1

From Petri Nets to Stochastic Petri Nets

1.1. Introduction

Petri nets (PNs) (Murata 1989) represent a well-known and versatile paradigm for the modeling and verification of various discrete-event systems (Silva and Teruel 1996). Mainly, PNs have been used to model systems in which some events can occur concurrently with some constraints on the concurrence, precedence or frequency of their occurrences (Peterson 1977). In fact, their graphical nature allows intuitive modeling of parallelism, synchronization, conflicts, concurrency, etc., in complex systems, while their formal semantics and mathematics foundation allow unambiguous descriptions, as well as formal verification.

PNs can be analyzed through either computing all reachable states or using methods in discrete mathematics such as matrix algebra. PN properties are used to detect deadlocks, overflow, irreversible situations, etc. By using extended versions of PNs such as stochastic PN and timed PNs, the designers become able to do a performance evaluation of their systems.

PNs enjoy numerous advantages that can be summarized in the following (Peterson 1981; Kamath and Viswanadham 1986; Girault and Valk 2001):

1) They have a powerful mathematical foundation, as well as an intuitive graphical representation. The graphical representation offers a flat view to PN models, making it possible to have simple and very explicit models. As well, their graphic modeling enables easy visualization of complex systems. Usually, in many similar techniques, only either a graphical or mathematical side is well developed.

2) They provide well-integrated abstraction and refinement mechanisms that enable an effective design of large-scale and complex systems.

3) They have been used in a wide range of application areas. Hence, there is a high degree of expertise in this modeling field.

4) There are many extensions of Petri nets such as colored, timed, stochastic, high-level, object-oriented Petri nets, etc., that fit well with specific requirements of a wide range of applications areas.

5) Their mathematical foundation is at the origin of all the analysis techniques proposed in order to verify the modeled systems. Indeed, they dispose of a well-developed qualitative/quantitative analysis panoply.

6) The state space can be given in a compact representation of the state. It is not required to explicitly enumerate all possible reachable states; instead, only the state evolution is provided.

7) Finally, they offer a careful balance of modeling power and decision power. In fact, Petri nets have been used in the modeling of a wide variety of systems. As for their decision power, the reachability problem is decidable in Petri nets (note that most problems can be converted into reachability problems).

Nevertheless, PNs have some disadvantages (Peterson 1981):

– **State-space explosion**: as systems become increasingly complex, their state spaces increase more and more, which can lead to state-space explosion problem.

– **A delicate modeling and verification coupling**: subclasses of PNs increase the decision power; however, a large number of systems can no longer be modeled. On the other hand, extensions of PNs may increase the modeling power, but at the expense of properties decidability.

The remainder of this chapter starts by introducing some intuitive aspects of Petri nets in section 1.2. Then, their formal aspects, such that structure, behavior, qualitative properties and analysis methods are presented in sections 1.3–1.5. After providing Petri net basics, a major class of Petri nets that considers time and quantitative properties, called stochastic Petri nets (SPN), is described in sections 1.6 and 1.7. Finally, this chapter ends with a conclusion.

1.2. Modeling with Petri nets

Discrete event systems are defined as systems of which their states are described by discrete variables and their state evolution depends on the occurrence of discrete events (Cassandras and Lafortune 2009). In discrete event systems' modeling process, two important concepts are considered, namely, *events* and *conditions*, and the relationships among them.

Conditions are descriptions of system states in terms of values of some variables, while events are actions taking place in the system. The occurrence of these events

is controlled by system states, i.e. conditions. As such, at a given time, the condition may hold which implies the occurrence of certain events. The conditions causing such occurrence are called *pre-conditions*. As well, the occurrence of these events may give new conditions. Such new conditions are called *post-conditions*.

This view can be intuitively modeled by Petri nets (Peterson 1981). In fact, conditions are modeled by **places**, depicted as circles, considered as variables. The values of these variables are given in terms of **tokens**, depicted as black dots, inside their corresponding places, which models the truth values of conditions. Events are modeled by **transitions**, depicted as bars. The pre-conditions of an event are the input places, connected by **direct arcs**, of the corresponding transition. Analogously, the post-conditions are the output places. The occurrence of an event is modeled by **firing** the corresponding transition. The state evolution after an event occurrence (i.e. firing a transition) is modeled by (i) **consuming** tokens from pre-conditions (i.e. places) of the corresponding transition and (ii) **producing** new tokens in its post-conditions.

Consider PN H shown in Figure 1.1. It models two processes $\mathtt{p}_1$ and $\mathtt{p}_2$ trying to access a critical resource. Initial marking (i.e. state) M_0 is modeled by one token in place $\mathtt{idle}_1$, one token in place $\mathtt{idle}_2$, one token in place res and zero token in the other places. This initial state, called initial marking, is denoted by $M_0(\mathtt{CS}_1, \mathtt{CS}_2, \mathtt{idle}_1, \mathtt{idle}_2, \mathtt{res}, \mathtt{wait}_1, \mathtt{wait}_2) = (0, 0, 1, 1, 1, 0, 0)$.

A token in places $\mathtt{idle}_i$, $\mathtt{wait}_i$ and $\mathtt{CS}_i$ means that process $\mathtt{p}_i$ is idle, waiting to access a critical resource and in a critical section, respectively. Transitions $\mathtt{request}_i$, $\mathtt{enter}_i$ and $\mathtt{free}_i$ are fired when process $\mathtt{p}_i$ requests a critical resource, enters a critical section and frees a critical resource, respectively.

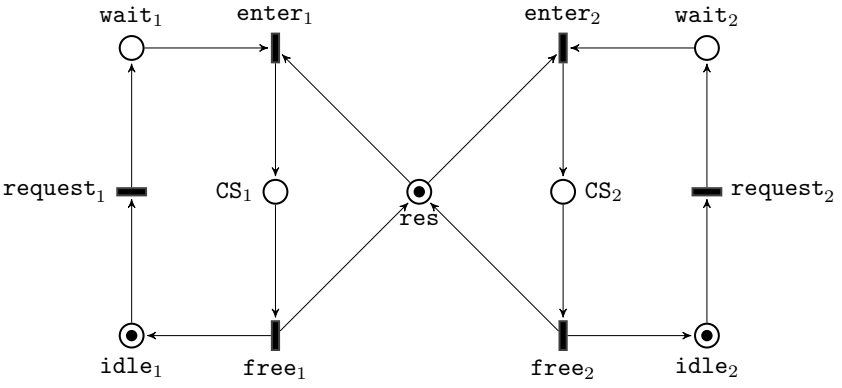


Figure 1.1. *A PN models mutual exclusion*

1.3. Petri net structure

A Petri net is a bipartite directed graph that consists of places, depicted as circles, transitions, depicted as bars, and arcs connecting either place to transition or transition to place.

DEFINITION 1.1.– ***Petri net***. *A Petri net is a 4-tuple* $\mathcal{N} = \langle P, T, F, M_0 \rangle$ *where*

– P *is a finite and non-empty set of places,*

– T *is a finite and non-empty set of transitions disjoint from* P*,*

– $F : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ *is a flow relation for a set of arcs,*

– $M_0 : P \longrightarrow \mathbb{N}$ *is an initial marking.*

Some authors refer to the unmarked net $\mathcal{N} = \langle P, T, F \rangle$ as the Petri net structure and refer to the marked net $\mathcal{N}' = \langle P, T, F, M_0 \rangle$ as the system. Furthermore, if the net marking is partially provided, then the Petri net is called a family (Chiola et al. 1991).

EXAMPLE 1.1.– *Consider PN* H *shown in Figure 1.1. Its formal definition is given by* $H = \langle P, T, F, M_0 \rangle$ *where*

– $P = \{\mathtt{idle}_1, \mathtt{wait}_1, \mathtt{CS}_1, \mathtt{idle}_2, \mathtt{wait}_2, \mathtt{CS}_2, \mathtt{res}\}$,

– $T = \{\mathtt{request}_1, \mathtt{enter}_1, \mathtt{free}_1, \mathtt{request}_2, \mathtt{enter}_2, \mathtt{free}_2\}$,

– $F(\mathtt{idle}_1, \mathtt{request}_1) = 1, F(\mathtt{request}_1, \mathtt{wait}_1) = 1$*, etc.*

– $M_0(\mathtt{CS}_1, \mathtt{CS}_2, \mathtt{idle}_1, \mathtt{idle}_2, \mathtt{res}, \mathtt{wait}_1, \mathtt{wait}_2) = (0, 0, 1, 1, 1, 0, 0)$.

DEFINITION 1.2.– ***Preset and postset***. *The inputs and outputs, also called preset and postset, respectively, of places and transitions can be defined formally as follows:*

– *The preset of place* p*, denoted by* $\bullet p$*, is given by* $\bullet p = \{t \in T | F(t, p) > 0\}$.

– *The postset of place* p*, denoted by* $p\bullet$*, is given by* $p\bullet = \{t \in T | F(p, t) > 0\}$.

– *The preset of transition* t*, denoted by* $\bullet t$*, is given by* $\bullet t = \{p \in P | F(p, t) > 0\}$.

– *The postset of transition* t*, denoted by* $t\bullet$*, is given by* $t\bullet = \{p \in P | F(t, p) > 0\}$.

EXAMPLE 1.2.– *Considering PN* H *shown in Figure 1.1, we have, for instance,*

1) $\bullet\mathtt{idle}_1 = \{\mathtt{free}_1\}$ *and* $\mathtt{idle}_1\bullet = \{\mathtt{request}_1\}$.

2) $\bullet\mathtt{enter}_1 = \{\mathtt{wait}_1, \mathtt{res}\}$ *and* $\mathtt{enter}_1\bullet = \{\mathtt{CS}_1\}$.

