A Comparison of the Expressiveness of SPA and Bounded SPN models

J. Hillston[1], L. Recalde[2], M. Ribaudo[3], M. Silva[2] *

[1] LFCS, University of Edinburgh

[2] DIIS, Universidad de Zaragoza

[3] Dipartimento di Informatica, Universitá di Torino

Abstract

This paper presents some transformation techniques from bounded SPN systems to corresponding SPA models, preserving concurrency. Initially, a simple algorithm is introduced, showing that the obtained SPA models simulate the net systems. Then the algorithm is improved in order to keep more of the net structure, when possible. For the case of non ordinary net systems the SPA language is extended by introducing a new cooperation operator.

1 Introduction and motivation

Stochastic extensions of Petri nets [1, 2, 11] and, more recently, of process algebras [10, 5, 9] have proved to be useful tools for performance modelling. The work presented in this paper is motivated by a desire to complete the establishment of a sound and efficient bridge between these modelling paradigms. In addition to its inherent analytical interest, such a bridge opens the possibility that some of the powerful theoretical results which have been developed for stochastic Petri nets (SPN) may be exploited in stochastic process algebras (SPA), and vice versa.

The transformation from SPA to bounded SPN, has previously been published [3, 4, 12]. This paper focuses on the other direction, the transformation from an arbitrary (ordinary) bounded SPN system to an equivalent SPA model. We prove that SPA languages and bounded SPN (with exponential probability distribution functions) have theoretically equivalent expressiveness, if SPA is extended through a new operator. This is addressed by showing that transformations between the paradigms exist.

The first type of transformation we propose uses complementary places: the starting net system is augmented with one complementary place for each of its original places. The places in the net system provide the focus to derive the corresponding SPA model. It is observed that the case of arc multiplicities cannot be treated without extending the language. Hence a new operator is introduced—*generalised cooperation*—to cope with non-ordinary bounded SPN. Applying this automatic transformation, the behaviour of the net system and the corresponding SPA model is the same both from the functional and the temporal point of view. However, this solution lacks elegance and, whilst remaining faithful to the fine-grained structure of the net system, it loses the more intuitive, application-level structure.

In the second part of the paper we propose an improved transformation technique for ordinary nets, in which we can reduce the number of complementary places that need to be added. This second technique is intended to maintain more of the initial structure when possible, and not just to simulate the starting net. Some examples are used to show the existence of net constructions for which the SPA coding seems to be rather unnatural. The difficulties in mapping an arbitrary bounded SPN system into a SPA model suggest that, although both formalisms are based on relatively similar concepts, there are basic differences between them.

The balance of the paper is as follows. Section 2 provides some background material on both formalisms and Section 3 introduces the first transformation algorithm, the one based on the total complementation of the net places. The generalised cooperation operator is defined and the starting SPN system and the corresponding SPA model are shown to exhibit the same behaviour. In Section 4 an improved algorithm is discussed, whose application leads to simpler SPA models. Some problematic examples are also shown and in Section 5 the differences in the notion of process behind both formalisms are discussed. Finally, Section 6 concludes this work.

2 SPN and SPA notations

We assume the reader is familiar with SPN [1, 2, 11] and SPA [10, 5, 9]. We will just present the notation to be used, and in particular for SPA we will concentrate on Performance Evaluation Process Algebra (PEPA) [10].

2.1 Stochastic Petri nets

A Petri net (PN) is a $4-$tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post} \rangle$, where P, T are the sets of *places* and *transitions*, and $\mathbf{Pre}$ and $\mathbf{Post}$ are the $|P| \times |T|$ sized, natural valued, *incidence matrices*. The net is *ordinary* if all arc weights are one.

A *net system* is a pair $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0} \rangle$, where $\mathbf{m_0} \in I\!N^{|P|}$, is the *initial marking*. A transition t is *enabled* in a marking $\mathbf{m}$ if all its input places contain at least as many tokens as

*This work has been partially supported by Projects CICYT TAP98-0679 and HI-1998-0196.

specified by input arcs weights ($\mathbf{m} \geq \mathbf{Pre}[P,t]$). Once enabled, transition t can *fire*. The occurrence of t at marking $\mathbf{m}$, denoted $\mathbf{m}[\ t\ \rangle\mathbf{m}'$ yields the marking $\mathbf{m}' = \mathbf{m} + \mathbf{C}[P,t] = \mathbf{m} + \mathbf{Post}[P,t] - \mathbf{Pre}[P,t]$.

Under the assumption of boundedness, starting from the initial marking $\mathbf{m}_0$ it is theoretically possible to compute the set of all reachable markings, the so-called *reachability set*, $\mathrm{RS}(\mathbf{m}_0)$. The reachability set does not contain information about the transition sequences fired to reach each marking. This information is captured by the *reachability graph* (RG) whose nodes are labelled with the reachable markings and whose arcs are labelled with the transitions that the system has to fire to move from state to state.

A place whose removal preserves the fireable sequences (i.e., the interleaving semantics) of the system is called a *(sequential) implicit place* [6, 15]. If it also maintains the concurrent semantics it is called *concurrent implicit* [15]. It can be seen that in a net without self loops these concepts coincide. By means of linear relaxations, a sufficient structural condition for a place to be sequentially implicit can be stated, which can be verified in polynomial time (see [15]).

Proposition 2.1 ([15])

Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a net system. A place $p \in P$ with initial marking $\mathbf{m_0}[p] \geq z$, where z is the optimal value of (2.1), is (sequential) implicit:

$$
\begin{array}{rll}
z = & \textit{min.}\ \mathbf{y} \cdot \mathbf{m_0} + \mu & (2.1)\\
& \textit{s.t.}\ \ \mathbf{y} \cdot \mathbf{C} \leq \mathbf{C}[p,T] & \\
& \qquad \mathbf{y} \cdot \mathbf{Pre}[P,t] + \mu \geq \mathbf{Pre}[p,t] \quad \forall t \in p^\bullet & \\
& \qquad \mathbf{y} \geq \mathbf{0}, \mathbf{y}[p] = 0 &
\end{array}
$$

Problem (2.1) is feasible iff $\mathbf{y} \geq \mathbf{0}$ exists verifying $\mathbf{y}[p] = 0$ and $\mathbf{y} \cdot \mathbf{C} \leq \mathbf{C}[p,T]$. In other words, if $\mathbf{y}[p] = 0$ and $\mathbf{y} \cdot \mathbf{C} \leq \mathbf{C}[p,T]$, place p can be made implicit by choosing $\mathbf{m_0}[p]$ big enough. A place p for which the problem is feasible is called *structurally implicit*. Note that having "too many tokens" in an implicit place has no effect on the net system behaviour.

