Well-Defined Generalized Stochastic Petri Nets: A Net-Level Method to Specify Priorities

Enrique Teruel, Giuliana Franceschinis, *Member*, *IEEE Computer Society*, and Massimiliano De Pierro

Abstract—Generalized Stochastic Petri Nets (GSPN), with immediate transitions, are extensively used to model concurrent systems in a wide range of application domains, particularly including software and hardware aspects of computer systems, and their interactions. These models are typically used for system specification, logical and performance analysis, or automatic code generation. In order to keep modeling separate from the analysis and to gain in efficiency and robustness of the modeling process, the complete specification of the stochastic process underlying a model should be guaranteed at the net level, without requiring the generation and exploration of the state space. In this paper, we propose a net-level method that guides the modeler in the task of defining the priorities (and weights) of immediate transitions in a GSPN model, to deal with confusion and conflict problems. The application of this method ensures well-definition without reducing modeling flexibility or expressiveness.

Index Terms—Stochastic Petri nets, priorities, conflict, confusion, modeling methodology.

1 Introduction

Priorities were early introduced as an extension to Petri nets (PN) [1], [2]: Transitions can be classified into *priority classes* with an *absolute* priority level or a *relation between transitions* can be specified [3], [4]. In any case, priorities modify the enabling rule by preventing a transition firing when another transition having priority over it is also enabled. Priorities were originally introduced with the aim of enlarging the expressive power of the formalism, but they have found another important motivation in the case of *generalized stochastic PN (GSPN)* [5], [6]: Time delays of very different magnitude caused computational and numerical problems for the solution that could be avoided by assuming that "fast" transitions occurred immediately; this implies that *immediate transitions* always occur before timed transitions, that is, they have priority.

The introduction of immediate transitions was not so immediate as it seemed to be at first sight. Due to their zero delay, *conflicts* between immediate transitions could not be resolved by a race policy (as conflicts between timed transitions were). Random switches were specified by assigning weights to potentially conflicting immediate transitions, but the appropriate weight assignment became a major modeling issue since it required the knowledge of the reachable markings in order to know the conflict situations that could arise. Particularly, we often face the problem of *confusion* [7] between immediate transitions, that typically leads to an incomplete specification of the stochastic process. A common way to break confusion is introducing priorities among potentially confused immediate transitions. On the other hand, different priority levels lead to more intricate behaviors, particularly to *indirect conflicts* [8].

With the incorporation of the convenient immediate transitions and priorities, with high-level extensions (see for instance [9]), and with the availability of tools (see http://www.daimi.au.dk/PetriNets/tools/), GSPN have become an attractive formalism for performance, or performability, modeling and evaluation (see, for instance, [8], [10], [11]), with applications in diverse fields, particularly software and computer systems, typically distributed software, and multiprocessor or networked computers [5], [12], [13], [14], [15], [16], [17], but also manufacturing systems [18], phased-mission systems [19], etc.

The problem of a complete and correct specification of the stochastic process underlying a model described using different flavors of stochastic PN with immediate transitions, has been tackled in several directions:

- In [8], [6], *net-level* reasoning is used to *detect* stochastic confusion and to signal which immediate transitions might become in direct or indirect conflict. This approach has been followed extensively and is incorporated in most tools because it exhibits the two appealing features of net-level, or structural, reasoning: efficiency because it works on the net, not on the state space, and robustness, that means that, typically, the correct model is correct for "any" property that we might want to analyze, and it is still correct if we change the timing and probabilities information, and even, to some extent, the initial state. Unfortunately, it was found recently [20] that the current net-level definition of GSPN is not complete, in the sense that there is a (somehow weird) situation that it does not take into account.

- *E. Teruel is with the Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna 1, 50018 Zaragoza, Spain. E-mail: eteruel@unizar.es.*
- *G. Franceschinis and M. De Pierro are with the Dipartimento di Informatica, Universita del Piemonte Orientale, Corso Borsalino 54, 15100 Alessandria, Italy. E-mail: giuliana@mfn.unipmn.it, depierro@di.unito.it.*

Manuscript received 8 Nov. 2001; revised 23 Jan. 2003; accepted 21 May 2003.
Recommended for acceptance by G. Ciardo.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 115343.

- In [21] and [22] *state space*, reasoning is used to *detect* problems: The complete and correct specification of the model is checked during the construction of the underlying stochastic model. The main advantage of this method is that it only signals actually problematic situations that affect the properties under analysis. The main drawback is the computational cost because it requires some state space generation and analysis, which might or might not be useful for the subsequent analysis of the model. Actually, this drawback is particularly relevant when the analysis technique does not require a complete state space generation, as it happens, for instance, with simulation.
- In [20], we proposed a *net-level* restriction to the priority specification to *prevent* the problems, by using a *subclass* of nets were they cannot appear. The "subclasses approach" had not been generally advised, due to the fact that topological subclasses being confusion free (e.g., free choice nets) are very limited from a modeling point of view, namely, to model mutual exclusion in the access to shared resources [7]. Nevertheless, in [20], completely general net topologies are allowed, only the priorities are restricted so that immediate transitions not intended to be in effective conflict have *different* priorities, hence only the firing of transitions which are intended to be in conflict need a probabilistic specification. This leads to the class of *detached priorities* GSPN, which *cannot exhibit stochastic confusion or indirect conflicts*, what was considered a desired feature. The question was how to assign different absolute priorities from as few as necessary relative modeling considerations, for which we outlined a method that was the seed of our current proposal.

Our position is that the possibility of some spurious warnings due to the use of net-level reasoning is a low price to pay for the efficiency and robustness that is gained. Going one step further, we would say that no net-level warning is completely spurious: if a net shows a potential problem that happens to be irrelevant in the actual behavior, perhaps that net is not "the" good model of the system, or the particular initial marking does not show a problem that with another marking would appear. A major advantage of nets is that they provide a concise and clear representation of systems whose state space is large and complex; the state space, which might be very valuable for some analytical purposes, is intentionally hidden to the user, who is supposed to produce and/or understand the net. The separation of (net-based) modeling and (possibly state space based) analysis becomes crucial in compositional modeling when we try to reuse models, e.g., of parts, or aspects in a system to produce models of the whole system. It is a desired feature that the effort devoted to obtain correct submodels reduces the effort to ensure that the complete model is correct. In this respect, net-level preventive methods show a potential to advance step-by-step [20].

The paper is organized as follows: Section 2 introduces some preliminary notions, among which the notion of well-defined GSPN (inspired by [21]) is defined for the first time in this paper. The situations leading to non-well-defined model and different ways to solve them are illustrated in Section 3. A net-level method to specify the solution of possible conflict and confusion situations by means of priorities and/or probabilities is introduced and illustrated in Section 4. In Section 5, it is proven that the method always produces well-defined GSPN, which is the main result of the paper. The application of the method is illustrated with an example in Section 6, where some of the advantages of the net-level approach are shown. Several remarks and directions of future work in Section 7 conclude the paper.

2 Preliminaries

2.1 The GSPN Formalism

The fundamentals of PN and GSPN are well-known [2], [6]. GSPN have been successfully used for the performance modeling and analysis of systems from diverse application fields. For simplicity, *inhibitor arcs* are omitted; in this work, a GSPN system is a 7-tuple $\langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{pri}, \mathbf{w}, \mathbf{m}_0 \rangle$, where P and T are the sets of *places* and *transitions*, $\mathbf{Pre}$ and $\mathbf{Post}$ are $|P| \times |T|$-dimensional arrays representing the *pre (post) incidence functions*, $\mathbf{pri}$ is a $|T|$-dimensional array representing absolute transition *priorities* (0 for *timed* transitions, and positive for *immediate* transitions), $\mathbf{w}$ is a $|T|$-dimensional array of *weights*, that are to be interpreted as rates for (exponentially) timed transitions and as weights for immediate transitions to be used in probabilistic conflict resolution, and $\mathbf{m_0}$ is a $|P|$-dimensional array representing the *initial marking*. For *pre (post) sets*, the conventional dot notation is used, e.g., $^\bullet\tau$ is the set of input places of the transitions in $\tau \subseteq T$. Concerning arrays, subscripting with elements and/or sets is flexibly used, e.g., $\mathbf{Pre}[\pi, t]$ is a subarray of $\mathbf{Pre}$ with rows corresponding to places in $\pi \subseteq P$ and column corresponding to transition $t \in T$.

