., Antsaklis, P.J.: Supervisory Control of Discrete Event Systems Using Petri Nets. Kluwer (1998)
10. Parigot, M., Pelz, E.: A Logical Formalism for the Study of Finite Behaviour of Petri Nets. In: Rozenberg, G. (ed.) APN 1985. LNCS, vol. 222, pp. 346–361. Springer, Heidelberg (1986)
11. Pelz, E.: Closure Properties of Deterministic Petri Net Languages. In: Brandenburg, F.J., Wirsing, M., Vidal-Naquet, G. (eds.) STACS 1987. LNCS, vol. 247, pp. 373–382. Springer, Heidelberg (1987)
12. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, Englewood Cliffs (1981)
13. Ramadge, P.J., Wonham, W.M.: The control of discrete event systems. Proceedings of the IEEE 77(1), 81–98 (1989)
14. Reutenauer, C.: The Mathematics of Petri Nets. Masson and Prentice-Hall (1990)
15. Sreenivas, R.S.: On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modeled by controlled Petri nets. IEEE Transactions on Automatic Control 42(7), 928–945 (1997)
16. Sreenivas, R.S.: On minimal representations of Petri net languages. IEEE Transactions on Automatic Control 51(5), 799–804 (2006)
17. Sreenivas, R.S.: On the Existence of Supervisory Policies That Enforce Liveness in Partially Controlled Free-Choice Petri Nets. IEEE Transactions on Automatic Control 57(2), 435–449 (2012)
18. Vidal-Naquet, G.: Deterministic Petri net languages. In: Girault, C., Reisig, W. (eds.) Application and Theory of Petri Net. Informatick-Fachberichte, vol. 52. Springer, New York (1982)