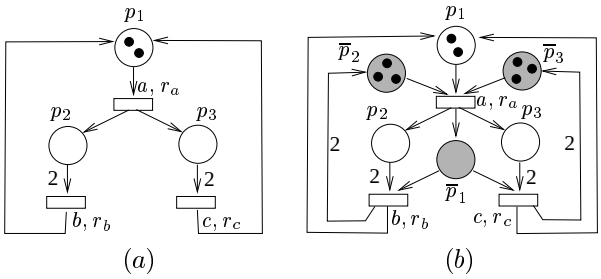


Figure 1. Implicit (or complementary) places

For example, the net system in Fig. 1(b) has been obtained by adding several implicit places (drawn in grey) to the net system in Fig. 1(a). Observe for instance place $\overline{p}_2$. It can be checked, using (2.1), that any marking of $\overline{p}_2$ greater than or equal to 3 will make it implicit. For $\overline{p}_1$, any marking greater than or equal to 0 will make it implicit.

A stochastic Petri net (SPN) system is given by a net structure plus an initial marking and a set of rates, i.e. $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}, \Lambda \rangle$ where $\Lambda : T \to \mathbb{R}^+$ assigns a rate to each transition, which determines the duration of the activity. The occurrence of t at marking $\mathbf{m}$ is denoted as $\mathbf{m}[\ (t, r_t)\ \rangle\mathbf{m}'$. Whenever more than one transition is enabled in a given marking, a *race condition* determines the one which will fire. The probability that a particular transition fires is given by the ratio of the transition rate to the sum of the rates of all the enabled transitions. Moreover, we will consider an infinite server semantics, that is, each transition can fire as many times as it is enabled. With this timing interpretation, the evolution of the discrete event system modelled by the PN can also be modelled using a Markov chain. It can be seen that this Markov chain is isomorphic to the reachability graph of the PN system, if the arcs are labelled with the rates of the transitions that are being fired [11].

2.2 Stochastic process algebras

Like in any stochastic process algebra (SPA), the basic elements of PEPA are *components* and *activities*. Each activity, a, is a pair (α, r) where α is the *action type* and r is the *activity rate*. Activities which are private to the component in which they occur are represented by the distinguished action type, τ. We assume that there is a countable set of components, which we denote $\mathcal{C}$, and a countable set, $\mathcal{A}$, of all possible action types. We denote by $\mathcal{A}ct \subseteq \mathcal{A} \times \mathbb{R}^+$, the set of activities, where $\mathbb{R}^+$ is the set of positive real numbers together with the symbol $\top$.

The syntax of PEPA expressions can be defined in terms of *sequential components* S and *model components* P:

$$
\begin{array}{rcl}
P & ::= & S \mid P \underset{L}{\bowtie} P \mid P/L \\
S & ::= & (\alpha, r).S \mid S + S \mid A
\end{array}
$$

The informal meaning of the combinators is:

Prefix: $(\alpha, r).P$ The component carries out activity (α, r) and subsequently behaves as P.

Choice: $P + Q$ The component represents a system which may behave either as component P or as Q: all the current activities of both components are enabled. The first activity to complete, determined by a *race condition*, distinguishes one component, the other is discarded.

Cooperation: $P \underset{L}{\bowtie} Q$ The components proceed independently with any activities whose types do not occur in the *cooperation set* L (*individual activities*). However, activities with action types in the set L require the simultaneous involvement of both components (*shared activities*). These activities are only enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q. When the set L is empty, we use the concise notation $P \parallel Q$ to represent $P \underset{\emptyset}{\bowtie} Q$. The main peculiarity of PEPA is that the shared activity occurs at the rate of the slowest participant. If an activity has an unspecified rate in a component, the component is *passive* with respect to that action type, i.e., the component does not influ-

ence the rate at which any shared activity occurs.

Hiding: P/L The component behaves as P but any activities of types within the set L are *hidden*, i.e. become τ. Such activities cannot be carried out in cooperation with any other component: the original action type of a hidden activity is not externally accessible; the duration is unaffected.

Constant: $A \stackrel{def}{=} P$ Constants are components whose meaning is given by a defining equation.

The action types which the component P may next engage in are the *current action types* of P, a set denoted $\mathcal{A}(P)$. This set is defined inductively over the syntactic constructs of the language (see [10]). The interpretation of PEPA expressions is given by a formal semantics, presented in the structured operational style [10].

The "states" of a PEPA model as it evolves are the syntactic terms, or derivatives. When a model component is defined it consists of one or more cooperating components, and they will be apparent in every derivative of the model, i.e. the global state will consist of a set of local states or derivatives, one for each sequential component.

The *derivation graph* $\mathcal{D}(C)$ is a graph in which syntactic terms form the nodes, and arcs represent the possible transitions between them: the operational rules define the form of this graph. It is isomorphic to the underlying Markov chain and can be regarded as analogous to the RG of a SPN.

Various forms of equivalence have been defined for PEPA. Here we use *strong equivalence* (denoted $\cong$). This is an observation-based equivalence: two components are considered equivalent if they are indistinguishable under external observation. This relation is a congruence, i.e. if a strongly equivalent component is substituted into a PEPA definition, the new definition is strongly equivalent to the original. Moreover, if the derivative set is partitioned by the relation, and a new model formed with one state for each equivalence class then the new model is strongly equivalent to the original. Formally, two PEPA components C_i, C_j are strongly equivalent if there is an equivalence relation between them such that, for any action type α, the total conditional transition rates from those components to any equivalence class, via activities of this type, are the same. The *total conditional transition rate* from a component P to a set of components S via an action type α is defined as $q[P, S, \alpha] = \sum_{Q \in S} q(P, Q, \alpha)$, where $q(P, Q, \alpha)$, is the sum of the activity rates labelling arcs connecting the corresponding nodes in the derivation graph which are also labelled by the action type α.

3 Equivalent expressiveness of bounded SPN and PEPA+

Since bounded SPN and PEPA models each have representations as finite Markov chains, a naive—and totally impractical—transformation from a bounded SPN could be constructed as follows: obtain the underlying Markov chain of the SPN [11] and define a single sequential PEPA component that directly encodes the Markov chain.

However, we aim to establish a more structural transformation from bounded SPN to PEPA. Ideally we would like to find a structural concept in the SPN to correspond with a certain structure in the PEPA model, the sequential components, thus achieving a *process oriented implementation* of bounded SPN models in PEPA models. Unfortunately, this direct mapping does not work for all bounded SPN but only for some net subclasses, thus suggesting the existence of a different notion of process behind the two formalisms. The problem addressed here is, in some sense, a classical one in the PN paradigm, related to the net system decomposition into sequential components running in parallel[14, 7]. (see [14], chapter 7, for the implementation of 1-bounded ordinary net systems by means of microprogramming techniques, or [7] for a software-based implementation using Ada-like languages).