A transition t has *concession* at marking $\mathbf{m}$ when $\mathbf{m} \geq \mathbf{Pre}[P, t]$ (componentwise). The *concession degree* quantifies it: $\max\{k \in \mathbb{N}_+ \mid \mathbf{m} \geq k \cdot \mathbf{Pre}[P, t]\}$. If no higher priority transitions have concession, then it can *occur* or *fire*, and we say t is *enabled*. Its occurrence yields marking $\mathbf{m}[t\rangle = \mathbf{m} + \mathbf{Post}[P, t] - \mathbf{Pre}[P, t]$. The occurrence of a *sequence* σ of transitions enabled at $\mathbf{m}$ yielding $\mathbf{m}'$ is denoted similarly: $\mathbf{m}[\sigma\rangle\mathbf{m}'$. If σ would be enabled at $\mathbf{m}$ disregarding priorities, then we say σ has concession at $\mathbf{m}$. We denote the *multiset* of transitions occurring in a sequence σ by $\parallel \sigma \parallel$. When $\parallel \sigma \parallel = \parallel \sigma' \parallel$, σ is a *permutation* of σ'. The length of a sequence σ is denoted by $|\sigma|$.

2.2 The Current Net-Level Definition of GSPN

According to [8], [6], the definition of a GSPN model requires that the user assigns absolute priorities to all immediate transitions, although no indications are given on how to define these priorities. Once they are assigned, the computation of probability for conflict resolution among immediate transitions requires the definition of the so called *extended conflict sets (ECS)* and the definition of weights for all immediate transitions. Intuitively, ECS are (transitive closures of) subsets of potentially conflicting equal priority transitions and they are computed automatically by applying structural conflict analysis on the net with priorities. In

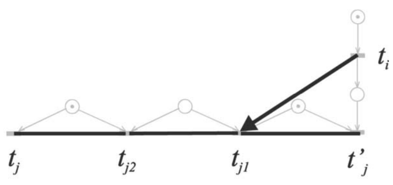


Fig. 1. A small example to illustrate the relative specification of priorities.

absence of confusion (structural methods have been defined to check this), the idea is that all ECS are independent so that their relative firing order is not relevant from the point of view of the solution. (Unfortunately, this might not be true due to pseudoconflicts [20], and this is why the current definition is not complete; ECS could be redefined to include this case, but we propose here another option.) This presumed independence allows to arbitrarily order different ECS with the same assigned priority before starting the reachability graph construction step. For instance, this can be achieved introducing new priorities, compatible with the user priority definition, so that the different ECS are always fired in a predefined order rather than trying all possible interleavings. This allows one to keep the number of vanishing markings under control.

Once ECS are computed and relatively ordered, the weights of transitions within the same ECS can be used to compute conflict resolution probabilities: These are function of the current marking and the transitions weights. Since the weights assigned to immediate transitions are not constrained to be within a certain range, to transform them into probabilities it is necessary to perform a normalization step among all enabled transitions in each ECS.

2.3 The Relative Priority Specification

An alternative way of specifying priorities of immediate transitions is by means of a binary *relation*. We write $\mathbf{pri}(t_i) > \mathbf{pri}(t_j)$ or $pri_i > pri_j$ and we draw a directed arc from t_i to t_j, when it is *specified* that t_i has priority over t_j; we write $pri_i = pri_j$ and we draw an undirected arc joining t_i and t_j when it is *specified* that they have the same priority. Therefore, the priority specification can be represented as a graph having the immediate transitions as nodes and directed or undirected arcs joining them when the corresponding relations are specified, as shown in Fig. 1, where it is specified that: $\mathbf{pri}(t_i) > \mathbf{pri}(t_{j1})$, $\mathbf{pri}(t'_j) = \mathbf{pri}(t_{j1})$, $\mathbf{pri}(t_{j1}) = \mathbf{pri}(t_{j2})$, and $\mathbf{pri}(t_{j2}) = \mathbf{pri}(t_j)$.

A priority specification *need not be complete*: For instance, the relation might specify that $\mathbf{pri}(t_i) > \mathbf{pri}(t_j)$ and $\mathbf{pri}(t_j) > \mathbf{pri}(t_k)$, but not that t_i has priority over t_k. In such case, we say that the latter relation is *deduced*, due to the transitive nature of the priority relation. Deduced relations affect the enabling just the same as specified relations, but it is important to make a distinction between them for the purpose of this work. In the example of Fig. 1, it can be deduced that t_i has priority over t_j, but this is not explicitly specified. In the graph, we draw arcs only for specified priorities; the remaining priority relations can be deduced following the graph paths. (In order to further emphasize the distinction specified/deduced, we shall denote by $\mathbf{pri}(t_i) > \mathbf{pri}(t_j)$, or similar expressions, only the specified relations, not the deduced ones.)

We denote by $[t]$ the *class* of transitions with priority (either specified or deduced) equal to that of t. In the example of Fig. 1, there are two classes: $[t_i] = \{t_i\}$ and $[t_j] = \{t'_j, t_{j1}, t_{j2}, t_j\}$.

For a given marking $\mathbf{m}$, we denote by (t), or by $(t)_\mathrm{m}$ when $\mathbf{m}$ is not obvious from context, a maximal subclass of $[t]$, containing t, whose transitions are enabled and they are connected through undirected arcs. Similarly, considering concession instead of enabling, we define $((t))$, or $((t))_\mathrm{m}$. In graph terms, a $((t))_\mathbf{m}$ is a connected component in the graph representing the priority specification after deleting the transitions without concession at $\mathbf{m}$, and the directed arcs. In the example of Fig. 1, for the depicted marking, $(t_i) = ((t_i)) = \{t_i\}$ and $((t_j)) = \{t_j\}$.

We say that the priority relation between two sets (typically subclasses) of transitions is specified when it is specified between at least two transitions, one in each set. In the example of Fig. 1, we would say that the priority relation between $\{t_i\}$ and $\{t'_j, t_{j1}, t_{j2}, t_j\}$ is specified, but not the priority relation between $\{t_j\}$ and $\{t'_j, t_{j1}\}$.

At each vanishing marking, we may have several subclasses $((t))$ with concession. Notice that it is even possible that two disjoint $((t_i)), ((t_j)) \subset [t_k]$ have concession. Among the subclasses with concession, only those with no other subclass having priority over them (specified or deduced) are enabled. Consider now the marking yielded by the occurrence of t_i in the example of Fig. 1; both $(t'_j) = \{t'_j\}$ and $(t_j) = \{t_j\}$ are enabled.

Now, that we have defined relative priorities, let us discuss how it is possible to turn a model with relative priorities into an equivalent model with absolute priorities. The motivation for this could be the availability of tools working only with absolute priorities (e.g., GreatSPN). The problem here is how to order any pair of transition classes, $[t]$ and $[t']$, which are not related in the relative priority definition. Notice that not being related is not the same as having equal priority: In the former case, *we do not assume a probabilistic firing policy*, as we shall do when the priority is equal (see below). Therefore, the firing order between nonrelated transitions should not be relevant; otherwise, the model would not be *well-defined* (see below the formalization of this concept). In case the model is well-defined, any firing order of nonrelated transitions produces the same effect; hence, nonrelated transitions can be relatively ordered in whichever way. The most convenient choice is to arbitrarily pick one class and say that it has priority over the other. This choice allows one to avoid considering useless interleavings when generating the vanishing portion of the reachability graph, hence reducing the number of irrelevant vanishing markings. This idea of introducing artificial priorities has been exploited for the generation of reduced reachability graphs in timed and untimed PN [8], [23], as commented in the previous subsection.