1.4. Dynamic behavior of Petri nets

Previously, we have considered the static part of PNs. In the present section, we deal with the dynamic evolution of PN marking controlled by two rules called "enabling rule" and "firing rule". Both enabling and firing rules are specified through arc multiplicities (weights) and place markings. As for arcs, the enabling rule depends on input arcs of a transition, while the firing rule considers both input and output arcs.

Firing a transition in PN models an occurrence of its corresponding event. Before firing any transition, we check if its pre-conditions hold; in such case, the transition is said to be *enabled*.

A transition t is enabled if each place in its preset contains at least as many tokens as the multiplicity of the arc connecting both of them.

DEFINITION 1.3.– ***Enabling rule***. *Transition t is enabled at marking M, denoted by $M[t\rangle$, if $M(p) \geq F(p,t), \forall\, p \in {\bullet}t$.*

EXAMPLE 1.3.– *Consider PN H shown in Figure 1.1. Transition $\mathtt{request}_1$ is enabled since we have ${\bullet}\mathtt{request}_1 = \{\mathtt{idle}_1\}$, $M(\mathtt{wait}_1) = 1$ (i.e. place $\mathtt{idle}_1$ contains one token) and $M(\mathtt{wait}_1) \geq F(\mathtt{idle}_1, \mathtt{request}_1)$.*

DEFINITION 1.4.– ***Firing rule***. *The firing of transition t enabled at marking M yields new marking M', denoted by $M[t\rangle M'$, such that $M'(p) = M(p) + F(t,p) - F(p,t),\ \forall\, p \in P$.*

Firing transition t (i) removes from each place in its preset as many tokens as the multiplicity of the arc connecting both of them, and produces to each place in its postset as many tokens as the multiplicity of the arc connecting both of them.

EXAMPLE 1.4.– *Consider PN H shown in Figure 1.1. At initial marking M_0, transitions $\mathtt{request}_1$ and $\mathtt{request}_2$ are fireable. In fact, the direct arc from $\mathtt{idle}_1$ to transition $\mathtt{request}_1$ means that the pre-condition of transition $\mathtt{request}_1$ is place $\mathtt{idle}_1$. This place contains one token, i.e. the pre-conditions of $\mathtt{request}_1$ hold; hence, it can fire (i.e. the corresponding event can occur). Firing $\mathtt{request}_1$ removes a token from $\mathtt{idle}_1$ and puts one token inside $\mathtt{wait}_1$, since we have an arc from $\mathtt{request}_1$ to $\mathtt{wait}_1$, which models the post-conditions of firing $\mathtt{request}_1$. Similarly to transition $\mathtt{request}_2$ with respect to places $\mathtt{idle}_2$ and $\mathtt{wait}_2$.*

PN H after firing $\mathtt{request}_1$ is shown in Figure 1.2a. Firing $\mathtt{request}_1$ at M_0 causes enabling of transition $\mathtt{enter}_1$ at new marking M_1. Firing the latter transition at M_1 leads to new marking illustrated in Figure 1.2b. In Figure 1.2c, transition $\mathtt{enter}_2$ cannot fire, since its pre-conditions are $\mathtt{wait}_2$ and $\mathtt{res}$ such that $\mathtt{res}$ does not contain any token which means that pre-conditions of $\mathtt{enter}_2$ do not hold. After

firing $\texttt{free}_1$ *at marking depicted in Figure 1.2c, the obtained marking is depicted in Figure 1.2d in which* $\texttt{enter}_2$ *becomes enabled.*

DEFINITION 1.5.– ***Transition sequence.*** *A transition sequence* $\sigma = t_1, t_2, \ldots, t_n$ *is fireable starting from marking* M_1 *if there exists a sequence of markings* $M_1, M_2, \ldots, M_n$ *such that* $M_i[t_i\rangle, \forall\, i \in \{1, 2, \ldots, n\}$.

$M_1[\sigma\rangle M_{n+1}$ denotes the firing of transition sequence σ starting from M_1 and M_{n+1} is reachable from M_1 by firing σ.

EXAMPLE 1.5.– *Consider Table 1.1, where the initial marking is denoted by* s_0. *Firing sequence* $\sigma = \texttt{request}_1, \texttt{enter}_1, \texttt{request}_2, \texttt{free}_1, \texttt{enter}_2, \texttt{free}_2$ *is fireable starting from* s_0, *since there exists a marking sequence* $s_0, s_2, s_5, s_7, s_1, s_3$, *such that* $s_0[\texttt{request}_1\rangle s_2[\texttt{enter}_1\rangle\ s_5[\texttt{request}_2\rangle s_7[\texttt{free}_1\rangle s_1[\texttt{enter}_2\rangle s_3[\texttt{free}_2\rangle$.

Since PN structure is static and firing transitions changes only the marking, which is considered as the dynamic part of PNs, the state evolution can be modeled as a graph where its nodes correspond to reachable markings, and its edges correspond to fired transitions. Such a graph is called the reachability graph. Figure 1.3 shows the reachability graph of PN H *depicted in Figure 1.1, and their corresponding values are listed in Table 1.1.*

DEFINITION 1.6.– ***Reachability set.*** *The reachability set of a PN having initial marking* M_0, *denoted by* $RS(M_0)$, *is defined as follows:* $RS(M_0) = \{M | M_0[\sigma\rangle M \ \wedge\ \sigma$ *is a fireable sequence*$\}$.

Note that (i) σ can be empty, i.e. $M_0 \in RS(M_0)$, (ii) $RS(M_0)$ can be infinite and (iii) an initial marking must be completely provided in order to compute the reachability set, i.e. this computation is not possible neither for PN structure nor for PN family (Chiola et al. 1991; Marsan et al. 1994).

EXAMPLE 1.6.– *Consider PN* H *depicted in Figure 1.1. Its reachability set is shown in Table 1.1, i.e.* $RS(M_0) = \{s_0, s_1, \ldots, s_7\}$.

DEFINITION 1.7.– ***Reachability graph.*** *The reachability graph of a PN* $\mathcal{N} = \langle P, T, F, M_0\rangle$, *denoted by* $RG(M_0) = \langle V, E\rangle$, *is a labeled directed graph, where*

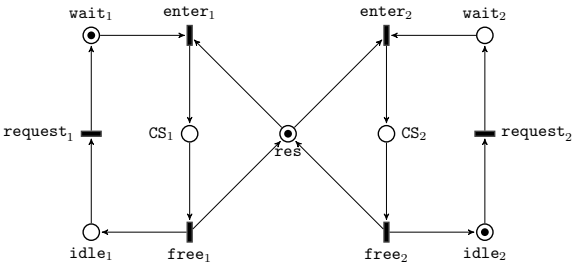
i) $V = RS(M_0)$, *and*

ii) $E = \{(M_i, t, M_j) | M_i \in RS(M_0) \wedge M_j \in RS(M_0) \wedge t \in T\}$.

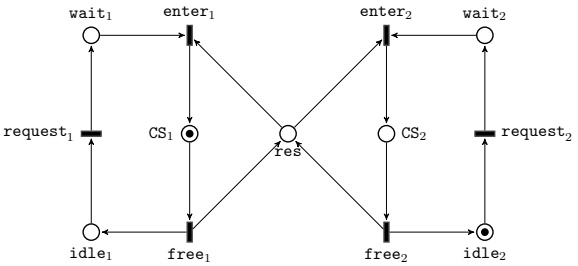
EXAMPLE 1.7.– *Consider PN* H *depicted in Figure 1.1. Its reachability graph is shown in Figure 1.3, i.e.* $RG(M_0) = \langle V, E\rangle$ *where:*

i) $V = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$, *and*

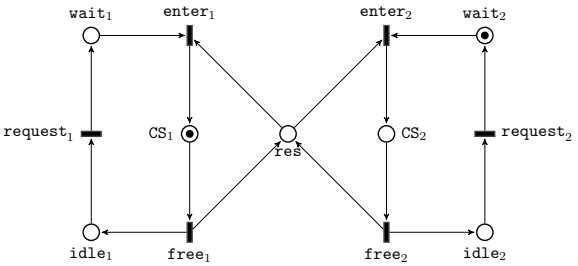
ii) $E = \{(s_0, \texttt{request}_2, s_1), (s_0, \texttt{request}_1, s_2), (s_1, \texttt{enter}_2, s_4), (s_1, \texttt{request}_1, s_3), \ldots\}$.



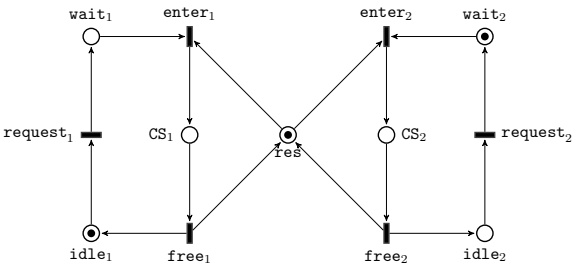
(a) Firing $\texttt{request}_1$



(b) Firing $\texttt{request}_1$ then $\texttt{enter}_1$



(c) Firing $\texttt{request}_1$ $\texttt{enter}_1$ $\texttt{request}_2$



(d) Firing $\texttt{request}_1$ $\texttt{enter}_1$ $\texttt{request}_2$ $\texttt{free}_1$