3.1 Place complementation in SPN and its implementation in PEPA

In this section we first explain the complementation of ordinary bounded Petri nets. This is a structural transformation by which we add a *complementary* place for every place of the net. The initial marking of the complementary places will be defined in such a way that they do not constrain the behaviour of the transitions they are connected to: so the behaviour of the net system is preserved under true concurrency. Although in the Markovian setting only interleaving semantics is relevant, this means that the same transformation can be applied if a different timing interpretation is used.

For each place p, we add its *complement*, which we denote as $\bar{p}$, such that $\bar{p}^\bullet = {}^\bullet p$, ${}^\bullet\bar{p} = p^\bullet$. The initial marking of this place will be $\mathbf{m}_0[\bar{p}] = \mathbf{b}[p] - \mathbf{m}_0[p]$, where $\mathbf{b}[p]$ denotes an upper bound of the number of tokens which may be present in place p. Provided with these markings, the complementary places are implicit places [15]. If the net is structurally bounded, i.e., bounded for every initial marking (there exists $\mathbf{y} > \mathbf{0}$ such that $\mathbf{y} \cdot \mathbf{C} \leq \mathbf{0}$) complementary places are structurally implicit. Hence, a possible marking for $\bar{p}$ is the one obtained applying (2.1). In general, this marking is not the minimal one that makes the place implicit, and improved techniques can be applied to get a tighter bound [15].

Definition 3.1 (Complementation) *Given a SPN $\mathcal{S}_1 = \langle P_1, T_1, \mathbf{Pre}_1, \mathbf{Post}_1, \mathbf{m}_{01}, \Lambda_1 \rangle$ its complementation is another SPN, $\mathcal{S}_2 = \langle P_2, T_2, \mathbf{Pre}_2, \mathbf{Post}_2, \mathbf{m}_{02}, \Lambda_2 \rangle$ with*

1. *$P_2 = P_1 \cup \overline{P}_1$ where $\overline{P}_1$ is the set of added places, one for each $p \in P_1$, such that $P_1 \cap \overline{P}_1 = \emptyset$*
2. *$T_2 = T_1$*
3. $\mathbf{Pre}_2 = \begin{bmatrix} \mathbf{Pre}_2[P_1, T_1] \\ \mathbf{Pre}_2[\overline{P}_1, T_1] \end{bmatrix} = \begin{bmatrix} \mathbf{Pre}_1[P_1, T_1] \\ \mathbf{Post}_1[P_1, T_1] \end{bmatrix}$

4. $\mathbf{Post}_2 = \begin{bmatrix} \mathbf{Post}_2[P_1, T_1] \\ \mathbf{Post}_2[\overline{P}_1, T_1] \end{bmatrix} = \begin{bmatrix} \mathbf{Post}_1[P_1, T_1] \\ \mathbf{Pre}_1[P_1, T_1] \end{bmatrix}$

5. $\Lambda_2 = \Lambda_1$

6. $\mathbf{m_{02}} = \begin{bmatrix} \mathbf{m}_{02}[P_1] \\ \mathbf{m}_{02}[\overline{P}_1] \end{bmatrix} = \begin{bmatrix} \mathbf{m}_{01}[P_1] \\ \mathbf{b}[P_1] - \mathbf{m}_{01}[P_1] \end{bmatrix}$

Example 1 Fig. 2(a) shows a simple SPN system and Fig. 2(b) shows the same system augmented with all complementary places (which are drawn in grey). In this trivial case, both models are 1-bounded.

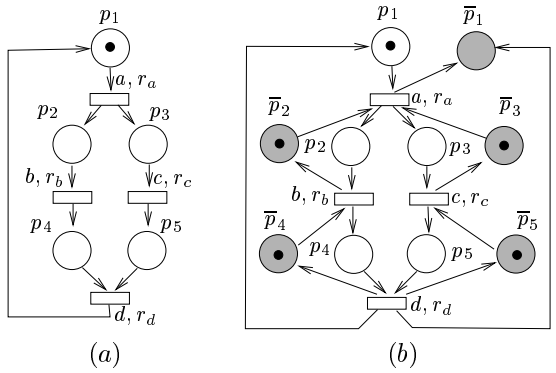


Figure 2. Complementary places in the ordinary case

Given a complemented net system we can generate a corresponding PEPA model by considering the set of state machines composed of each place and its complement. Being 1-bounded, each state machine is mapped onto a PEPA sequential component, and we need to consider one instance of the component for each token in each place. Synchronisations over net transitions become cooperations between sequential components. As we have noted earlier, adding too many tokens in complementary places does not change the net system behaviour. However, this impacts on the PEPA behavioural model since "extra tokens" are mapped onto redundant instances of the same component causing an unnecessary exponential growth of the state space. Fig. 3 shows an abstract algorithm for the translation technique we have informally discussed. Applying the algorithm to the complemented net in Fig. 2(b) we obtain:

First step: Sequential components

$$P_1 \stackrel{def}{=} (a, r_a).\overline{P}_1 \quad \overline{P}_1 \stackrel{def}{=} (d, \top).P_1 \quad P_2 \stackrel{def}{=} (b, r_b).\overline{P}_2$$
$$\overline{P}_2 \stackrel{def}{=} (a, \top).P_2 \quad P_3 \stackrel{def}{=} (c, r_c).\overline{P}_3 \quad \overline{P}_3 \stackrel{def}{=} (a, \top).P_3$$
$$P_4 \stackrel{def}{=} (d, r_d).\overline{P}_4 \quad \overline{P}_4 \stackrel{def}{=} (b, \top).P_4 \quad P_5 \stackrel{def}{=} (d, r_d).\overline{P}_5$$
$$\overline{P}_5 \stackrel{def}{=} (c, \top).P_3$$