Finally, once priority classes are defined, the probability of firing one transition, belonging to one of the enabled subclasses, at a marking $\mathbf{m}$ is computed (normalization step) using the weights of the transitions *in its subclass*:

$$\mathrm{Prob}(t, \mathbf{m}) = \frac{\mathbf{w}(t)}{\sum_{t' \in (t)_\mathrm{m}} \mathbf{w}(t')}.$$

Notice that the weights do have a meaning *locally*, that is, within each subclass. Compared to the normalization step in the current definition of GSPN that always involves all enabled transitions within an ECS, now we can avoid the generation of some useless vanishing markings when two disjoint subclasses (t) and (t'), included in the same priority class $[t]$, are enabled: In this case, the transition weights in the two subsets can be normalized independently, and their relative firing order can be chosen arbitrarily. In the example (after firing t_i), both firings of t_j or t'_j can occur, independently or concurrently, each of them with probability one.

The probability of firing sequence σ at $\mathbf{m}$, $\mathrm{Prob}(\sigma, \mathbf{m})$, is the product of the probabilities of the successive occurrences of its transitions.

2.4 Well-Defined GSPN

Our definition of a *well-defined* model is based on the algorithmic definition in [21]. Given a model and a vanishing marking $\mathbf{m}$, the algorithm in [21] recursively builds all possible immediate firing sequences starting in $\mathbf{m}$ and reaching a tangible marking, and decides *on-the-fly* whether the model is well-defined with respect to $\mathbf{m}$: If there are several tangible markings reachable from $\mathbf{m}$ through some immediate sequences, then it is checked whether the model contains enough information to compute the probability of reaching each of those tangible markings. Moreover, if the measures to be computed from the model depend on the set of transitions fired to reach a marking (that is, if we are not only interested in the markings that can be reached but also how they are reached), then the model is well-defined when it contains enough information to compute the probability of each sequence (or set of sequences, equal up to a permutation). If all immediate sequences are finite and acyclic when all vanishing markings have been checked, the algorithm can guarantee that the model is well-defined, or warn the modeler that it is not.

We rephrase this algorithmic definition by a bijection between some sets of tuples (tangible marking reached, sequence, probability of the sequence). For a (vanishing) marking $\mathbf{m}$ where an immediate transition t is enabled, we denote by $\mathcal{P}_{\mathrm{m},t}$ the set of all tuples $(\overline{\mathbf{m}}, \sigma, \rho)$ where $\overline{\mathbf{m}}$ is tangible, the immediate sequence σ starts with t, and $\mathbf{m}[\sigma\rangle\overline{\mathbf{m}}$ occurs with probability ρ. For sets of transitions: $\mathcal{P}_{\mathrm{m},\tau} = \bigcup_{t\in\tau} \mathcal{P}_{\mathrm{m},t}$.

Definition 1. *A (vanishing) marking* $\mathbf{m}$ *is* well-defined *iff for every pair of different* (t) *and* (t') *enabled at* $\mathbf{m}$*:*

$$\mathcal{P}_{\mathrm{m},(t)} \approx \mathcal{P}_{\mathrm{m},(t')}$$

meaning that a bijection exists between the two sets, such that the correspondence between a $(\overline{\mathbf{m}}, \sigma, \rho) \in \mathcal{P}_{\mathrm{m},(t)}$ *and a* $(\overline{\mathbf{m}}', \sigma', \rho') \in \mathcal{P}_{\mathrm{m},(t')}$ *entails:*

- $\overline{\mathbf{m}} = \overline{\mathbf{m}}'$,
- $\parallel \sigma \parallel = \parallel \sigma' \parallel$,
- $\rho = \rho'$.

In such case, we define $\mathcal{P}_{\mathrm{m}} = \mathcal{P}_{\mathrm{m},(t)}$*, for an arbitrary* (t).

A GSPN is well-defined when every reachable vanishing marking is.

Let $\mathbf{m}[t\rangle\mathbf{m}'$; the sets $\mathcal{P}_{\mathrm{m},t}$ can be defined constructively as follows:

```
if $\mathbf{m}'$ is tangible
    then $\mathcal{P}_{\mathrm{m},t} = \{(\mathbf{m}', t, \mathrm{Prob}(t, \mathbf{m}))\}$
    else $\mathcal{P}_{\mathrm{m},t} = \bigcup_{(\overline{\mathbf{m}},\sigma,\rho)\in\mathcal{P}_{\mathrm{m}'}} \{(\overline{\mathbf{m}}, t\sigma, \mathrm{Prob}(t, \mathbf{m}) \cdot \rho)\}$
```

Notice that $\mathcal{P}_{\mathrm{m},t}$ can only be (recursively) computed when all successor markings of $\mathbf{m}[t\rangle$ are well-defined.

Later on, in Sections 3 and 4, several examples of well-defined and non-well-defined models will be shown.

For the interested reader, compared to the algorithmic approach of [21], we abstract from the details that do not apply to GSPN (the definition in [21] applies to a much larger class of timed PN), and we strengthen some requirements, necessary to fit a structure based approach as the one presented in Section 4. While in [21], the *weight classes* are sets of transitions that the modeler *assumes* might be concurrently enabled and whose relative firing order may affect the system behavior, here they are *deduced* from the priority specification and the net structure; while the algorithm in [21] checks whether such assumptions of the modeler are correct, taking into account the particular performance measures of interest and the initial marking, here we prove that following the structural method in Section 4 to specify priorities the model is well-defined for any performance measures (based on probabilities of markings and/or firing count vectors), and this is why our definition must consider every sequence and marking.

2.5 The Conflict and Causality Structure of a Net

Transition t_i is said to be in *effective conflict relation* with t_j at marking $\mathbf{m}$ (where they are both enabled), denoted $\mathrm{Conf}(t_i, t_j, \mathbf{m})$, when $\mathbf{m}[t_i\rangle\mathbf{m}'$ and the concession degree of t_j decreases from $\mathbf{m}$ to $\mathbf{m}'$. The notion of (effective) conflict is dynamical, it depends on the marking. The static structure of the net contains information on the *potential* dynamic conflicts. The very basic net construct used to model conflicts is a place with more than one output transition. Transition t_i is said to be in *structural conflict relation* with t_j, denoted $\mathrm{SConf}(t_i, t_j)$, when ${}^\bullet t_i \cap {}^\bullet t_j \neq \emptyset$.

Transition t_i is said to be *causally connected* to t_j at marking $\mathbf{m}$ (where t_i is enabled) when $\mathbf{m}[t_i\rangle\mathbf{m}'$ and the concession degree of t_j increases from $\mathbf{m}$ to $\mathbf{m}'$. The very basic net construct used to model causality is a place connecting two transitions. Transition t_i is said to be *structurally causally connected* to t_j when $t_i{}^\bullet \cap {}^\bullet t_j \neq \emptyset$. For our purpose, causal connection from t_i to t_j at marking $\mathbf{m}$ is particularly relevant in the case that a third transition, t_k, has concession at $\mathbf{m}$, together with t_i, while t_j has not concession at $\mathbf{m}$, denoted $\mathrm{Caus}_{t_k}(t_i, t_j, \mathbf{m})$. This is because this is the situation that originates the confusion and indirect conflict problems (see Section 3), when the firing of t_i produces the enabling of t_j, which (ultimately) disturbs the previous enabling of t_k. The structural version of this connection holds iff a marking exists where t_k has concession and t_j has not, and t_j gets concession by the occurrence of t_i. Formally:

Definition 2. *A transition* t_i *is structurally causally connected to* t_j *conditioned to* t_k*, denoted* $\mathrm{SCaus}_{t_k}(t_i, t_j)$ *iff* $t_i{}^\bullet \cap {}^\bullet t_j \neq \emptyset$ *and not* $\mathbf{Pre}[t_i{}^\bullet \cap {}^\bullet t_j, t_j] \leq \mathbf{Pre}[t_i{}^\bullet \cap {}^\bullet t_j, t_k]$ *(componentwise).*