Figure 1.2. *Firing transitions in PN H at M_0*

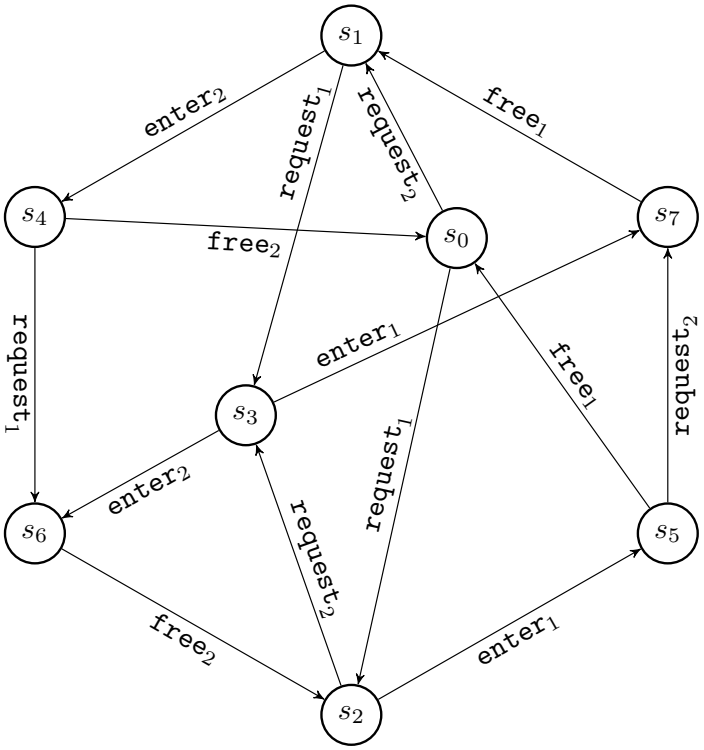


Figure 1.3. *Reachability graph of* H

	$\texttt{CS}_1$	$\texttt{CS}_2$	$\texttt{idle}_1$	$\texttt{idle}_2$	res	$\texttt{wait}_1$	$\texttt{wait}_2$
s_0	0	0	1	1	1	0	0
s_1	0	0	1	0	1	0	1
s_2	0	0	0	1	1	1	0
s_3	0	0	0	0	1	1	1
s_4	0	1	1	0	0	0	0
s_5	1	0	0	1	0	0	0
s_6	0	1	0	0	0	1	0
s_7	1	0	0	0	0	0	1

Table 1.1. *Reachable markings of* H

1.5. Petri net analysis

Besides its modeling power, a major reason for using Petri nets is its decision power. Numerous analysis techniques have been developed for PN verification. Indeed, Petri nets support the analysis of many properties and problems associated with concurrent systems (Murata 1989). The properties can be analyzed either based on a reachability graph, called behavioral properties, or independently from a reachability graph, called structural properties.

In this section, we discuss only important properties and their analysis problems. For further reading about properties analysis, the reader is referred to (Peterson 1981; Murata 1989; Esparza and Nielsen 1994; Marsan et al. 1994; Girault and Valk 2001; Reisig 2012).

1.5.1. *Petri net properties*

Reachability

The reachability problem for Petri nets consists of deciding if a marking M is reachable from the initial marking. It has been proven that this problem is decidable (Kosaraju 1982; Mayr 1984; Reutenauer 1990; Lambert 1992). The reachability problem is a central concept in Petri nets since most problems can be converted into reachability problems.

Liveness and deadlock-freedom

A Petri net is said to be live if every transition can always fire again in the future at any reachable marking. It has been shown that the liveness problem is decidable since it is recursively equivalent to the reachability problem (Araki and Kasami 1976).

This kind of liveness is a strong property. Thus, it is relaxed to different levels of liveness (Peterson 1977; Murata 1989). A transition in a Petri net having initial marking M_0 is

– **L0-live (or dead)** if it can never be enabled at any reachable marking.

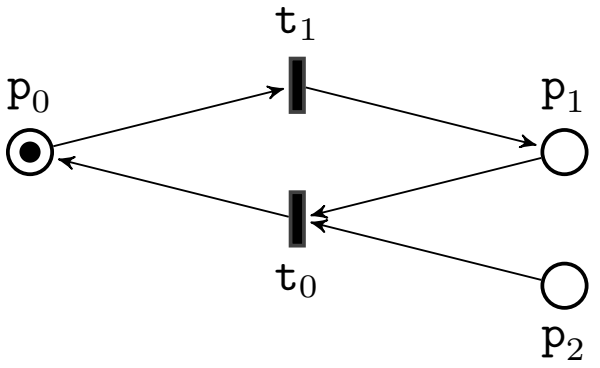
– **L1-live** if it can be enabled at least once in certain firing sequence starting from M_0.

– **L2-live** if it can fire at least k times in certain firing sequence starting from M_0, where k is a positive integer number.

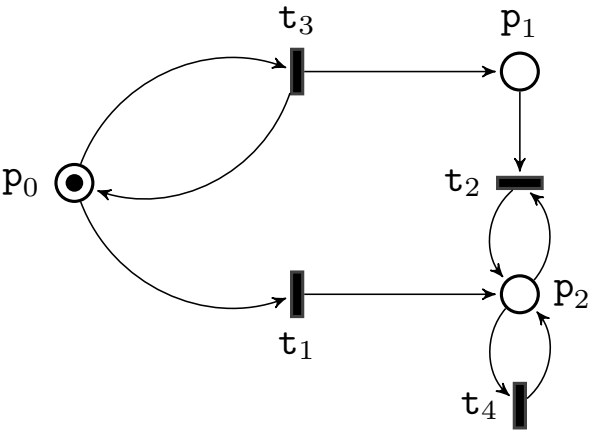
– **L3-live** if it appears infinitely often in certain firing sequence starting from M_0.

– **L4-live (or live)** if it is L1-live for any reachable marking from M_0.

A Petri net is Lk-live if all transitions are Lk-live, such that $k \in 0, 1, 2, 3, 4$.



(a) $\mathtt{t}_0$ is L0-live and $\mathtt{t}_1$ is L1-live



(b) $\mathtt{t}_2$ is L2-live, $\mathtt{t}_3$ is L3-live and $\mathtt{t}_4$ is L4-live

Figure 1.4. *Different levels of liveness*

EXAMPLE 1.8.– *Petri nets in Figure 1.4 show different levels of liveness, where* $\mathtt{t}_0$ *in Figure 1.4a is dead; and transitions* $\mathtt{t}_1$, $\mathtt{t}_2$, $\mathtt{t}_3$ *and* $\mathtt{t}_4$ *in Figure 1.4b are L1-, L2-, L3- and L4-live, respectively.*

Boundedness

A Petri net is bounded if its reachable set is finite. It has been proved that boundedness is decidable (Karp and Miller 1969). In fact, a Petri net is unbounded if there exists a reachable marking M and transition sequence σ such that $M[\sigma\rangle M + L$, where L is a non-zero marking (Esparza and Nielsen 1994).

A Petri net is *k-bounded* if, for any reachable marking, there is at most k tokens in any place. A Petri net is safe if it is 1-bounded.

If a place in a Petri net represents a buffer, it would be interesting to verify whether this place is bounded or safe to guarantee that there will be no overflows in the buffer.

EXAMPLE 1.9.– *PNs shown in Figures 1.4a and 1.4b are safe and non-bounded, respectively.*

Home state and reversibility

A marking of a Petri net is a home state if it can be reached from all reachable markings. De Frutos-Escrig (1986) and Best and Esparza (2016) have shown that a home state problem is decidable. A Petri net is said to be reversible if its initial marking is a home state.

EXAMPLE 1.10.– *Marking $M(p_0, p_1, p_2) = (0, 0, 1)$ is a home state for the Petri net shown in Figure 1.4b, such that it can be reached by firing t_1 once and firing t_2 until it becomes no longer enabled. The Petri net in Figure 1.1 is reversible as shown in its reachability graph depicted in Figure 1.3.*

Conservation

A Petri net is said to be conservative if the number of tokens in any reachable marking remains constant, i.e. after any transition firing the number of consumed tokens equals the number of produced tokens. This property would be useful to show that the resources (modeled by tokens) are neither created nor destroyed during the state evolution of the system.

EXAMPLE 1.11.– *Reachable markings of the Petri net depicted in Figure 1.5a are $(1, 1, 0, 0)$, $(0, 0, 1, 1)$, $(1, 0, 0, 1)$ and $(0, 1, 1, 0)$ (the marking of p_0, p_1, p_2 and p_3).*

Persistence

A Petri net is persistent if any two different transitions t_1 and t_2 are enabled at any reachable marking M, then firing of t_1 will not disable t_2, and vice versa. Grabowski (1980), Mayr (1981) and Müller (1981) have proved that this problem is decidable.

EXAMPLE 1.12.– *Consider the Petri net shown in Figure 1.5a. At initial marking, only transition t_0 is enabled. Firing this transition leads to a new marking $M_1(p_0, p_1, p_2, p_3) = (0, 0, 1, 1)$ at which both transitions t_1 and t_2 are enabled. Firing t_1 at M_1 will not disable t_2, and vice versa.*

Fairness

There are two types of fairness: bounded-fairness and unconditional fairness. Two transitions are in a bounded-fair relation if the occurrence number of one is bounded, while the other does not yet fire. A PN is bounded-fair if any two transitions are in a bounded-fair relation.

A firing sequence is unconditionally fair if it is either finite or all transitions appear infinitely often in it. A PN is unconditionally fair if all fireable sequences from its initial marking are unconditionally fair.

EXAMPLE 1.13.– *Consider the Petri net shown in Figure 1.5a. All firing sequences starting from the initial marking are words generated from the following regular expression* $(\mathtt{t}_0(\mathtt{t}_1\,\mathtt{t}_2|\mathtt{t}_2\,\mathtt{t}_1))^*$*, i.e. are concatenations of* $\mathtt{t}_0\ \mathtt{t}_1\ \mathtt{t}_2$ *and/or* $\mathtt{t}_0\ \mathtt{t}_2\ \mathtt{t}_1$*. Thus, this Petri net is bounded-fair.*

The Petri net depicted in Figure 1.5b is unfair. In fact, both transitions $\mathtt{t}_0$ and $\mathtt{t}_2$ can fire, sequentially, infinitely without any firing of neither $\mathtt{t}_1$ nor $\mathtt{t}_3$. As for the PN illustrated in Figure 1.5c, it is unconditionally fair but not bounded-fair since place p_3 is an unbounded place making the occurrence number of $\mathtt{t}_3$, without firing other transition, unbounded as well.