Second step: Model equation

$$M_1 \stackrel{def}{=} P_1, \quad M_2 \stackrel{def}{=} \overline{P}_2, \quad M_3 \stackrel{def}{=} \overline{P}_3,$$
$$M_4 \stackrel{def}{=} \overline{P}_4, \quad M_5 \stackrel{def}{=} \overline{P}_5,$$

$$L_1 = \{a, d\}, \quad count_1 = (1, 0, 0, 1)$$
$$L_2 = \{a, b\}, \quad count_2 = (2, 1, 0, 1)$$
$$L_3 = \{c\}, \quad count_3 = (2, 1, 1, 1)$$
$$L_4 = \{d\}, \quad count_4 = (2, 1, 1, 2)$$
$$L_5 = \emptyset, \quad count_5 = (2, 1, 1, 2)$$

$$Sys_1 \stackrel{def}{=} M_1, \quad Sys_2 \stackrel{def}{=} Sys_1 \underset{\{a,d\}}{\bowtie} M_2,$$
$$Sys_3 \stackrel{def}{=} Sys_2 \underset{\{a,b\}}{\bowtie} M_3, \quad Sys_4 \stackrel{def}{=} Sys_3 \underset{\{c\}}{\bowtie} M_4,$$
$$Sys_5 \stackrel{def}{=} Sys_4 \underset{\{d\}}{\bowtie} M_5$$

$$\mathcal{M} \equiv P_1 \underset{\{a,d\}}{\bowtie} \overline{P}_2 \underset{\{a,b\}}{\bowtie} \overline{P}_3 \underset{\{c\}}{\bowtie} \overline{P}_4 \underset{\{d\}}{\bowtie} \overline{P}_5$$

Observe that although the structure of PEPA terms is rather linear, such terms are capable of capturing arbitrary connectivity in the SPN. We are simulating the structure of the net (augmented with implicit places), and not just its underlying reachability graph.

Example 2 Consider the non ordinary net system in Fig. 1(a), in which some of the arcs have a weight equal to two. We will see that in this case the addition of complementary places is not enough to allow us to represent the same net system. During its evolution, the net system augmented with complementary places (Fig. 1(b)) can reach a marking with three tokens in place p_2 (p_3) and one token in p_3 (p_2), and the enabled transition b (c) consumes *two* out of *three* tokens when firing. By applying the steps outlined in the algorithm of Fig. 3 we obtain:

First step: sequential components

$$P_1 \stackrel{def}{=} (a, r_a).\overline{P}_1 \quad \overline{P}_1 \stackrel{def}{=} (b, \top).P_1 + (c, \top).P_1$$
$$P_2 \stackrel{def}{=} (b, r_b).\overline{P}_2 \quad \overline{P}_2 \stackrel{def}{=} (a, \top).P_2$$
$$P_3 \stackrel{def}{=} (c, r_c).\overline{P}_3 \quad \overline{P}_3 \stackrel{def}{=} (a, \top).P_3$$

Second step: model equation

$$M_1 \stackrel{def}{=} P_1 \parallel P_1, \quad M_2 \stackrel{def}{=} \overline{P}_2 \parallel \overline{P}_2 \parallel \overline{P}_2,$$
$$M_3 \stackrel{def}{=} \overline{P}_3 \parallel \overline{P}_3 \parallel \overline{P}_3$$

$$L_1 = \{a, b, c\}, \quad count_1 = (1, 1, 1)$$
$$L_2 = \{a\}, \quad count_2 = (2, 1, 1)$$
$$L_3 = \emptyset, \quad count_3 = (2, 1, 1)$$

$$Sys_1 \stackrel{def}{=} M_1, \quad Sys_2 \stackrel{def}{=} Sys_1 \underset{\{a,b,c\}}{\bowtie} M_2,$$
$$Sys_3 \stackrel{def}{=} Sys_2 \underset{\{a\}}{\bowtie} M_3$$

$$\mathcal{M} \equiv (P_1 \parallel P_1) \underset{\{a,b,c\}}{\bowtie} (\overline{P}_2 \parallel \overline{P}_2 \parallel \overline{P}_2) \underset{\{a\}}{\bowtie} (\overline{P}_3 \parallel \overline{P}_3 \parallel \overline{P}_3)$$

The PEPA model above does not represent the net system of Fig. 1(b) because it is not possible to observe states in which *two* out of *three* components P_2 (P_3) synchronise

Given a net system $\mathcal{S} = \langle P \cup \overline{P}, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m}_0, \Lambda \rangle$, perform the following steps

Step 1: Derivation of sequential components

For each pair $p_i \in P$, $\overline{p}_i \in \overline{P}$, define PEPA sequential components P_i and $\overline{P_i}$ as follows

1. for each transition $t \in p_i^\bullet$, the PEPA sequential component P_i enables an activity of type t with the same rate as t with derivative $\overline{P_i}$; $\mathcal{A}(P_i) = \{t \mid t \in p_i^\bullet\}$
2. for each transition $t \in {}^\bullet p_i$, the PEPA sequential component $\overline{P_i}$ enables a passive activity of type t with derivative P_i; $\mathcal{A}(\overline{P}_i) = \{t \mid t \in {}^\bullet p_i\}$

Step 2: Model equation

1. for each p_i define a model component $M_i \stackrel{def}{=} P_i \parallel \ldots \parallel P_i \parallel \overline{P_i} \parallel \ldots \parallel \overline{P_i}$ where the number of replica of P_i and $\overline{P_i}$ is given by $\mathbf{m}_0[p_i]$ and $\mathbf{m}_0[\overline{p}_i]$.
2. the cooperation sets and the system equation are built up recursively, starting from a base case. We make use of an auxilliary array of size $|T|$, $count_i(t)$, to keep track of the appearances of each transition/action type within cooperation sets.

$$
\begin{aligned}
L_1 &= \mathcal{A}(P_1) \cup \mathcal{A}(\overline{P_1}) \\
count_1(t) &= \begin{cases} 1 & \text{if } t \in L_1 \\ 0 & \text{otherwise} \end{cases} \qquad \forall t \in T \\
Sys_1 &\stackrel{def}{=} M_1 \\
L_i &= \mathcal{A}(P_i) \cup \mathcal{A}(\overline{P_i}) \setminus \{t \mid t \in \bigcup_{j=1}^{i-1} L_j \wedge count_{i-1}(t) = |{}^\bullet t| - 1\} \\
count_i(t) &= \begin{cases} count_{i-1}(t) + 1 & \text{if } t \in L_i \\ count_{i-1}(t) & \text{otherwise} \end{cases} \qquad \forall t \in T \\
Sys_i &\stackrel{def}{=} Sys_{i-1} \underset{L_{i-1}}{\bowtie} M_i \\
\mathcal{M} &\equiv Sys_{|P|}
\end{aligned}
$$

Figure 3. PEPA model construction for (complemented) ordinary SPN

over action type b (c). For example, the current representation of M_2, $\overline{P}_2 \parallel \overline{P}_2 \parallel \overline{P}_2$, represents the case in which the P_2 components do not synchronise at all, i.e. as if the arc from p_2 to b had multiplicity 1. If we chose instead to represent M_2 by $\overline{P}_2 \underset{\{b\}}{\bowtie} \overline{P}_2 \underset{\{b\}}{\bowtie} \overline{P}_2$ we would capture the case when the arc had multiplicity 3. If we defined M_2 to be $\overline{P}_2 \underset{\{b\}}{\bowtie} \overline{P}_2 \parallel \overline{P}_2$ two instances of P_2 must synchronise on b, but then two instances are fixed. In contrast in the SPN the tokens have no identity and the transition can be fired by any two tokens out of the possible three. Thus, current PEPA notation is *not capable of expressing the case* represented by such a non ordinary net system (except by a flat purely sequentialised model representing the state space.) To model such a behaviour we need to introduce a new operator.