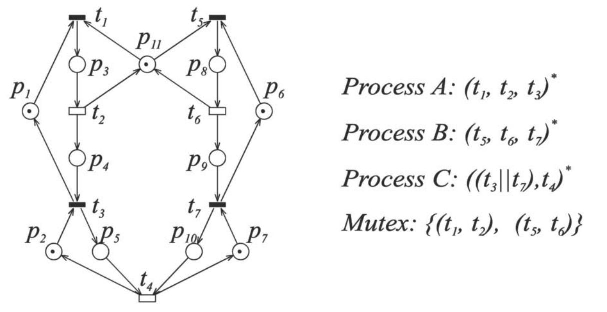

t_1
t_5
p_{11}
p_3
p_8
p_1
p_6
t_2
t_6
p_4
p_9
t_3
t_7
p_2
p_5
p_{10}
p_7
t_4
Process A: $(t_1, t_2, t_3)^*$
Process B: $(t_5, t_6, t_7)^*$
Process C: $((t_3||t_7), t_4)^*$
Mutex: $\{(t_1, t_2), (t_5, t_6)\}$

Fig. 2. A GSPN model to illustrate conflict and confusion situations.

For convenience, in running text, we often say t_i is *upstream* t_j (conditioned to t_k).

The relations above are intended for nets without priorities. Priorities may break some of the conflict relations or conditioned causality connections, by removing the enabling of one of the transitions involved. Therefore, in a net with priorities, we don't usually say there is a conflict between t_i and t_j when either t_i has priority over t_j or viceversa (e.g., a timed transition is never said to be in conflict with an immediate one, even if they share some input place). Similarly, $\mathrm{Caus}_{t_k}(t_i, t_j)$ is only meaningful when neither t_i has priority over t_k nor viceversa.

3 Conflicts and Confusion

Let us illustrate, first of all, the notions of direct and indirect conflicts, and confusion, the reason why, when these situations arise, there is a need to specify the firing policy to obtain a well-defined model, and some ways of specifying this policy. We use as running example the net shown in Fig. 2. It models a concurrent program consisting of three cyclic processes (A, B, C) that are synchronized as follows:

- The actions "t_3" in A and C (respectively the actions "t_7" in B and C) represent a synchronous (or rendezvous) communication between both processes.
- The critical sections of A and B, (t_1, t_2) and (t_5, t_6), respectively, represent the access to a shared resource, and must be executed in mutual exclusion.

Each process contains an action (t_2, t_6, t_4, respectively) that takes some time to complete (a computation, or an access to a database), while the communication between processes or the arbitration of accesses is assumed to be instantaneous.

3.1 Example of Direct Conflict

Assume first that no priorities are specified at all (other than that of immediate transitions over timed ones), so $[t] = \{t\}$ for all (immediate) t. At the (vanishing) initial marking, a conflict is effective: Both t_1 and t_5 are enabled and, depending on which one fires, two different tangible markings are reached: $\mathbf{m}_1 = p_2 + p_3 + p_6 + p_7$ and $\mathbf{m}_2 = p_1 + p_2 + p_7 + p_8$, respectively. The conflict is neither solved by priorities nor by probabilities (we assumed that only conflicts between transitions with *equal* priority can be solved by probabilities), so the model is, obviously, not well-defined.

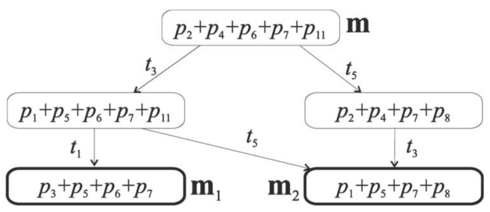


Fig. 3. Part of the reachability graph of the GSPN in Fig. 2. Tangible markings have a thick border.

Two ways to specify the solution of the conflict, either probabilistic or by priorities, are possible, depending on the actual behavior of the system:

- If $pri_1 = pri_5$, the conflict is solved by probabilities, after assigning suitable w_1 and w_5, and now $\mathcal{P}_{m_0} = \left\{ \left(\mathbf{m}_1, t_1, \frac{w_1}{w_1+w_5}\right), \left(\mathbf{m}_2, t_5, \frac{w_5}{w_1+w_5}\right) \right\}$.
- If, say, $pri_1 > pri_5$ the conflict disappears, only t_1 is enabled and then $\mathcal{P}_{m_0} = \{(m_1, t_1, 1)\}$.

In any case, from m_0 to the next tangible markings the behavior is well-defined. But, other problems can arise later on, depending on how the solution of the conflict was specified. In fact, it is worth noticing that, despite the fact that, in this example, the initial conflictive marking is (vanishing and) transient, the decision to assign equal or different priorities to t_1 and t_5 *does affect* the behavior later on (see below), so the matter is not at all irrelevant.

3.2 Example of Confusion

Assume we decided $pri_1 = pri_5$ and consider the (vanishing) marking $\mathbf{m}$ (see Fig. 3), where $\mathrm{Caus}_{t_5}(t_3, t_1, \mathbf{m})$ and, naturally, $\mathrm{SCaus}_{t_5}(t_3, t_1)$. From $\mathbf{m}$, two tangible markings can be reached after some immediate firings. The marking $\mathbf{m}_2$ can be reached by the firing of either t_5 followed by t_3, without effective conflicts, or t_3 followed by t_5 winning its conflict with t_1; so, depending on which sequentialization of t_3 and t_5 is taken, the probability to reach $\mathbf{m}_2$ varies, what in GSPN terminology is called *stochastic confusion*. Moreover, if t_5 is fired first, before becoming in conflict with t_1, the marking $\mathbf{m}_1$ cannot be reached, while it is reachable after the firing of t_3, if t_1 wins the subsequent conflict. In summary, $\mathcal{P}_{\mathrm{m},(t_3)} \not\approx \mathcal{P}_{\mathrm{m},(t_5)}$, so the model is not well-defined. (The same problem arises with the other upstream transition, t_7, at a corresponding marking.)

Two ways to specify the solution of the confusion, either probabilistic or by priorities, are possible, depending on the actual behavior of the system:

- If $\{t_1, t_3, t_5, t_7\}$ have equal priority, so they form a class, the confusion is solved by probabilities and now

$$\mathcal{P}_{\mathrm{m}} = \left\{ \left(\mathbf{m}_1, t_3 t_1, \frac{w_3}{w_3 + w_5} \cdot \frac{w_1}{w_1 + w_5}\right), \right.$$
$$\left. \left(\mathbf{m}_2, t_3 t_5, \frac{w_3}{w_3 + w_5} \cdot \frac{w_5}{w_1 + w_5}\right), \left(\mathbf{m}_2, t_5 t_3, \frac{w_5}{w_3 + w_5}\right) \right\}.$$

- If, say, $pri_3 > pri_5$ (and correspondingly $pri_7 > pri_1$), the confusion disappears, only t_3 is enabled at $\mathbf{m}$ and then

$$\mathcal{P}_{\mathrm{m}} = \left\{ \left(\mathbf{m}_1, t_3 t_1, \frac{w_1}{w_1 + w_5} \right), \left(\mathbf{m}_2, t_3 t_5, \frac{w_5}{w_1 + w_5} \right) \right\}.$$

The current practice in GSPN modeling is breaking confusion by use of priorities (the net-level GSPN definition requires absence of stochastic confusion and the definition of dpGSPN [20] enforces absence of confusion). This is a sensible option because it is likely that the confused transitions, say t_3 and t_5 in the example, are not intended to be in conflict, so confusion is signaled as a "modeling mistake." But, all in all, it is a question of representing the actual system, so, if the modeler requires a probabilistic solution of the confusion situation, this should be allowed, perhaps after a warning.