Mutual exclusion

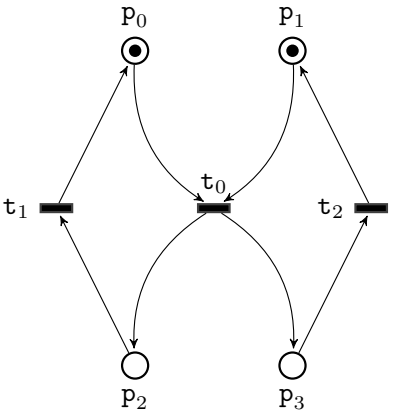
There are two types of mutual exclusion dealing with the impossibility of simultaneous sub-markings or firing concurrency. Two places p and q are mutually exclusive if both of them cannot be marked at the same marking, i.e. places p and q are mutually exclusive if $\forall M \in RS(M_0), M(p) \times M(q) = 0$. Two transitions are mutually exclusive if they cannot be both enabled at any reachable marking.

EXAMPLE 1.14.– *Consider the Petri net shown in Figure 1.5b. Its reachable markings are* $(1,0,0)$*,* $(0,1,0)$ *and* $(0,0,1)$ *(the values in each vector are the markings of places* $\mathtt{p}_0$*,* $\mathtt{p}_1$ *and* $\mathtt{p}_2$*, respectively). Hence, each of the two places in this Petri net is mutually exclusive.*

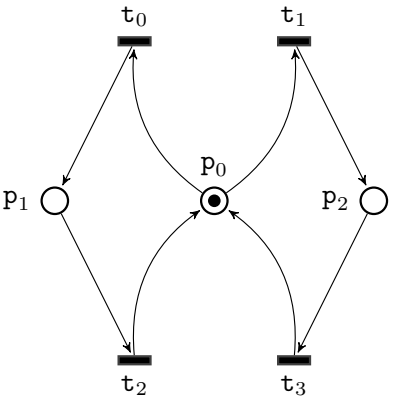
Furthermore, at any reachable marking listed above, there exists one and only one fireable transition. Thus, each of the two transitions in this Petri net is mutually exclusive.

1.5.2. *Temporal logic*

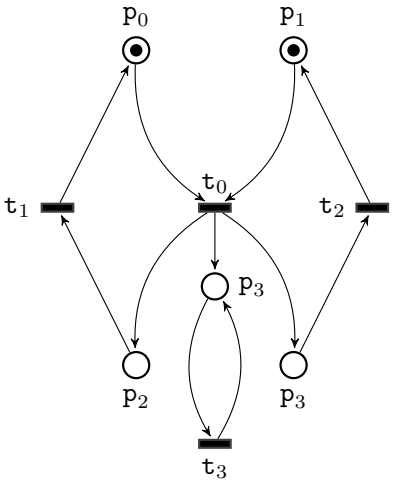
To consider a large set of properties in Petri nets, researchers have studied decidability issues using *model checking* (Baier and Katoen 2008).



(a) Conservative, persistent and bounded-fair Petri net



(b) Unfair Petri net



(c) Unconditionally fair Petri net

Figure 1.5. *Conservation, persistence and fairness*

Model checking refers to a wide range of techniques that check whether a given property formalized by a *temporal logic* is verified by a given system modeled, often, by a transition system. The model checker confirms that either the system satisfies the property or violates it. In the latter case, a counter-example is provided.

Among the properties that can be checked by model checking, there are two important types of properties, namely, the liveness and safety properties (Girault and

Valk 2001). Roughly speaking, a liveness property (eventually) must be satisfied in the future by all future executions, i.e. "good things" do happen. For instance, if a process asks for a critical resource, it will have access to it in the future. A safety property (always) must be satisfied in each reachable state in the future, i.e. "bad things" do not happen. For instance, a critical resource will never be used by two processes or more at the same time. It has been shown in Alpern and Schneider (1987) that any property can be decomposed as a conjunction of safety and liveness properties.

Aforementioned, usually a model checker requires formalizing properties as formulae of temporal logic and describing systems as transition systems. In the following, we briefly present transition systems and temporal logic.

DEFINITION 1.8.– ***Transition system.*** *A transition system TS is a tuple $\langle S, T, E, I, \mathcal{A}, \mathcal{L}\rangle$, where:*

– *S is a set of states and T is a set of actions,*

– *$E \subseteq S \times T \times S$ is a transition relation on S,*

– *$I \subseteq S$ is a set of initial states,*

– *$\mathcal{A}$ is a set of atomic propositions,*

– *$\mathcal{L} : S \longrightarrow 2^{\mathcal{A}}$ is a labeling function that assigns truth value to atomic propositions at each state, i.e. $a \in \mathcal{A}$ is true at $s \in S$ if $a \in \mathcal{L}(s)$.*

In PN context, S, T and E are obtained from the reachability graph, I contains a single initial marking, and the atomic propositions are conditions on place markings.

EXAMPLE 1.15.– *Consider transition system TS depicted in Figure 1.6 of Petri net H shown in Figure1.1. TS is defined as follows: $TS = \langle S, T, E, I, \mathcal{A}, \mathcal{L}\rangle$, where:*

– *S is the set of reachable states of H, i.e. $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,*

– *$T = \{\mathtt{request}_1, \mathtt{request}_2, \mathtt{enter}_1, \mathtt{enter}_2, \mathtt{free}_1, \mathtt{free}_2\}$,*

– *$E = \{(s_0, \mathtt{request}_2, s_1), (s_0, \mathtt{request}_1, s_2), (s_1, \mathtt{enter}_2, s_4), (s_1, \mathtt{request}_1, s_3), \ldots\}$,*

– *$I = \{s_0\}$,*

– *$\mathcal{A} = \{(\mathtt{idle}_1 = 1), (\mathtt{idle}_2 = 1), (\mathtt{wait}_1 = 1), (\mathtt{wait}_2 = 1), (\mathtt{CS}_1 = 1), (\mathtt{CS}_2 = 1)\}$, such that $(\mathtt{idle}_1 = 1)$ means $\mathtt{idle}_1$ is marked by one token. Similarly to the other places,*

– *$\mathcal{L}$ is the labeling function such that $\mathcal{L}(s_0) = \{(\mathtt{idle}_1 = 1), (\mathtt{idle}_2 = 1)\}$, $\mathcal{L}(s_1) = \{(\mathtt{idle}_1 = 1), (\mathtt{wait}_2 = 1)\}$, $\mathcal{L}(s_2) = \{(\mathtt{wait}_1 = 1), (\mathtt{idle}_2 = 1)\}$, etc.*

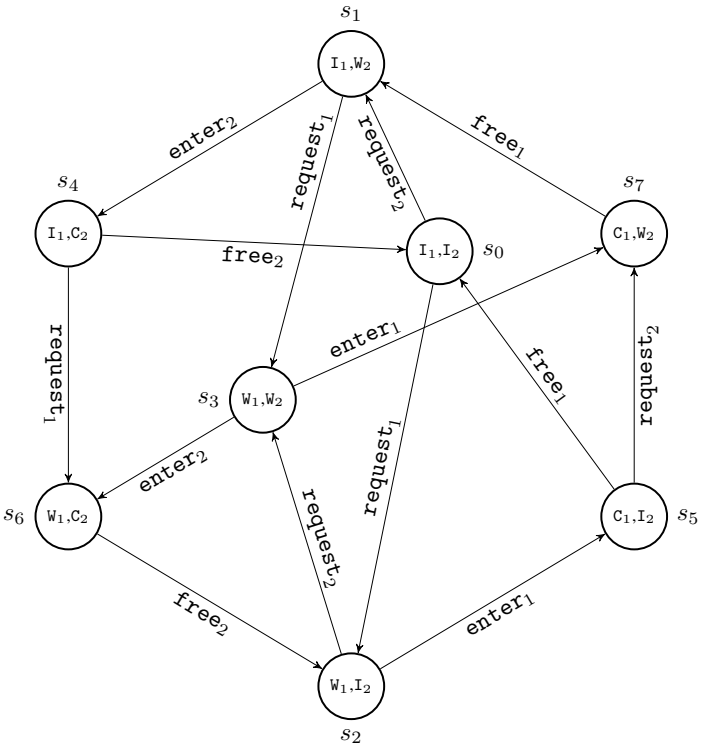


Figure 1.6. *Transition system of H shown in Figure 1.1, where I_i, W_i and C_i denote ($idle_i = 1$), ($wait_i = 1$) and ($CS_i = 1$) for each $i \in \{1, 2\}$, respectively*

DEFINITION 1.9.– ***Finite path, maximal path and path***. *A finite path is a state sequence $s_0 s_1 \ldots s_n$, where $\exists\ (s_{i-1}, t, s_i) \in E$, such that $t \in T$, for all $i \in \{1, 2, \ldots, n\}$. A maximal path π of a transition system with no terminal state is an infinite sequence of states $s_0 s_1 s_2 \ldots$, such that $\forall\ i, \exists\ t \in T, (s_i, t, s_{i+1}) \in E$. A maximal path $\pi = s_0\ s_1\ s_2 \ldots$ is a path, if $s_0 \in I$.*

EXAMPLE 1.16.– *Consider transition system TS depicted in Figure 1.6. $s_0 s_2 s_5 s_7$ is a finite path, $s_2 s_5 s_7 s_1 s_4 s_0\ s_1 \ldots$ is a maximal path, and $s_0 s_2 s_5 s_7 s_1 s_4 s_0 \ldots$ is a path.*

DEFINITION 1.10.– ***Trace***. *The trace of a maximal path $\pi = s_0 s_1 s_2 \ldots$ is defined as $trace(\pi) = \mathcal{L}(s_0)\mathcal{L}(s_1)\mathcal{L}(s_2) \ldots$. That is, the trace registers the valid propositions in each state of π.*

EXAMPLE 1.17.– *The trace of $s_0 s_2 s_5 s_7 s_1 s_4 s_0 \ldots$ is $\{I_1, I_2\}\{W_1, I_2\}\{C_1, I_2\}$ $\{C_1, W_2\}\{I_1, W_2\}\{I_1, C_2\}\ \{I_1, I_2\} \ldots$.*

As for temporal logics, there are two categories, namely, linear time logic (LTL) and branching time logic (CTL). In this book, we will restrict our attention to the linear temporal logic. A comprehensive formal description of model checking can be found in Baier and Katoen (2008).