3.2 Defining PEPA+

We introduce a more general form of interaction between PEPA components by defining a new combinator, *generalised cooperation*. Whereas previous PEPA combinators have been unary (e.g. hiding) or binary (e.g. choice), generalised cooperation deals with a set of components. Given n components cooperating over an action type α, the combinator is able to select k out of n components for the actual execution of the shared activity.

Let Ω be a *multiset* of PEPA components and L be a *multiset* of action types. Then $\underset{L}{\bowtie} \Omega$ denotes the generalised cooperation of the components in Ω over the types in L. Informally, this can be interpreted as follows. For all action types which do not appear in L, components in Ω proceed independently and concurrently (*individual activities*). For action types in L with multiplicity n ($n \geq 1$), $n+1$ components from the multiset Ω must cooperate (*shared activities*). Thus when $\Omega = \{P, Q\}$, and all elements of L appear with multiplicity one, then $\underset{L}{\bowtie} \Omega = P \underset{L}{\bowtie} Q$; when $L = \emptyset$, then $\underset{L}{\bowtie} \Omega = P \parallel Q$. The formal semantics of generalised cooperation is presented in Fig. 4.

Following the construction in Fig. 3, we see that increasing the marking of a place is represented by increasing the number of instances of this component in the intermediate component M_i. When we wish to represent arcs with multiplicity greater than one, we change this parallel com-

$$\frac{P \xrightarrow{(\alpha,r)} P'}{\Omega \xrightarrow{(\alpha,r)} \Omega'}\ (P \in \Omega), \qquad \text{where } \Omega' = \Omega \setminus P \cup P'$$

$$\frac{P \xrightarrow{(\alpha,r)} P'}{\bowtie_L P \cup \Omega \xrightarrow{(\alpha,r)} \bowtie_L P' \cup \Omega}\ (\alpha \notin L) \qquad \frac{\bowtie_L \Omega \xrightarrow{(\alpha,r)} \bowtie_L \Omega'}{\bowtie_L P \cup \Omega \xrightarrow{(\alpha,r)} \bowtie_L P \cup \Omega'}$$

$$\frac{P \xrightarrow{(\alpha,r_1)} P' \quad \bowtie_{L\setminus\{\alpha\}} \Omega \xrightarrow{(\alpha,r_2)} \bowtie_{L\setminus\{\alpha\}} \Omega'}{\bowtie_L P \cup \Omega \xrightarrow{(\alpha,R)} \bowtie_L P' \cup \Omega'}\ (\alpha \in L), \qquad R = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(\bowtie_L \Omega)} \min(r_\alpha(P), r_\alpha(\bowtie_L \Omega))$$

Figure 4. Structured operational semantic rules for generalised cooperation

position of instances to be a generalised cooperation over these instances, requiring cooperation of the same number of instances as the multiplicity of the arc. For instance, the PEPA+ components for the net system in Fig. 1(b) are:

$$M_1 \stackrel{def}{=} \bowtie_{\emptyset} \{P_1, P_1\}, \qquad M_2 \stackrel{def}{=} \bowtie_{\{b\}} \{\overline{P}_2, \overline{P}_2, \overline{P}_2\},$$
$$M_3 \stackrel{def}{=} \bowtie_{\{c\}} \{\overline{P}_3, \overline{P}_3, \overline{P}_3\}$$

and so we have

$$\mathcal{M} \equiv (\bowtie_{\emptyset} \{P_1, P_1\}) \bowtie_{\{a,b,c\}} (\bowtie_{\{b\}} \{\overline{P}_2, \overline{P}_2, \overline{P}_2\}) \bowtie_{\{a\}} (\bowtie_{\{c\}} \{\overline{P}_3, \overline{P}_3, \overline{P}_3\})$$

See Fig. 5 for the modified algorithm. Observe that also in this general case, we are simulating the structure of the net, and not just its underlying reachability graph.

Proposition 3.1 *For any bounded SPN system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m}_0, \Lambda \rangle$ and the PEPA model $\mathcal{M}$ obtained by the algorithm presented in Fig. 5, $RG(\mathcal{S}) = \mathcal{D}(\mathcal{M})/\cong$, where "$=$" denotes graph isomorphism, and $\mathcal{D}(\mathcal{M})/\cong$ is the quotient set of $\mathcal{M}$ w.r.t. strong equivalence.*

4 Reducing the number of implicit places

The transformation technique we have discussed in Section 3.1 can be applied to any bounded SPN model but has two disadvantages:

1. The structure of the starting net, in terms of its sequential processes, is lost, because for each place we consider a subnet composed of that place and the corresponding complementary one.
2. The resulting PEPA model is *verbose*, since for each place we add a PEPA component.

From a Petri net point of view, the closest object to a PEPA sequential component is a state machine. A PEPA model can be seen as a superposed automata, a particular class of net systems, obtained by composing state machines over common transitions. We have seen that it is possible to decompose an ordinary net into state machines by adding one complementary place for each place. These are then translated into PEPA basic sequential components, the number of copies (instances) being $\mathbf{m}_0[p] + \mathbf{m}_0[\overline{p}]$. More generally setting we would like to be able to partition the net into state machines by adding a smaller number of complementary places, as we shall discuss in the following sections.

4.1 The case of ordinary nets

Let us go back to Example 1: in Fig. 2(b) we have added one complementary place for each starting place. However, as shown in Fig. 6, a single place is sufficient to decompose the net into two state machines, the former composed by places p_1, p_2, p_4, the latter formed by $\overline{p}_{35}, p_3, p_5$.

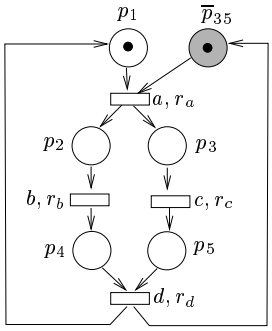


Figure 6. Reducing the number of complementary places for the net system in Fig. 2(a)

We present in this section an abstract greedy algorithm (see Fig. 8) that can be used for adding complementary places into ordinary net systems. The key idea is to decompose the net into a set of subnets which can be "closed" by the addition of some complementary places, becoming state machines. These subnets may be obtained using a *compatibility relationship*, defined as follows.