3.3 Example of Indirect Conflict

Assume now that we decided $pri_1 > pri_5$. (In Fig. 3, the arc from $p_1 + p_5 + p_6 + p_7 + p_{11}$ to $\mathbf{m}_2$ disappears, everything else remains unchanged.) In such case, at $\mathbf{m}$, if t_3 is fired, then t_1 becomes enabled and, having priority over t_5, it fires removing concession from t_5. In GSPN terminology, it is said that t_3 is in *indirect conflict* relation with t_5: From the point of view of t_5, it is like t_3 was in conflict with it because, after the choice of firing t_3, the token in p_{11} is removed without a chance for t_5 to grab it.

Formally, we find that the model is not well-defined because $\mathcal{P}_{\mathrm{m},(t_3)} \not\approx \mathcal{P}_{\mathrm{m},(t_5)}$. Therefore, the modeler should specify a conflict resolution policy between t_3 and t_5. Two ways to do so, either probabilistic or by priorities, are possible, depending on the actual behavior of the system:

- If $pri_3 = pri_5$, they form a class, the indirect conflict is solved by probabilities, and now

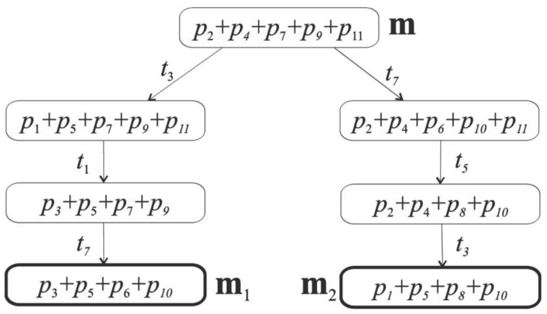
$$\mathcal{P}_{\mathrm{m}} = \left\{ \left(\mathbf{m}_1, t_3 t_1, \frac{w_3}{w_3 + w_5} \right), \left(\mathbf{m}_2, t_5 t_3, \frac{w_5}{w_3 + w_5} \right) \right\}.$$

- If, say, $pri_3 > pri_5$, the indirect conflict disappears, only t_3 is enabled at $\mathbf{m}$, and then $\mathcal{P}_{\mathrm{m}} = \{(\mathbf{m}_1, t_3 t_1, 1)\}$.

The same as we would recommend breaking confusion with priorities in case it fits the modeling needs, we would also recommend breaking indirect conflicts with priorities and modeling actual conflicts of the system with direct conflicts of the net.

3.4 Pseudoconflict: A Weird Sort of Indirect Conflict

Indirect conflicts such as the above one are taken into account in the definition of *extended conflict sets (ECS)* in GSPN (see [8], [6]). But, there is another (weird) sort of indirect conflict that was not taken into account and which was first signaled in [20]. Assume that $pri_1 = pri_5$ and, in order to break confusion, we assigned $pri_3 < pri_5$ and $pri_7 < pri_1$. Although t_3 and t_7 appear to be independent (in GSPN terminology, they are in different ECS), it can be easily checked that their firings are not at all independent. Consider the (vanishing) marking $\mathbf{m}$ (see Fig. 4), from which two tangible markings can be reached after some immediate firings. Due to the higher priority of t_1 and t_5, transitions upstream, t_3 and t_7, happen to be in a sort of indirect conflict relation: Depending on which one fires first, a different higher priority transition is fired, leading to different tangible markings. This situation was called *pseudoconflict* in [20] because, despite the firing of one transition not reducing the concession degree of the other, the final effect is very much like they were in conflict, as the reachability graph clearly shows.



$$\begin{tikzcd}
 & \boxed{p_2+p_4+p_7+p_9+p_{11}}\ \mathbf{m} \arrow[dl, "t_3"'] \arrow[dr, "t_7"] & \\
\boxed{p_1+p_5+p_7+p_9+p_{11}} \arrow[d, "t_1"] & & \boxed{p_2+p_4+p_6+p_{10}+p_{11}} \arrow[d, "t_5"] \\
\boxed{p_3+p_5+p_7+p_9} \arrow[d, "t_7"] & & \boxed{p_2+p_4+p_8+p_{10}} \arrow[d, "t_3"] \\
\boxed{p_3+p_5+p_6+p_{10}}\ \mathbf{m}_1 & & \mathbf{m}_2\ \boxed{p_1+p_5+p_8+p_{10}}
\end{tikzcd}$$

Fig. 4. Part of the reachability graph of the GSPN in Fig. 2.

Formally, again the model is shown to be non-well-defined because $\mathcal{P}_{\mathrm{m},(t_3)} \not\approx \mathcal{P}_{\mathrm{m},(t_7)}$. Therefore, the modeler should either solve the pseudoconflict by probabilities, after assigning $pri_3 = pri_7$, or by priorities, e.g., $pri_3 > pri_7$.

Observe that no pseudoconflict would appear if the priorities of t_3 and t_7 were higher than the priorities of both t_1 and t_5 because they both would fire before the conflict downstream was solved, or said in other words, they would "push" tokens towards the conflict. The same as we would recommend breaking confusion and indirect conflicts with priorities, in case it fits the modeling needs, we would recommend also the "pushing" orientation, to avoid pseudoconflicts.

4 A Method to Assign Priorities

All situations above share a common pattern: Two transitions which are not intended to interfere, not in priority relation (hence, their relative firing policy is not specified), happen to become enabled at the same time, and the firing order *is* relevant. More precisely, in all cases, we find some structurally conflicting immediate transitions and either one or both are preceded by different immediate transitions upstream. Moreover, the problems can propagate upstream, depending on how we solve them (see the example in Fig. 6 later on).

Here, we propose a method to specify (relative) priorities that guarantees that the model is well-defined. The basic method is Algorithm 3, which works as follows: While there are required priorities, the modeler chooses the orientation for one of them; depending on the choice taken, it might be required to specify also the priorities between transitions upstream, either to break a possible confusion or a possible indirect conflict, except in the case that such specifications had already been introduced (they are in **pri**). At the beginning, it is only required to specify the priority of transitions in structural conflict, to solve direct conflicts.

Algorithm 3 (Method to specify priorities)

```
Input - $\langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{pri}, \mathbf{w}, \mathbf{m_0} \rangle$
    % pri indicates only the timed transitions
    % w of timed transitions are defined
```

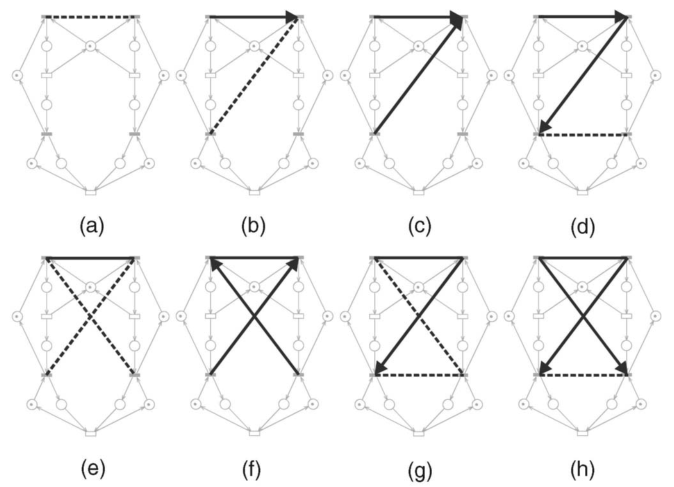


Fig. 5. Several intermediate and final steps in the application of the method to the net in Fig. 2. Dashed lines indicate required orderings. Undirected arcs mean equal priority, while a directed arc from t to t' means that t has priority over t'.

```
Output - pri (and w) for immediate transitions

$\mathrm{Req} = \{(t_i, t_j) \mid \mathrm{SConf}(t_i, t_j)\}$ % the required priorities,
    % initially all pairs of transitions in structural conflict
$\mathrm{ToProp} = \emptyset$ % the specifications that originate propagation
    % upstream, initially empty

while $\mathrm{Req} \neq \emptyset$ do
    User_modeling: The user specifies for a $(t_i, t_j) \in \mathrm{Req}$
        $pri_i > pri_j$, $pri_i < pri_j$, or $pri_i = pri_j$
        not contradicting previous specifications;
        move $(t_i, t_j)$ from Req to ToProp

    Propagation: foreach $(t_i, t_j) \in \mathrm{ToProp}$ do
        if $pri_i \geq pri_j$ then
            $\mathrm{Req} = \mathrm{Req} \cup \{(t_k, t_j) \mid \mathrm{SCaus}_{t_j}(t_k, t_i)\} - \mathbf{pri}$
        if $pri_i \leq pri_j$ then
            $\mathrm{Req} = \mathrm{Req} \cup \{(t_k, t_i) \mid \mathrm{SCaus}_{t_i}(t_k, t_j)\} - \mathbf{pri}$
        remove $(t_i, t_j)$ from ToProp

(foreach $[t] \neq \{t\}$ do Require_weights_assignment)
```

In Section 5, we prove that this method always produces a well-defined model. Notice that, since the method works at the net level, without even considering the initial marking, well-definition is guaranteed for *every* initial marking.