DEFINITION 1.11.– ***LTL formula.*** *An LTL formula is inductively defined as follows:*

$$\Phi ::= \mathit{true} \mid a \mid \neg\Phi \mid \Phi_1 \vee \Phi_2 \mid \boldsymbol{X}\Phi \mid \Phi_1 \, \boldsymbol{U} \, \Phi_2$$

where $a \in \mathcal{A}$ is an atomic proposition.

EXAMPLE 1.18.– *All the following formulae are LTL formulae*

– $\Phi_1 ::= (\mathcal{CS}_1 = 1)$,

– $\Phi_2 ::= \neg(\mathcal{CS}_1 = 1) \vee \neg(\mathcal{CS}_1 = 1)$,

– $\Phi_3 ::= \boldsymbol{X}(\neg(\mathcal{CS}_1 = 1) \vee \neg(\mathcal{CS}_1 = 1))$,

– $\Phi_4 ::= (\mathtt{wait}_2 = 1)\boldsymbol{U}(\mathcal{CS}_2 = 1)$.

Informally, formula $\mathbf{X}\Phi$ holds for a maximal path π if its second state satisfies Φ. For instance, consider path $\pi_0 = s_0 s_2 s_5 s_7 s_1 s_4 s_0 \ldots$. Formula Φ_3 holds for π_0 since $(\neg(\mathtt{CS}_1 = 1) \vee \neg(\mathtt{CS}_1 = 1))$ is satisfied in s_2.

As for formula $\Phi_1 \mathbf{U} \Phi_2$, it holds for a maximal path π if there exists a state s that satisfies Φ_2, such that each prior state thereof satisfies Φ_1. For instance, consider the maximal path $\pi_1 = s_7 s_1 s_4 s_0 \ldots$. Formula Φ_4 holds for π_1, since formula $(\mathtt{wait}_2 = 1)$ holds for s_7 and s_1; and formula $(\mathtt{CS}_2 = 1)$ is satisfied in s_4. As for the maximal path $\pi_2 = s_7 s_1 s_3 s_7 s_1 s_3 s_7 \ldots$, formula Φ_4 does not hold.

Operators *eventually* (denoted by **F**) and *always* (denoted by **G**) can be expressed via operator **U**, such that ($\mathbf{F}\Phi \equiv \mathbf{true}\mathbf{U}\Phi$) and ($\mathbf{G}\Phi \equiv \neg\mathbf{F}\neg\Phi$).

1.5.3. *Analysis methods*

In the literature, Petri net analysis methods can be categorized as *enumeration* or *net-driven*.

The first category consists of the computation of the reachability graph (totally or partially). If the reachability graph is finite, it can be used for a proof system or for decision procedures, where the most important properties are decidable. On the other hand, we can use the *coverability tree* for unbounded systems; however, important properties such as liveness and reachability are not decidable (Peterson 1981). Basically, enumeration-based methods are applicable to any Petri net class;

however, due to computational complexity, their use is limited to small nets (Murata 1989).

To overcome the state-space explosion problem, net-driven approaches reason on the net structure to extract useful information about the net behavior.

Two different approaches are structure theory and net transformations. The former is based on graph theory and/or linear algebra. The latter transforms a Petri net to another one (typically by reduction) while preserving the properties to be analyzed. The resulting models are expected to be easier to analyze (Girault and Valk 2001).

Analysis by transformation transforms a net system $\mathcal{S}$ into a net system $\mathcal{S}'$, preserving a set of properties Π in such a way that verifying set Π in $\mathcal{S}'$ becomes easier than in $\mathcal{S}$ such that $\mathcal{S}'$ may belong to a Petri net subclass for which state enumeration can be avoided.

A special class of transformation methods is reduction methods in which a net system $\mathcal{S}$ is transformed into a net system $\mathcal{S}'$ (eventually using a reduction sequence) in such a way that $\mathcal{S}'$ is smaller than $\mathcal{S}$, preserving a set of properties Π of $\mathcal{S}$.

In structural analysis techniques, the net behavior is reasoned from its structure, and its initial marking is considered as a parameter. We can find two subcategories of structural analysis:

1) *Linear algebra*: using matrix equations, they allow fast verification without the necessity of enumeration in certain cases.

2) *Graph-based techniques*: they are effective in analyzing some subclasses of Petri nets.

Although net-driven techniques are powerful, in many cases, they are applicable only to special subclasses of Petri nets (Murata 1989). For further information about these analysis techniques, the reader is referred to Peterson (1981), Murata (1989) and Girault and Valk (2001).

1.6. Stochastic Petri nets

In this section, we present a class of Petri net enriched by temporal concepts, namely SPNs. Indeed, Petri nets, in their basic definition, do not involve any concept of time quantification. The notion of time is abstract in basic Petri nets, i.e. only logical relationships in terms of time such as transition t_1 fires *before* transition t_2, transition t_3 fires *after* transition t_4, etc., are allowed. Thus, performance evaluation of systems using basic Petri nets is not possible.

On the other hand, the time concept is viral in certain real applications whose efficiency is a crucial aspect. Hence, time-augmented Petri nets are introduced. In general, there are two possible ways to do this (Bause and Kritzinger 2002):

– **Timed places Petri nets (TPPNs)**: in this class of Petri nets, tokens fired onto a place p are unavailable to all its output transitions for a certain time. Once this time has elapsed, the tokens become available.

– **Timed transitions Petri nets (TTPNs)**: in this class of Petri nets, once a transition becomes enabled, it does not fire immediately; instead, it fires after a certain time.

Time-augmented Petri nets are also classified depending upon time nature. In fact, if time is deterministic, then such Petri nets are called "timed Petri nets". On the other hand, if the firing time is a random variable following a certain distribution law, then they are called "stochastic Petri nets". Different families of SPN can be defined based on their distribution laws of the firing time.

Moreover, the nature of SPN depends also on other firing characteristics (Michel 2001), namely:

Selection policy: given a set of enabled transitions at a certain marking, there are two policies concerned with how tokens are reserved until the transition firing:

1) *Preselection policy*. First, an enabled transition reserves all tokens it needs so that these tokens are not available for other transitions. Then, it waits for its firing time to elapse. Finally, it fires immediately according to the firing rule of Petri nets.

2) *Race policy*. First, an enabled transition waits until its firing time interval has elapsed. Then, it fires immediately according to the firing rule of Petri nets provided it is still enabled at that time, i.e. the required tokens are not already consumed by another transition.

Service policy: given an enabled transition at a certain marking at which it can fire k times, there are three policies concerned with how many times the transition should be fired when its firing time has elapsed:

1) *Single-server*. One single firing is performed. It means that transition can only offer one service at time.

2) *Infinite-server*. k firings are performed. It means that transition can ensure any number of simultaneous services.

3) *Multiple-server*. $Min(k, deg(t))$ firings are performed. It means that transition t can ensure $deg(t)$ simultaneous services at maximum.

Memory policy: assume that each timed transition is associated with a timer. When a timed transition becomes enabled, its corresponding time is set to an initial value and then starts to decrease. When the timer reaches the value zero, the corresponding transition fires. Given a set of enabled transitions at a certain marking, there are three policies concerned with how firing a transition should influence the timers of the other transitions:

1) *Resampling*. At each firing, the timers of all the timed transitions are discarded (restart mechanism).

2) *Enabling memory*. At each firing, the timers of all the disabled timed transitions are restarted. As for timed transitions that are still enabled, their corresponding timers hold their present value (continue mechanism).

3) *Age memory*. At each firing, the timers of all the timed transitions hold their present values (continue mechanism).

In the following, we start by providing a brief introduction of stochastic processes. Then, the basic concepts of Markov chains (MC) are presented. Finally, we consider two major classes of SPN, namely, SPN having exponential law, generalized SPN and their quantitative properties.

1.6.1. *Stochastic process*

Stochastic processes are used to model the evolution of systems that exhibit probabilistic behaviors. They are defined by enumerating possible reachable states by the modeled systems and the probabilities upon time of transitions between these states.

Mathematically, a stochastic process is a family of random variables $\mathcal{X}(t)$ defined over the same probability space taking values in a set S, where parameter t denotes time and used to index each random variable and S is the state space of the process (Bause and Kritzinger 2002). Stochastic processes can be classified according to the state space or the nature of time.

If the state space is discrete, then the process is called *discrete-state process* or chain, whereas if the state space is continuous, then it is called *continuous-space process*. Analogically, if the time is continuous, then the process is called *continuous-time process*; otherwise, it is *called discrete-time process*.

1.6.2. *Markov process*

A particular class of stochastic processes, called Markov process, has found a wide range of applications in the research community. A Markov process is a stochastic

process whose evolution depends only on the present, without being concerned with its evolution history. This is known as *Markov property*, as well as "memoryless".