Definition 4.1 (Compatibility) *Two places are compatible if and only if they do not have any common input transition*

Given a net system $\mathcal{S} = \langle P \cup \overline{P}, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m}_0, \Lambda \rangle$, perform the following steps

Step 1: Derivation of sequential components

For each pair $p_i \in P$, $\overline{p}_i \in \overline{P}$ define PEPA components P_i and $\overline{P}_i$ as in the algorithm described in Fig. 3:

Step 2: Model equation

1. for each p_i, define a model component $M_i \stackrel{def}{=} \underset{K_i}{\bowtie} \mathcal{S}_i$ where

$$\begin{array}{rclll} \mathcal{S}_i(P_i) & = & \mathbf{m}_0[p_i] & & \\ \mathcal{S}_i(\overline{P}_i) & = & \mathbf{m}_0[\overline{p}_i] & & \\ \mathcal{S}_i(Q) & = & 0 & \text{otherwise} & \end{array} \qquad \begin{array}{rcll} K_i(t) & = & \mathbf{Pre}[p_i, t] - 1 & \forall t \in p_i^\bullet \\ K_i(t') & = & \mathbf{Post}[p_i, t'] - 1 & \forall t' \in {}^\bullet p_i \\ K_i(t'') & = & 0 & \text{otherwise.} \end{array}$$

2. The cooperation sets and the system equation are built up recursively, as in the algorithm in Fig. 3.

Figure 5. PEPA model construction for an arbitrary bounded SPN

(i.e., a fork) or output transition (i.e., join).

In order to reduce the number of machines running in parallel, hence the complexity of the decomposition, for each place we can compute a maximal connected subnet that fulfils the previous relationship. Then, if necessary, the subnet is closed by adding a place in such a way that each transition has one input and one output place. The algorithm should proceed until all the original places have been considered, i.e. until the decomposition covers the initial net. The abstract code of the algorithm is shown in Fig. 8.

Let us show how the algorithm works on the example of Fig. 7(a). We can start with p_1 and add p_3 and p_4, since the subnet defined by p_1, p_3, p_4 and their input and output transitions have no forks or joins. However, we cannot go on: if p_2 were added, there would be a join in b, while if p_5 were added there would be a join in c. This component is then completed by adding the complementary place $\overline{p}_{134}$, as shown in Fig. 7(b) (for readability, arcs belonging to different state machines are drawn differently.) The remaining places, p_2 and p_5 are not compatible (f is an output transition for both of them), hence we need two more state machines, one formed by p_2 and $\overline{p}_2$, and one formed by p_5 and $\overline{p}_5$. Applying the linear programming problem in (2.1), the initial marking we obtain is: zero tokens in $\overline{p}_{134}$, two tokens in $\overline{p}_2$, and one token in $\overline{p}_5$. It can be seen that in fact one token in $\overline{p}_2$ would be enough to make it implicit [15], but we do not address here the technicalities for "optimal" simulation. From each state machine we derive the corresponding PEPA sequential component and we then map synchronisations on transitions onto cooperations. We obtain:

$$P_1 \stackrel{def}{=} (a, r_a).P_3, \qquad P_2 \stackrel{def}{=} (b, r_b).\overline{P}_2 + (f, r_f).\overline{P}_2$$
$$\overline{P}_2 \stackrel{def}{=} (a, r_a).P_2 + (e, r_e).P_2, \; P_3 \stackrel{def}{=} (b, r_b).P_4 + (c, r_c).P_1$$
$$P_4 \stackrel{def}{=} (d, r_d).P_3 + (e, r_e).\overline{P}_{134}, \;\; \overline{P}_{134} \stackrel{def}{=} (f, r_f).P_1$$
$$P_5 \stackrel{def}{=} (c, r_c).\overline{P}_5 + (f, r_f).\overline{P}_5, \;\; \overline{P}_5 \stackrel{def}{=} (b, r_b).P_5$$

$$\mathcal{M} \stackrel{def}{=} (\overline{P}_2 \parallel \overline{P}_2) \underset{\{a,b,e,f\}}{\bowtie} P_1 \underset{\{b,c,f\}}{\bowtie} \overline{P}_5$$

With respect to the "verbose coding" of Section 3.1 we have now three different components with two copies (instances) of one of them, instead of the five components that would have been obtained by applying the algorithm of Fig. 3. Moreover, as we said, one of the copies of $\overline{P}_2$ is not needed and improved techniques could have been used to detect this before the implementation, thus obtaining a simpler model: $\mathcal{M} \stackrel{def}{=} \overline{P}_2 \underset{\{a,b,e,f\}}{\bowtie} P_1 \underset{\{b,c,f\}}{\bowtie} \overline{P}_5$.

Notice that in general many different decompositions may exist. The proposed greedy algorithm does not guarantee to find the minimal number of sequential machines, although variants can be obtained for sub-optimality. In the worst case, the number of complementary places equals the number of starting places (i.e., we have the same size as the "verbose" net), but usually it is smaller.

4.2 The case of non ordinary nets

The case of non ordinary nets is much more complex. For instance, the net system in Fig. 9 is not conservative in a strict sense, i.e. the total number of tokens is changed when some transitions are fired. This can be understood as a change of the "objects" represented by the tokens. This concept has no corresponding counterpart in PEPA (in process algebra in general) since tokens can be interpreted as different states within the same process. This problem can be theoretically avoided by simulating the weighted net system using a corresponding ordinary one (that can always be obtained), or by using the "verbose" simulation by complementation as presented in Section 3. However, notice that this illustrates the fact that we are dealing with two different notions of process.