Concerning modeling flexibility, it can be observed in the algorithm that it is allowed to specify probabilistically the solution of indirect conflicts, and even stochastic confusion, differently from current GSPN practice. We insist once more that we would not recommend this in general because we believe that it is good modeling practice to reflect all conflicts or decisions in the system by structural conflicts of the net, but we admit that some particular modeling case might require it. If, following this recommendation, it is only specified that $\mathbf{pri}(t_i) = \mathbf{pri}(t_j)$ in cases where $\mathrm{SConf}(t_i, t_j)$, then the obtained net is a dpGSPN [20]. We recommend also to take the "pushing" orientation (the transition upstream has priority over the transition downstream) whenever possible, to reduce the number of required specifications.

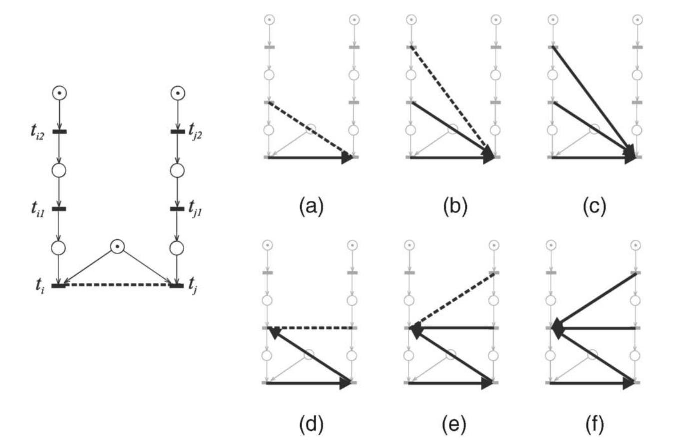


Fig. 6. An example showing the need of the iterative propagation upstream.

Only the essentials of the method are shown in Algorithm 3, although, of course, any practical user-friendly implementation of this method in a tool should also provide, together with a graphic interface:

- Automatic verification of coherence. If a user specification at some point violates the transitive nature of the priority relation (what can be detected as a cycle in the priority graph), the user should be informed, showing up the cycle. Alternatively, only the "legal" options could be allowed. In particular, if a required specification can be deduced from previous ones (see a couple of cases along the explanation of the example in Fig. 5), then the tool could inform the user and merely wait for confirmation.
- Undo capability. Assuming that the user might make a mistake and later realize it when another specification shows up an incoherence, a "design history" should be kept during the process to allow undoing a wrong specification together with all the others motivated by it.
- (Optional) Recommendations. To use the "pushing" orientation, or not to solve stochastic confusion by probabilities.

Let us show now how the method works on the example of Section 3, see Fig. 5. Only a few cases are shown. Initially, the specification of (t_1, t_5) is required (Fig. 5a). If $pri_1 > pri_5$, then (t_3, t_5) is required (Fig. 5b) to break the indirect conflict. If we choose the "pushing" orientation, $pri_3 > pri_5$ (Fig. 5c), we are done. If instead we take the "nonpushing" orientation, $pri_3 < pri_5$, a pseudoconflict might appear between t_3 and t_7, so (t_3, t_7) is required (Fig. 5d) and, after deciding a specification for it, we finish. Now, consider the case that we initially choose $pri_1 = pri_5$. The specification of both (t_3, t_5) and (t_1, t_7) is required (Fig. 5e) to break the confusion. If we take the "pushing" orientation for both (Fig. 5f), we are done, but if we decide, e.g., $pri_3 < pri_5$, the specification of (t_3, t_7) is required to break a potential pseudoconflict (Fig. 5g). If we decide $pri_1 < pri_7$, then necessarily $pri_3 < pri_7$, but if we take instead the "nonpushing" orientation, i.e., $pri_1 > pri_7$ it is still required to specify (t_3, t_7) (Fig. 5h). As another case, if in Fig. 5e, we decide $pri_1 = pri_7$ and $pri_3 = pri_5$, then necessarily $pri_3 = pri_7$.

The example in Fig. 6 illustrates the necessity of the propagation and the diversity of situations that we might reach depending on how the required priorities are specified. Initially, it is required to specify (t_i, t_j). We decide $pri_i > pri_j$ and then (t_{i1}, t_j) is required to break a potential indirect conflict (Fig. 6a). If we take the "pushing" orientation, $pri_{i1} > pri_j$, (t_{i2}, t_j) is required to break another potential indirect conflict (Fig. 6b). If we take again the "pushing" orientation, $pri_{i2} > pri_j$, we are done: We need not specify the priority of t_{j1} and t_{j2} because no matter that they occur before or after t_{i2} and t_{i1}, the token they give shall wait in front of t_j until these have occurred.

Assume now that, after Fig. 6a, we decide to take the "nonpushing" orientation, $pri_{i1} < pri_j$. Now, it is required to specify (t_{i1}, t_{j1}) to break a potential pseudoconflict (Fig. 6d). We can decide, for instance, that $pri_{i1} < pri_{j1}$, and then (t_{i1}, t_{j2}) is required (Fig. 6e). Finally, deciding $pri_{i2} < pri_{j1}$, for instance, leads to a well-defined model (Fig. 6f).

One interesting thing to observe in this example is that, although different, and among many other possibilities that the reader might want to try, *both* models in Figs. 6c and 6f are well-defined (so the method led to correct models in a flexible way), but *none* of the others is (so the propagation was necessary). It is also worthwhile commenting on the case of Fig. 6f, where we started not taking the "pushing" orientation, so we had to solve a pseudoconflict between t_{i1} and t_{j1}. We did in this case by priorities: The solution of the pseudoconflict upstream *overrides* the policy given for the actual conflict; actually, in this case, the token in the right side always "wins," although, in the actual conflict, it is the left side that has the priority! (It would have been the same if the actual conflict was solved by probabilities or by the reverse priorities because the fact is that this conflict *never* becomes effective.)

5 THE METHOD PRODUCES WELL-DEFINED MODELS

In order to prove that application of Algorithm 3 always produces a well-defined model, we first show a basic property of this algorithm. This property states how the firing of a transition in one subclass with concession can modify the concession of other subclasses.

Property 4. *Let* $\mathbf{m}$ *be a vanishing marking where* $((t_i))_{\mathrm{m}} \neq \emptyset$ *and* $((t_j))_{\mathrm{m}} \neq \emptyset$. *Let* $\mathbf{m}' = \mathbf{m} + \mathbf{Post}[P, t'_i] - \mathbf{Pre}[P, t'_i]$ *for an arbitrary* $t'_i \in ((t_i))_{\mathrm{m}}$ *(i.e.,* $\mathbf{m}' = \mathbf{m}[t'_i\rangle$ *if* t'_i *was enabled).*

If Algorithm 3 did not require to specify a priority between $((t_i))_{\mathrm{m}}$ *and* $((t_j))_{\mathrm{m}}$*, then* $((t_j))_{\mathrm{m}'} = ((t_j))_{\mathrm{m}}$.