Let $P[\mathcal{X}(t) = x]$ denote the probability that the stochastic process will have value x at time t and $P[\mathcal{X}(t) = x|\mathcal{X}(t_n) = x_n]$ denote the probability that the stochastic process will have value x at time t such that it had value x_n at time t_n. Formally, a Markov process is a stochastic process $\{\mathcal{X}(t)\}$ whose conditional probability density function is such that

$$\begin{aligned} P[\mathcal{X}(t) = x|\mathcal{X}(t_n) = x_n, \mathcal{X}(t_{n-1}) = x_{n-1}, \ldots, \mathcal{X}(t_0) = x_0] \\ = P[\mathcal{X}(t) = x|\mathcal{X}(t_n) = x_n]. \end{aligned} \quad [1.1]$$

where $t > t_n > t_{n-1} > \cdots > t_0$.

If the state space of a Markov process is discrete, then it is called a *Markov chain* (MC). If the time is continuous (discrete), then is called continuous time MC (CTMC) (respectively, discrete time MC (DTMC)).

Furthermore, if the future evolution of a Markov process is independent of a particular instant t_n in equation [1.1] so that it is completely determined by knowing the present state, then the Markov process is said to be *time-homogeneous*. That is, the Markov process is invariant to shifts in time. For a time-homogeneous Markov process, the following condition holds:

$$P[\mathcal{X}(t+s) = x|\mathcal{X}(t_n+s) = x_n] = P[\mathcal{X}(t) = x|\mathcal{X}(t_n) = x_n]. \quad [1.2]$$

In CTMCs (either time-homogeneous or not), the probabilities of transitions between states are given as functions of time. The amount of time for which the process stays in a state before moving to another one is called the *sojourn time* or *holding time* which is exponentially distributed. Indeed, the exponential distribution is the only continuous distribution that satisfies the property of memoryless (Bause and Kritzinger 2002).

For the remainder of this book, we consider only discrete-state space, time-homogeneous Markov processes.

1.6.3. *Stochastic Petri nets having exponential law*

An SPN having an exponential law $\mathcal{N}$ is a Petri net augmented by a function Λ that assigns to each transition t_i in $\mathcal{N}$ a firing delay λ_i which is exponentially distributed. The distribution of a random variable $\mathcal{X}_i$ of the firing delay of transition t_i is given by

$$F_{\mathcal{X}_i}(x) = 1 - e^{-\lambda_i x}. \quad [1.3]$$

In this book, we refer to timed transition Petri nets with race, single server and resampling mode; and exponentially distributed stochastic firing time by simply SPNs.

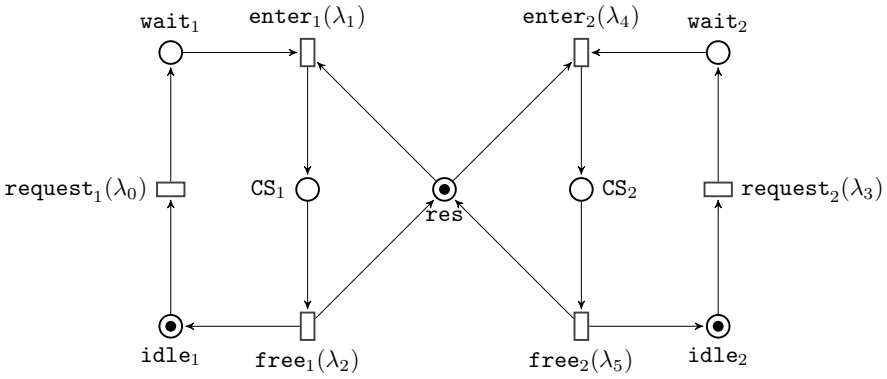


Figure 1.7. *SPN $\mathcal{N}$ models mutual exclusion*

DEFINITION 1.12.– ***Stochastic Petri net****. An SPN is a 5-tuple $\mathcal{N} = \langle P, T, F, M_0, \Lambda \rangle$ where:*

– P is a finite and non-empty set of places,

– T is a finite and non-empty set of transitions disjoint from P,

– $F : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ is a flow relation for a set of arcs,

– $M_0 : P \longrightarrow \mathbb{N}$ is an initial marking,

– $\Lambda : T \longrightarrow \mathbb{R}^+$ is a function that associates with each transition $t_i \in T$ an exponentially distributed firing time λ_i.

EXAMPLE 1.19.– *Consider SPN $\mathcal{N}$ shown in Figure 1.7. Both transitions $\texttt{request}_1$ and $\texttt{request}_2$ are enabled at the initial marking $M_0(\texttt{CS}_1, \texttt{CS}_2, \texttt{idle}_1, \texttt{idle}_2, \texttt{res}, \texttt{wait}_1, \texttt{wait}_2) = (0, 0, 1, 1, 1, 0, 0)$. Before firing, $\texttt{request}_1$ waits until a certain time has elapsed. This time is exponentially distributed with rate λ_0, i.e. the average time for $\texttt{request}_1$ to fire is $\frac{1}{\lambda_0}$. Similarly to transition $\texttt{request}_2$ with respect to rate λ_3.*

The nature of SPNs does not modify the basic behavior of the underlying non-timed model. This is useful since it is possible to study this class of SPNs using the basic model properties as well as the available theoretical results (Marsan et al. 1994). Indeed, the enabling and firing rules of basic Petri nets are preserved in SPNs. That is, once $\texttt{request}_1$ has fired, $\mathcal{N}$ marking becomes $M_2(\texttt{CS}_1, \texttt{CS}_2, \texttt{idle}_1, \texttt{idle}_2, \texttt{res}, \texttt{wait}_1, \texttt{wait}_2) = (0, 0, 0, 1, 1, 1, 0)$. As well, firing $\texttt{request}_2$ at M_0 leads to marking $M_2(\texttt{CS}_1, \texttt{CS}_2, \texttt{idle}_1, \texttt{idle}_2, \texttt{res}, \texttt{wait}_1, \texttt{wait}_2) = (0, 0, 1, 0, 1, 0, 1)$.

Considering probabilistic aspects of SPNs, the next marking depends on which transition fires first. Given n enabled transitions $t_1, t_2, \ldots, t_n$ having rates $\lambda_1, \lambda_2, \ldots, \lambda_n$ at marking M, the probability that t_i fires first is given by

$$P[t_i \text{ fires first at } M] = \frac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_n}. \quad [1.4]$$

Using equation [1.4], the probability that $\texttt{request}_1$ fires first at M_0 equals $\frac{\lambda_0}{\lambda_0+\lambda_3}$ provided that only transitions $\texttt{request}_1$ and $\texttt{request}_2$ are enabled at M_0.

Given the following:

1) the next marking of an SPN depends only on its current marking,

2) the state space of SPNs is discrete,

3) the probabilities of transitions between markings are given as a function exponentially distributed time, and

4) the probability of changing from a marking M to some other marking is independent of the sojourn time spent in M.

Then, an SPN describes a time-homogeneous CTMC that is isomorphic to its reachability graph. Thus, the quantitative analysis of SPNs can be carried out by analyzing the corresponding CTMC. The latter is obtained straightforwardly by computing the reachability graph of a given SPN based on the same algorithm used for the underlying PN where the transition rate between two markings (states) is the rate of the corresponding fired transition in SPN.

EXAMPLE 1.20.– *Consider CTMC illustrated in Figure 1.8.*

This CTMC can be obtained by computing the reachability graph of SPN $\mathcal{N}$ depicted in Figure 1.7 where the label of any transition between two markings is the rate of the corresponding fired transition. For instance, marking s_1 is obtained from s_0 by firing transition $\texttt{request}_2$; thus, the rate of going from s_0 to s_1 is the rate of transition $\texttt{request}_2$, i.e. λ_3.

The next step in the quantitative analysis of SPNs is to compute a *matrix of transition rates* Q=(q_{ij}) of order $n = |RS(SPN)|$, i.e. its order is the number of reachable markings. The element q_{ij} is the transition rate between corresponding markings M_i and M_j, i.e. the rate fired transition t_k such that $M_i[t_k\rangle M_j$. The element q_{ii} is computed such that $\sum_n^{j=1} q_{ij} = 0$, i.e. $q_{ii} = -\sum_n^{j=1} q_{ij,i\neq j}$.

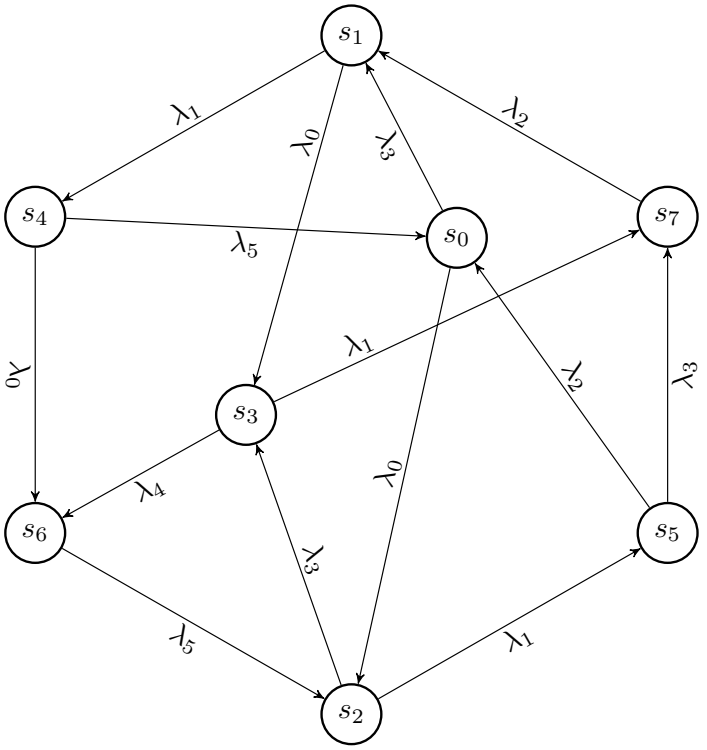


Figure 1.8. *Corresponding CTMC of SPN $\mathcal{N}$ shown in Figure 1.7, where states $s_0, \ldots, s_7$ are listed in Table 1.1*