5 Process interpretations in SPN and SPA

Processes are a basic concept in process algebra. There is also a well-known notion of process which has been defined for ordinary 1-bounded (S)PN [13], but it is a com-

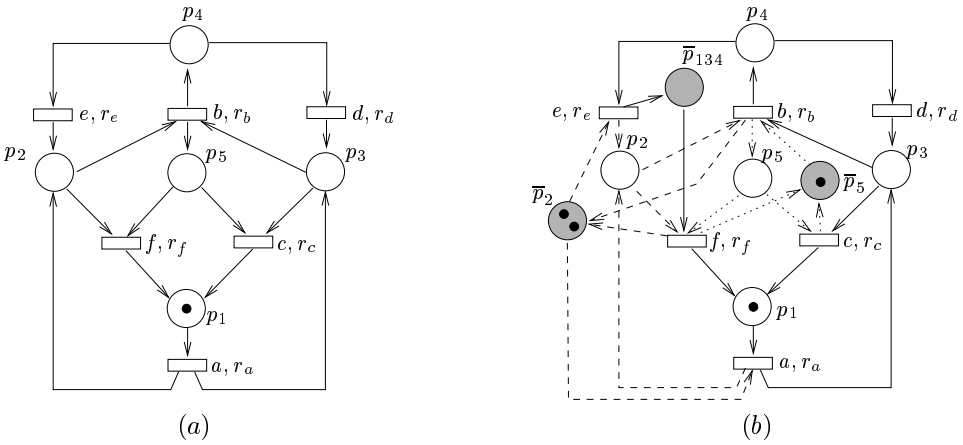


Figure 7. State machine decomposition

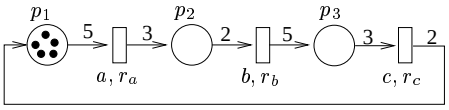


Figure 9. Non ordinary net

pletely different idea. In the Petri net context, a process is a purely concurrent trace generated by the dynamics of a net system. Thus, conflicts do not appear, that is, all decisions are taken. Processes exhibit all concurrency and cannot be cyclic. They can be represented with particular infinite marked graphs, called *occurrence nets* [13]. In PEPA instead, a basic process is a sequential component that can perform actions and take decisions. Concurrency is introduced by the parallel composition of several sequential components. Hence, the same name is used for two quite different concepts, which are difficult to compare.

Moreover, also Petri net structures that look similar to PEPA basic processes may happen to be quite different. In the following, a simple example will be used to illustrate the difference. Certainly general conclusions cannot be obtained from just one example, but some interesting issues are raised here that may help to understand where the differences between these two formalisms lie.

The net system in Fig. 10(a) describes three machines (modelled by transitions a, b, c). The first two machines are fed by two input buffers (places p_1 and p_2) and perform similar actions. Machine 3 requires two objects (e.g. pallets) of any kind, prepared by the previous machines. After performing its actions on these two objects, it sends one pallet to each buffer. To find a corresponding PEPA model, we can complement the net and then apply the algorithm of Fig. 5. However, as we have already pointed out, this is not a nice or elegant solution and we are looking for less "verbose codings".

When the initial marking is $\mathbf{m}_0 = \{p_1, p_2\}$, this net system is equivalent to the one in Fig. 10(b), obtained adding

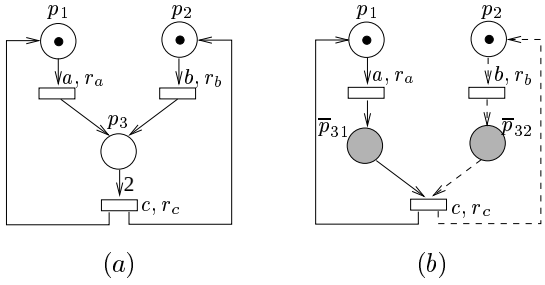


Figure 10. A problematic net system for $\mathbf{m}_0[p_1] + \mathbf{m}_0[p_2] \geq 3$

the complementary places $\overline{p}_{31}$ and $\overline{p}_{32}$, and then removing p_3, which has become implicit. Moreover, the derived PEPA model

$$\begin{array}{cccccc} P_1 & \stackrel{def}{=} & (a, r_a).\overline{P}_1 & \overline{P}_1 & \stackrel{def}{=} & (c, r_c).P_1 \\ P_2 & \stackrel{def}{=} & (b, r_b).\overline{P}_2 & \overline{P}_2 & \stackrel{def}{=} & (c, r_c).P_2 \end{array}$$

$$\mathcal{M} \stackrel{def}{=} P_1 \underset{\{c\}}{\bowtie} P_2$$

is equivalent to the net system, and not only its simulation by complementation.

However, a minor change in the initial marking causes several problems. Consider for example $\mathbf{m}_0 = \{2p_1, p_2\}$. Now, even if we add $\overline{p}_{31}$ and $\overline{p}_{32}$, place p_3 is not implicit, and cannot be removed as it was before. In this case we are not able to find a "conceptually" equivalent PEPA model.

Observe that the difficulties have appeared when a place has been marked with more than one token. In general, multiple tokens within a place do not constitute a real problem since they can be mapped onto repeated instances of the same component. In this case however, repeating an instance of P_1 is not enough. Not even if generalised cooperation is used, since there are some markings in the net which have no corresponding states in $\underset{c}{\bowtie}\{P_1, P_1, P_2\}$. For example, $\mathbf{m} = \{p_1, 2p_2\}$ is not "reachable" in the PEPA model because there is no state $\underset{c}{\bowtie}\{P_1, P_2, P_2\}$.

Consider a net system $\mathcal{S} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m}_0, \Lambda \rangle$ and perform the following steps:

1. $P_{temp} = P$
2. **repeat**
   (a) select $p_i \in P_{temp}$ and compute a maximal connected subnet that contains p_i satisfying the compatibility relationship, and let $S_i = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$ be the set of places in this subnet
   (b) **if** the subnet is not closed (${}^\bullet S_i \neq S_i^\bullet$) **then**
      i. add a complementary place $\overline{p}_{i_1 i_2 \cdots i_k}$ and connect it to the transitions in the subnet in such a way that each transition has exactly one input and one output arc
      ii. compute the initial marking of the added place to make it implicit (notice that this can be done by solving the linear programming problem in (2.1))
   (c) $P_{temp} = P_{temp} \setminus S_i$

   until $P_{temp} = \emptyset$
3. For each subnet build the corresponding PEPA sequential component considering the places in the subnet and their input and output transitions. Notice that these subnets are in fact state machines and therefore no synchronisation appears.
4. Synchronise all the PEPA sequential components on their common transitions as already defined in the second part of Step 2 of Fig. 3.

Figure 8. Greedy algorithm for state machines computation

We may think that the problem is due to having arc weights, and construct an ordinary net system with the same interleaving semantics, i.e., the same firing language (see Fig. 11(a)). Unfortunately, the problem still remains: When moving to PEPA we cannot model the "swapping" of the pallets between the two buffers. Again, we need to introduce implicit places (Fig. 11(b)) to obtain a corresponding PEPA model (Fig. 11(c)).