Moreover, if $((t_h))_{\mathrm{m}} = \emptyset$ *but* $((t_h))_{\mathrm{m}'} \neq \emptyset$ *and* t_h *has priority over* t_j*, then Algorithm 3 did not require to specify a priority between* $((t_h))_{\mathrm{m}'}$ *and* $((t_j))_{\mathrm{m}}$.

Proof. No $t'_j \in ((t_j))_{\mathrm{m}}$ loses concession by the firing of t'_i because this implies $\mathrm{SConf}(t'_i, t'_j)$, hence the specification of (t'_i, t'_j) was required, contradiction.

No $t''_j \in ((t_j))_{\mathrm{m}'} - ((t_j))_{\mathrm{m}}$ exists. This would mean that $\mathrm{Caus}_{t'_j}(t'_i, t''_j)$, so $\mathrm{SCaus}_{t'_j}(t'_i, t''_j)$, for every $t'_j \in ((t_j))_{\mathrm{m}}$ particularly for the one such that (t'_j, t''_j) was specified (as equal priority). But, then the propagation phase of Algorithm 3 would have required the specification of (t'_i, t'_j), contradiction.

Finally, assume $t'_h \in ((t_h))_{\mathrm{m}'}$ and Algorithm 3 required the specification of t'_h and $t'_j \in ((t_j))_{\mathrm{m}}$ (and $pri(t'_h) > pri(t'_j)$ was assigned). Since $\mathrm{Caus}_{t'_j}(t'_i, t'_h)$, it follows that $\mathrm{SCaus}_{t'_j}(t'_i, t'_h)$, hence Algorithm 3 would have required the specification of (t'_i, t'_j), contradiction. □

It is worth noticing that we did not require t'_i to be actually enabled at $\mathbf{m}$, only that it had concession, which is more general, and this is what we are going to use in the proof of Theorem 5, the main result of the paper, which states that the application of the method guarantees well-definition.

Theorem 5. *If* $\langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{pri}, \mathbf{w}, \mathbf{m}_0\rangle$ *is a GSPN where* $\mathbf{pri}$ *(and* $\mathbf{w}$*) was obtained by application of Algorithm 3, then it is well-defined.*

Proof. For any vanishing marking $\mathbf{m}$ where two different $(t_i) \neq \emptyset$ and $(t_j) \neq \emptyset$ are enabled, we prove that $\mathbf{m}$ is well-defined, that is, for arbitrary $t'_i \in (t_i)$ and $t'_j \in (t_j)$, we can build two sequences, starting with t'_i or t'_j respectively, which are a permutation of each other, with the same probability.

Actually, we prove a slightly more general result, namely, we prove it for $((t_i)) \neq \emptyset$ and $((t_j)) \neq \emptyset$ with *concession* at $\mathbf{m}$, with no *specified* priority between them, and such that no transition in a $[t_h]$ with *specified* higher priority over at least one of them has concession.

In what follows we denote by $\mathbf{m}(t\rangle\mathbf{m}'$ the occurrence of a transition t having concession at $\mathbf{m}$ when no transition in a $[t']$ with *specified* priority over $((t))$ has concession. So to say, $\mathbf{m}(t\rangle\mathbf{m}'$ is a "relaxed version" of enabling and occurrence, where only specified priorities are taken into account, and not deduced ones. A sequence σ of occurrences of this kind is denoted similarly: $\mathbf{m}(\sigma\rangle\mathbf{m}'$. Abusing notation, we denote by $\mathrm{Prob}(\sigma, \mathbf{m})$ the probability of firing the sequence with respect to this "relaxed version" of enabling.

We fix an arbitrary tangible marking $\overline{\mathbf{m}}$ and we define the "distance" from a vanishing marking $\mathbf{m}$ to $\overline{\mathbf{m}}$ as the length of the longest sequence from $\mathbf{m}$ to $\overline{\mathbf{m}}$ using the "relaxed version" of enabling, that is, $\mathrm{Distance}(\mathbf{m}, \overline{\mathbf{m}}) = \max\{|\sigma| \mid \mathbf{m}(\sigma\rangle\overline{\mathbf{m}}\}$. (Notice that all, but not only, the sequences that are *actually enabled*—taking also *deduced* priorities into account—are included.) We prove the result, for a fixed $\overline{\mathbf{m}}$, by induction on this distance, λ. Once this is proven for whichever tangible marking ($\overline{\mathbf{m}}$ arbitrary), performing the union on the different reachable $\overline{\mathbf{m}}$ yields the result.

The property to prove is $\mathrm{Property}(\mathbf{m}, \overline{\mathbf{m}})$: If $((t_i)) \neq \emptyset$ and $((t_j)) \neq \emptyset$ have *concession* at $\mathbf{m}$, the priority between them is not specified and there is no transition with concession in a $[t_h]$ with *specified* higher priority over at

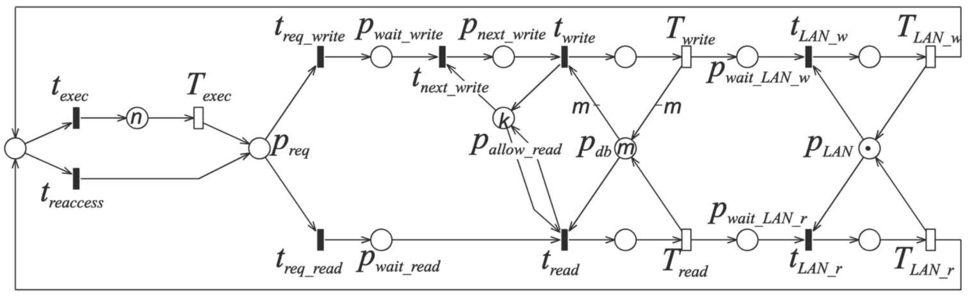


Fig. 7. A GSPN model of a readers and writers system.

least one of them, then, if $\mathbf{m}(t'_i\sigma_i\rangle\overline{\mathbf{m}}$ for some $t'_i \in ((t_i))$, there exist a $t'_j \in ((t_j))$ and a σ_j such that $\mathbf{m}(t'_j\sigma_j\rangle\overline{\mathbf{m}}$, $\| t'_i\sigma_i \| = \| t'_j\sigma_j \|$, and $\mathrm{Prob}(t'_i\sigma_i, \mathbf{m}) = \mathrm{Prob}(t'_j\sigma_j, \mathbf{m})$.

The property holds trivially in the base case $\lambda = 1$ because, if $\mathbf{m}(t'_i\rangle\overline{\mathbf{m}}$, since $\overline{\mathbf{m}}$ is *tangible*, clearly $((t_j))$ has lost concession by the firing of t'_i and then $\mathrm{SConf}(t'_i, t_j)$, so they are required to be specified by Algorithm 3 to solve the conflict.