EXAMPLE 1.21.– *Consider CTMC depicted in Figure 1.8. Its matrix of transition rates Q is given as follows:*

$$Q = \begin{bmatrix} -(\lambda_3+\lambda_0) & \lambda_3 & \lambda_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -(\lambda_0+\lambda_1) & 0 & \lambda_0 & \lambda_1 & 0 & 0 & 0 \\ 0 & 0 & -(\lambda_3+\lambda_1) & \lambda_3 & 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & 0 & -(\lambda_4+\lambda_1) & 0 & 0\lambda_4 & \lambda_1 & \\ \lambda_5 & 0 & 0 & 0 & -(\lambda_5+\lambda_0) & 0 & \lambda_0 & 0 \\ \lambda_2 & 0 & 0 & 0 & 0 & -(\lambda_2+\lambda_3) & 0 & \lambda_3 \\ 0 & 0 & \lambda_5 & 0 & 0 & 0 & -\lambda_5 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & -\lambda_2 \end{bmatrix}.$$

Given a matrix of transition rates, the following step is computing the steady-state distribution $\Pi = [\pi_1, \pi_2, \ldots, \pi_n]$, i.e. the probability of being in reachable markings $(s_1, s_2, \ldots, s_n)$, respectively. To this end, we resolve the following equation system:

$$\Pi \times Q = 0, \textit{ such that } \sum_{n}^{i=1}(\pi_i) = 1. \quad [1.5]$$

EXAMPLE 1.22.– *Consider CTMC depicted in Figure 1.8. Its steady-state distribution* $\Pi = [\pi_0, \ldots, \pi_7]$ *is given as follows:*

$$[\pi_0, \pi_1, \ldots, \pi_7] \times \begin{bmatrix} q_{00} & \cdots & q_{07} \\ \vdots & \ddots & \vdots \\ q_{70} & \cdots & q_{77} \end{bmatrix} = [0,0,0,0,0,0,0,0], \textit{ such that } \sum_{7}^{i=0}(\pi_i) = 1.$$

1.6.4. *Quantitative properties*

After computing the steady-state distribution $\Pi = [\pi_1, \pi_2, \ldots, \pi_n]$, we can then evaluate the following quantitative properties (Marsan et al. 1994; Bause and Kritzinger 2002).

Probability of being in a subset of markings: let $B \subseteq RS(SPN)$ be some markings of interest. Then, the probability of being in a state of the corresponding subset is given by

$$P[B] = \sum_{s_i \in B} \pi_i. \quad [1.6]$$

Mean number of tokens: let $B(p,n)$ denote the subset of $RS(\mathcal{N})$ for which the number of tokens in place p is n, and m_p denotes the maximum number of tokens that can present on place p. Then, the average number of tokens in place p is given by

$$\bar{m} = \sum_{m_p}^{n=1}(nP[B(p,n)]). \quad [1.7]$$

Probability of firing transition t_j: let EN_t denote the subset of $RS(\mathcal{N})$ in which a given transition t is enabled. Then, the probability r that an observer, who looks randomly into the SPN, sees transition t firing next is given by

$$r = \sum_{s_i \in EN_t} \pi_i \left(\frac{\lambda_t}{-q_{ii}} \right) = \sum_{s_i \in EN_t} \pi_i P[t \text{ fires first at } s_i]. \quad [1.8]$$

Throughput at a transition t: the throughput at a transition t is given by

$$d = \sum_{s_i \in EN_t} \pi_i \lambda_t. \quad [1.9]$$

1.7. Generalized stochastic Petri nets

Aforementioned, the firing delay of any transition in SPNs is stochastic and exponentially distributed to model time-consuming activities. Nevertheless, not all activities in real systems are so. For instance, the verification of a logical condition can be proceeded in no time. Hence, the second type of transition, called immediate, is introduced into SPNs, in order to separate the two aspects in the modeling paradigm. Moreover, the introduction of immediate transitions in SPNs does not increase, significantly, the analysis complexity (Marsan et al. 1994). Stochastic Petri nets in which transitions are either immediate or timed are called *generalized SPNs* (GSPNs).

In GSPNs, immediate transitions (depicted as filled bars) fire with priority over timed transitions (depicted as unfilled bars), in zero time, as soon as they become enabled. In fact, enabling an immediate transition disables all timed transitions, i.e. timed transitions in a GSPN are disabled if there exists an enabled immediate transition. Considering firing rule, GSPNs follow the same firing rule of basic Petri nets.

Consider GSPN $\mathcal{G}$ with marking M depicted in Figure 1.9. Transitions $\texttt{enter}_1$ and $\texttt{enter}_2$ are immediate transitions, and the other transitions are timed. In fact, transition $\texttt{enter}_1$ models the verification of a logical condition (if process $\texttt{p}_1$ is waiting for critical resource and the critical resource is free). This action is not a time-consuming activity; therefore, it is natural to model it by an immediate transition that fires in zero time. Similarly to transition $\texttt{enter}_2$ with respect to process $\texttt{p}_2$. At marking M, only transition $\texttt{enter}_1$ is enabled (which disables all timed transitions including $\texttt{request}_2$). Firing $\texttt{enter}_1$ consumes one token from place res and one token from place $\texttt{wait}_1$, and produces one token into place $\texttt{CS}_1$. After firing $\texttt{enter}_1$, transition $\texttt{request}_2$ becomes enabled.

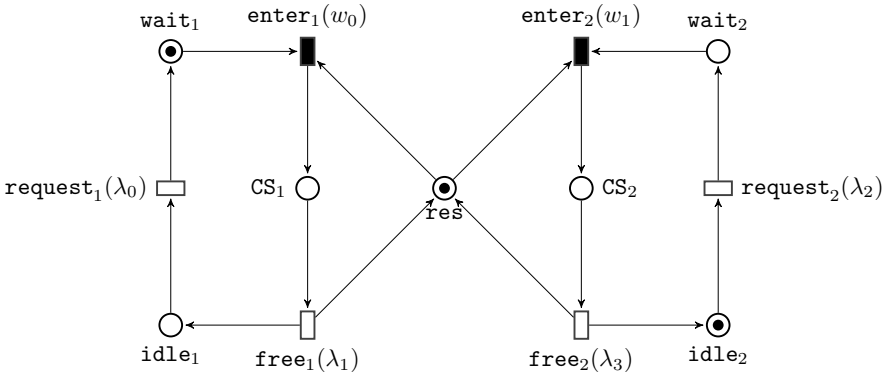


Figure 1.9. *GSPN $\mathcal{G}$ models mutual exclusion*

GSPNs represent an extension of SPNs, where transitions are divided into two sets: a set T_1 of timed transitions and a set T_2 of immediate transitions.

DEFINITION 1.13.– ***Generalized stochastic Petri net***. *A generalized SPN is 7-tuple* $\mathcal{G} = \langle P, T, F, M_0, T_1, T_2, \Lambda \rangle$ *where:*

1) P *is a finite and non-empty set of places,*

2) T *is a finite and non-empty set of transitions disjoint from* P*,*

3) $F : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ *is a flow relation for a set of arcs,*

4) $M_0 : P \longrightarrow \mathbb{N}$ *is an initial marking,*

5) $T_1 \subseteq T$ *is a set of timed transitions,*

6) $T_2 \subset T$ *is a set of immediate transitions, such that* $T_1 \neq \emptyset, T_1 \cap T_2 = \emptyset, T_1 \cup T_2 = T$*,*

7) $\Lambda : T \rightarrow \mathbb{R}^+$*, where* $\Lambda(t_i)$ *is a firing rate/weight of* t_i*.*

In GSPNs, firing an immediate transition t at a marking M_v yielding a marking M'_v is performed in zero time; hence, an observer that looks to the net marking will not see M_v since the latter will be *vanished* instantaneously.

In other words, when the net reaches marking M_v in any instant d, transition t fires immediately and the new net marking becomes M'_v in same instant d; thus, the sojourn time in marking M_v is null. Such markings are called *vanishing markings*. A vanishing marking is a marking that enables an immediate transition.

On the other hand, a *tangible marking* is a marking that either enables a timed transition or is a dead-end marking. The stochastic process sojourns in such markings are exponentially distributed. Therefore, these markings are not left immediately.

Another concept raises from introducing immediate transition which is *timeless trap*. A GSPN is said to have a timeless trap if it can reach a marking that enables an infinite fireable sequence of immediate transitions. Indeed, when a GSPN is in a timeless trap, it means that no timed transition can be fired in the future.

Consider the stochastic process illustrated in Figure 1.10, where its states are listed in Table 1.2. Markings s_1 and s_2 are vanishing markings. Indeed, both of them enable immediate transitions. Since s_1 and s_2 are the only vanishing markings and the only markings that describe states in which a process is waiting to access a free critical resource, then the probability that a random observer sees such situation is null.