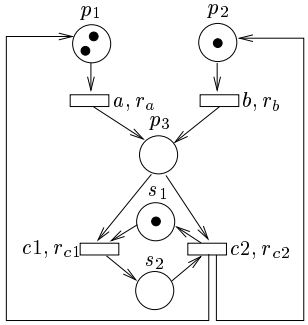
With a slight modification of the system in Fig. 11(a) we obtain the one of Fig. 12(a). In this case we have sequentialised the return of raw material to the buffers deciding that machine 3 returns material first to P_1 and then to P_2. With this small net modification we change the global behaviour of the system, since now interleaving between consumption of objects and return of raw material may be observed. However now we are able to derive a PEPA model (Fig. 12(b)) using only two state machines, one representing both types of products, the other responsible for the sequentialisation in the return of raw material, instead of three that were necessary for the net system in Fig. 11(a). Notice that we can exchange the returning sequence and nothing changes, but we cannot observe concurrency when returning raw material back to the two buffers.

6 Conclusions and further work

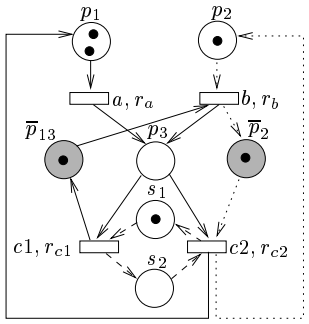
We have presented a sound relationship between bounded SPN and PEPA, thus establishing a bridge between these modelling paradigms. This is achieved via a transformation of bounded SPN systems, augmented with complementary places, into corresponding PEPA+ models, an extension of previous PEPA. The technique can be easily adapted to the other SPA languages by suitably changing the rates computed for synchronising transitions and by introducing immediate transitions when necessary.

This bridge opens the possibility that some of the powerful theoretical results which have been developed for SPA may be exploited in SPN. For example the use of modal logics to specify performance measures [8]. Moreover, this relationship between the two formalisms offers the potential for a multi-paradigm modelling approach, in which the modeller would be free to adopt a component-based approach, choosing the most appropriate paradigm for each model component, depending on his personal preferences.

Still, subtle differences emerge between the two paradigms. The notion of *process* underlying the two formalisms is different and this seems worthy of further investigation. Whilst both formalisms were developed to aid in the analysis of system properties such as concurrency and causality, process algebras which have their origins in semantics, are perhaps closer to a programming language perspective. In this context a component can be seen as an abstract program with its program counter—the current derivative captures the local state. Whilst there are similarities in the notion of local state represented by the marking of a Petri net, the tokens do not necessarily represent the state



(a)

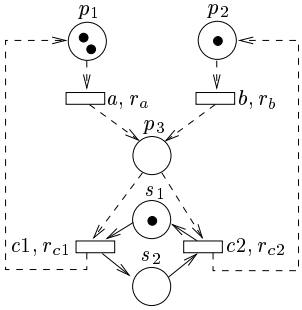


(b)

$$
\begin{array}{rcl}
P_1 & \stackrel{def}{=} & (a, r_a).P_3 \\
P_3 & \stackrel{def}{=} & (c1, r_{c1}).\overline{P}_{13} + (c2, r_{c2}).P_1 \\
\overline{P}_{13} & \stackrel{def}{=} & (b, r_b).P_3 \\
P_2 & \stackrel{def}{=} & (b, r_b).\overline{P}_2 \\
\overline{P}_2 & \stackrel{def}{=} & (c2, r_{c2}).P_2 \\
S_1 & \stackrel{def}{=} & (c1, r_{c1}).S_2 \\
S_2 & \stackrel{def}{=} & (c2, r_{c2}).S_1 \\
\mathcal{M} & \stackrel{def}{=} & (P_1 \parallel P_1 \parallel \overline{P}_{13}) \underset{\{b,c2\}}{\bowtie} \\
 & & \qquad (P_2 \parallel \overline{P}_2) \underset{\{c1,c2\}}{\bowtie} S_1
\end{array}
$$

(c)

Figure 11. Simulation of the net in Fig. 10(a) with $\mathbf{m}_0 = \{2p_1, p_2\}$, by means of an ordinary one



(a)

$$
\begin{array}{rcl}
P_1 & \stackrel{def}{=} & (a, r_a).P_3 \\
P_2 & \stackrel{def}{=} & (b, r_b).P_3 \\
P_3 & \stackrel{def}{=} & (c1, r_{c1}).P_1 + (c2, r_{c2}).P_2 \\
S_1 & \stackrel{def}{=} & (c1, r_{c1}).S_2 \\
S_2 & \stackrel{def}{=} & (c2, r_{c2}).S_1 \\
\mathcal{M} & \stackrel{def}{=} & (P_1 \parallel P_1 \parallel P_2) \underset{\{c1,c2\}}{\bowtie} S_1
\end{array}
$$

(b)

Figure 12. Ordinary non problematic net (but not exactly the same system)

of a single sequential component. Moreover, their meaning may vary depending on the place to which they belong.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley, 1995.

[2] G. Balbo and M. Silva, editors. *Proc. of Human Capital and Mobility MATCH—Performance Advanced School*, Jaca, Spain, 1998.

[3] M. Bernardo. *Theory and Application of Extended Markovian Process Algebra*. PhD thesis, Universitá di Bologna, February 1999.

[4] M. Bernardo, N. Busi, and R. Gorrieri. A distributed semantics for EMPA based on stochastic contextual nets. *The Computer Journal*, 38(6), 1995. Special Issue: Proc. of 3rd Process Algebra and Performance Modelling Workshop.

[5] M. Bernardo and R. Gorrieri. A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time. *Theoretical Computer Science*, 201:1–54, July 1998.

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

[7] J. M. Colom, M. Silva, and J. L. Villarroel. On software implementation of Petri nets and colored Petri nets using high-level concurrent languages. In *Proc. 7^{th} European Workshop on Application and Theory of Petri Nets*, pages 207–241, Oxford, England, July 1986.

[8] J. H. G. Clark, S. Gilmore and M. Ribaudo. Exploiting Modal Logic to Express Performance Measures. In *Proc. of 11^{th} Int. Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Chicago, USA, March 2000.

[9] H. Hermanns. *Interactive Markov Chains*. PhD thesis, Universität Erlangen-Nürnberg, September 1999.

[10] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[11] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Trans. on Computers*, 31(9):913–917, 1982.

[12] M. Ribaudo. *On the relationship between Stochastic Petri Nets and Stochastic Process Algebras*. PhD thesis, Universitá di Torino, April 1995.

[13] G. Rozenberg. Behaviour of elementary net systems. In *Petri Nets: Central Models and Their Properties*, volume 254 of *LNCS*. Springer Verlag, 1986.

[14] M. Silva. *Las Redes de Petri: en la Automática y la Informática*. AC, 1985.

[15] M. Silva, E. Teruel, and J. M. Colom. Linear algebraic and linear programming techniques for the analysis of net systems. In G. Rozenberg and W. Reisig, editors, *Lectures in Petri Nets. I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 309–373. Springer, 1998.