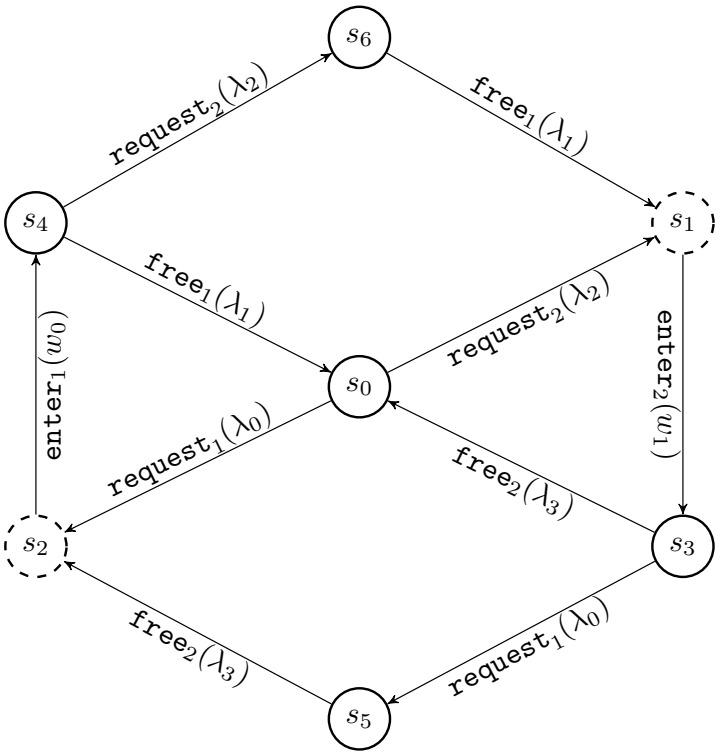


Figure 1.10. *Corresponding stochastic process of GSPN $\mathcal{G}$ shown in Figure 1.9.*

	$\texttt{CS}_1$	$\texttt{CS}_2$	$\texttt{idle}_1$	$\texttt{idle}_2$	$\texttt{res}$	$\texttt{wait}_1$	$\texttt{wait}_2$
s_0	0	0	1	1	1	0	0
s_1	0	0	1	0	1	0	1
s_2	0	0	0	1	1	1	0
s_3	0	1	1	0	0	0	0
s_4	1	0	0	1	0	0	0
s_5	0	1	0	0	0	1	0
s_6	1	0	0	0	0	0	1

Table 1.2. *Reachable markings of $\mathcal{G}$*

1.7.1. *Embedded Markov chain*

GSPNs do not describe a continuous-time Markov process. Indeed, sojourn time in vanishing marking is null and in tangible markings is exponentially distributed; hence, we do not use sojourn time to analyze GSPNs; instead, we consider the probability

of going from a marking M (either vanishing or tangible) to marking M'. These probabilities are given by

$$P[M \longrightarrow M'] = \frac{\sum_{t_i \in \{M[t_i\rangle M'\}} \Lambda(t_i)}{\sum_{t_j \in EN(M)} \Lambda(t_j)}. \quad [1.10]$$

where $EN(M)$ denotes the set of enabled transitions at marking M.

These probabilities are independent of the sojourn time in marking M. Therefore, we analyze the embedded MC of the corresponding stochastic process of GSPNs.

Let $\hat{T}$ denote the tangible state set, $\hat{V}$ vanishing state set, $m = |\hat{V}|$, and $n = |\hat{T}|$. Using embedded MC, the *transition probability matrix* is given as follows (Bause and Kritzinger 2002):

$$P = \left[\begin{array}{ccc|ccc} c_{11} & \cdots & c_{1m} & d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mm} & d_{m1} & \cdots & d_{mn} \\ \hline e_{11} & \cdots & e_{1m} & f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ e_{n1} & \cdots & e_{nm} & f_{n1} & \cdots & f_{nn} \end{array}\right]. \quad [1.11]$$

where

$c_{ij} = P[M_i \longrightarrow M_j | Mi \in \hat{V} \wedge M_j \in \hat{V}]$, $d_{ij} = P[M_i \longrightarrow M_j | Mi \in \hat{V} \wedge M_j \in \hat{T}]$,
$e_{ij} = P[M_i \longrightarrow M_j | Mi \in \hat{T} \wedge M_j \in \hat{V}]$, $f_{ij} = P[M_i \longrightarrow M_j | Mi \in \hat{T} \wedge M_j \in \hat{T}]$.

That is, the upper-left side describes the transition probabilities between vanishing states, the upper-right side describes transition probabilities from vanishing states to tangible ones, the bottom-left side describes transition probabilities from tangible states to vanishing ones and, finally, the bottom-right side describes the transition probabilities between tangible states.

EXAMPLE 1.23.– *Assume mean firing rates* $\lambda_0 = 6, w_0 = 300, \lambda_1 = 3, \lambda_2 = 2$, $w_1 = 300$ *and* $\lambda_3 = 1$. *Solving the equation system, we obtain the following steady-state distribution:*

$$\begin{array}{rc} & \begin{array}{ccccccc} s_1 & s_2 & s_0 & s_3 & s_4 & s_5 & s_6 \end{array} \\ \textit{Vanishing states} \left\{ \begin{array}{c} s_1 \\ s_2 \end{array} \right. & \left[\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \\ \textit{Tangible states} \left\{ \begin{array}{c} s_0 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \end{array} \right. & \left[\begin{array}{ccccccc} \frac{2}{8} & \frac{6}{8} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{7} & 0 & 0 & \frac{6}{7} & 0 \\ 0 & 0 & \frac{3}{5} & 0 & 0 & 0 & \frac{2}{5} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}.$$

The steady-state distribution $\tilde{\Pi} = [\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_{m+n}]$ – of being in states (either vanishing or tangible) $s_1, s_2, \ldots, s_{m+n}$, respectively – of the embedded MC is given by

$$\tilde{\Pi} \times P = \tilde{\Pi}, \text{ such that } \sum_{m+n}^{i=1} (\tilde{\pi}_i) = 1. \quad [1.12]$$

From this steady-state distribution, we can compute the steady-state distribution of the original stochastic process as follows:

$$\pi_i = \begin{cases} \dfrac{\tilde{\pi}_i \times \left(\sum\limits_{t_j \in ET(M_i)} w_j \right)^{-1}}{\sum\limits_{M_k \in \hat{T}} \tilde{\pi}_k \times \left(\sum\limits_{t_j \in ET(M_k)} w_j \right)^{-1}} & \text{if } M_i \in \hat{T} \\ 0 & \text{otherwise.} \end{cases} \quad [1.13]$$

Given the steady-state distribution of tangible states, we can compute quantitative properties presented in section 1.6.4. Note that throughput and firing probability concern only timed transitions.

EXAMPLE 1.24.– *Consider GSPN $\mathcal{G}$ illustrated in Figure 1.9.*

The steady-state distribution $\tilde{\Pi} = [\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_0, \tilde{\pi}_3, \tilde{\pi}_4, \tilde{\pi}_5, \tilde{\pi}_6]$ of its embedded MC is calculated by resolving the following equation system:

$$[\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_0, \tilde{\pi}_3, \tilde{\pi}_4, \tilde{\pi}_5, \tilde{\pi}_6] \times \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{2}{8} & \frac{6}{8} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{7} & 0 & 0 & \frac{6}{7} & 0 \\ 0 & 0 & \frac{3}{5} & 0 & 0 & 0 & \frac{2}{5} \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = [\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_0, \tilde{\pi}_3, \tilde{\pi}_4, \tilde{\pi}_5, \tilde{\pi}_6].$$

such that $\sum_6^{i=0} (\tilde{\pi}_i) = 1$.

That is,

$$[\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_0, \tilde{\pi}_3, \tilde{\pi}_4, \tilde{\pi}_5, \tilde{\pi}_6] \times \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ \frac{2}{8} & \frac{6}{8} & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{1}{7} & 0 & 0 & \frac{6}{7} & 0 & 1 \\ 0 & 0 & \frac{3}{5} & 0 & 0 & 0 & \frac{2}{5} & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = [\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_0, \tilde{\pi}_3, \tilde{\pi}_4, \tilde{\pi}_5, \tilde{\pi}_6, 1].$$

Solving the equation system, we obtain the following steady-state distribution of the embedded MC:

$$\begin{aligned}
\tilde{\pi}_1 &= P[(0,0,1,0,1,0,1)] = 0.1450,\\
\tilde{\pi}_2 &= P[(0,0,0,1,1,1,0)] = 0.1884,\\
\tilde{\pi}_0 &= P[(0,0,1,1,1,0,0)] = 0.0960,\\
\tilde{\pi}_3 &= P[(0,1,1,0,0,0,0)] = 0.1450,\\
\tilde{\pi}_4 &= P[(1,0,0,1,0,0,0)] = 0.1884,\\
\tilde{\pi}_5 &= P[(0,1,0,0,0,1,0)] = 0.1242,\\
\tilde{\pi}_6 &= P[(1,0,0,0,0,0,1)] = 0.1130.
\end{aligned}$$

Using equation [1.13], we obtain the following steady-state distribution $[\pi_0, \pi_1, \pi_2, \pi_3, \pi_4]$, probabilities of being in tangible states s_0, s_3, s_4, s_5 and s_6, respectively, of the original stochastic process:

$$\begin{aligned}
\pi_0 &= P[(0,0,1,1,1,0,0)] = 0.0427,\\
\pi_1 &= P[(0,1,1,0,0,0,0)] = 0.0830,\\
\pi_2 &= P[(1,0,0,1,0,0,0)] = 0.1507,\\
\pi_3 &= P[(0,1,0,0,0,1,0)] = 0.4975,\\
\pi_4 &= P[(1,0,0,0,0,0,1)] = 0.2261.
\end{aligned}$$

1.8. Conclusion

In this chapter, we have presented Petri nets that constitute the basic form of other classes of Petri nets. Their intuitive aspects of modeling have been discussed. As well, the formal modeling and verification based on Petri net are presented.

SPNs that provide a common extension of Petri nets combining qualitative and quantitative verification are introduced. Finally, several of their underlying concepts such as MC, steady-state distribution, etc., are illustrated via certain examples.

In spite of their modeling and decision power, Petri nets face several shortcomings in the design and analysis of *dynamic-structure systems*. Hence, it is required to introduce dynamic structures to Petri nets in order to model/verify such systems in a natural way. In the next chapter, we are interested in the extension of Petri nets onto dynamic-structure formalisms.