For the step of the induction, consider $\mathrm{Distance}(\mathbf{m}, \overline{\mathbf{m}}) = \lambda$. Let $\mathbf{m}(\sigma\rangle\overline{\mathbf{m}}$, with $|\sigma| = \lambda$, and starting with a $t'_i \in ((t_i))$, i.e., $\sigma = t'_i\sigma_i$. Let $\mathbf{m}' = \mathbf{m}(t_i\rangle$. Clearly, $\mathrm{Prob}(\sigma, \mathbf{m}) = \mathrm{Prob}(t'_i, \mathbf{m}) \cdot \mathrm{Prob}(\sigma_i, \mathbf{m}')$. Due to Property 4, we know that the same $((t_j))$ with concession at $\mathbf{m}$ still has concession at $\mathbf{m}'$; moreover, some other class $((t_k))$ may have concession at $\mathbf{m}'$, but surely not with specified higher priority over $((t_j))$. The first transition in σ_i might be a $t'_j \in ((t_j))$ or instead a $t'_k \in ((t_k))$. In any case, by the induction hypothesis (notice that $\mathrm{Distance}(\mathbf{m}', \overline{\mathbf{m}}) < \lambda$, otherwise $\mathrm{Distance}(\mathbf{m}, \overline{\mathbf{m}}) > \lambda$; actually, $\mathrm{Distance}(\mathbf{m}', \overline{\mathbf{m}}) = \lambda - 1$ because $|\sigma_i| = \lambda - 1$), there surely exists a sequence starting with a transition $t'_j \in ((t_j))$, $t'_j\sigma_x$, such that $\| t'_j\sigma_x \| = \| \sigma_i \|$ and $\mathrm{Prob}(t'_j\sigma_x, \mathbf{m}') = \mathrm{Prob}(\sigma_i, \mathbf{m}')$. Therefore, we have $\mathbf{m}(t'_it'_j\sigma_x\rangle\overline{\mathbf{m}}$. Let $\mathbf{m}'' = \mathbf{m}(t'_it'_j\rangle$. Clearly, $\mathbf{m}(t'_jt'_i\rangle\mathbf{m}''$ too, hence $\mathbf{m}(t'_jt'_i\sigma_x\rangle\overline{\mathbf{m}}$, with $\| t'_jt'_i\sigma_x \| = \| \sigma \|$, and $\mathrm{Prob}(t'_jt'_i\sigma_x, \mathbf{m}) = \mathrm{Prob}(\sigma, \mathbf{m})$, due to the preservation of the classes with concession, so we are done. □

Observe that once we have proven $\mathrm{Property}(\mathbf{m}, \overline{\mathbf{m}})$, showing that for any $((t_i))$ and $((t_j))$ with concession at $\mathbf{m}$ there exist $t'_i\sigma_i$ and $t'_j\sigma_j$ with concession that both reach $\overline{\mathbf{m}}$ with the same firing count vector and probability, it is easy to see that this property is true in particular for those (t_i) and (t_j) that are enabled at $\mathbf{m}$. It is then possible to apply again the same reasoning to the (shorter) sequence σ_i (or σ_j). Hence, proceeding step by step and choosing each time one of the enabled sets for the first firing in the sequence, we can incrementally build the sought enabled firing sequences.

6 An Application Example

Typically, the situations that are found in real application cases, where immediate transition subnets use to be small parts in a large model, do not illustrate the subtleties of immediate transitions better than the small examples shown in the paper do. Nevertheless, the method is intended to be applied *whenever* GSPN (with immediate transitions) are used to model systems from *whichever* application domain.

In this section, we present an example to illustrate the application of the method and also to summarize its main features. The net in Fig. 7 models the behavior of a version of the well-known readers and writers system, adapted from [6, Fig. 57], where n processes may access a common database for either reading or writing. Up to $m \leq n$ readers may access the database concurrently; instead, a writer requires exclusive access to the resource. A process can either execute in its local memory for a while before requesting a new access or immediately proceed to the next access. A local area network (LAN) transfer phase follows each read or write access to the shared memory. In order to guarantee the access to writers, whenever $k < n$ or more processes are waiting to write no new readers are granted access. (Typically, $k = 1$, so whenever some processes are waiting to write they get the resource, one after the other, immediately after all current readers finish.)

Something that a software engineer might want to analyze is, for instance, the performance effects of design decisions such as the number of processes allowed in the system (to avoid bottlenecks), the number of concurrent database accesses (perhaps related with the availability of licenses), or the number of waiting writers that block readers (in order to get a good compromise between the readers and writers "satisfaction"). Therefore, typically, we want to perform several analysis or simulations with the same net but different initial markings and of course we need to be sure that the model is well-defined in all cases.

In the model shown in Fig. 7 all activities that take a significant amount of time to complete (private executions, database accesses, and LAN accesses) are modeled as paths $t_{act} \rightarrow p_{act} \rightarrow T_{act}$, where the subindex "act" refers to the corresponding activity:

- Immediate transition t_{act}. Its occurrence represents the start of an activity of the corresponding kind.
- Place p_{act} (the label is not shown in the figure). Its marking represents the number of activities of the same kind progressing in parallel.
- Timed transition T_{act}. Its occurrence represents the completion of an activity; the transition rate parameter, $\mathbf{w}(T_{act})$, is the inverse of the average time; infinite server semantics is used since all activities proceed in parallel.

The immediate transitions, including the ones that represent activity initiations, allow to model the routing

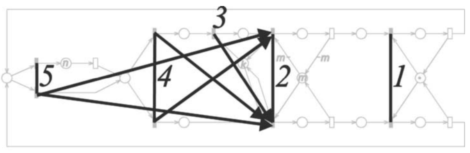


Fig. 8. A relative (and absolute) priority specification for the readers and writers system.

aspects of the system. Depending on the intended behavior, the routing decisions will either be modeled probabilistically, computed from the conflicting transition weights, or by priority relations. This must be established.

Following the method introduced in the paper, initially it is required to specify the following relations, corresponding to five structural conflicts in the net:

$$(t_{exec}, t_{reaccess}), (t_{req\_write}, t_{req\_read}), (t_{next\_write}, t_{read}),$$
$$(t_{write}, t_{read}), \text{ and } (t_{LAN\_w}, t_{LAN\_r}).$$

A reasonable specification (of course not the only one, it all depends on the intended behavior of the system) proceeds as follows:

1. On the average, each process needs to reaccess without first executing in local memory a given percentage of times, so probabilities will decide the free choice between t_{exec} and $t_{reaccess}$: $\mathbf{pri}(t_{exec}) = \mathbf{pri}(t_{reaccess})$. Similarly, $\mathbf{pri}(t_{req\_write}) = \mathbf{pri}(t_{req\_read})$.
2. Although it is not a free choice, we also take $\mathbf{pri}(t_{LAN\_w}) = \mathbf{pri}(t_{LAN\_r})$ because the LAN is allocated to any waiting process with equal probability whether it just read or wrote. (Nevertheless, the average time to complete the LAN access does depend on the kind of access, and this is why we could not model LAN accesses by a mere timed transition with single server semantics.)
3. In order to satisfy the specification that a number of writers waiting block readers, it is clear that we should finish counting waiting writers before granting access to new readers, so $\mathbf{pri}(t_{next\_write}) > \mathbf{pri}(t_{read})$. This specification "propagates upstream" (subsequently) requiring $(t_{req\_write}, t_{read})$ and $(t_{reaccess}, t_{read})$. In both cases, the "pushing" orientation is reasonable to give the opportunity to processes to issue immediate writing requests before granting access to new readers: $\mathbf{pri}(t_{req\_write}) > \mathbf{pri}(t_{read})$ and $\mathbf{pri}(t_{reaccess}) > \mathbf{pri}(t_{read})$.
4. The nonfree choice between t_{write} and t_{read} is never effective when $k = 1$, what can be easily detected reasoning on the place invariant involving p_{allow_read} and p_{next_write}. Although in that case the requirement to specify a relation between t_{write} and t_{read} would be spurious (and easily avoidable with net-level techniques), it is not so if we want a specification that is valid independently of the initial marking, as we do. Since whenever the conflict is effective there are less than k writers waiting, let probabilities decide: $\mathbf{pri}(t_{write}) = \mathbf{pri}(t_{read})$. This specification "propagates upstream" requiring (subsequently) $(t_{req\_read}, t_{write})$ and $(t_{reaccess}, t_{write})$, which can both be deduced from previous specifications. (Propagation would require also $(t_{next\_write}, t_{read})$, but it is already specified.)

TABLE 1
Some State Space Sizes in the Example

n	m	k	Tangible	Vanishing	
				with **pri**	without **pri**
3	3	1	46	124	139
10	3	1	2937	9595	11568
20	3	1	47867	152495	191438
10	3	3	2447	9614	11283
20	3	3	41237	150504	183963
10	10	3	2509	10165	11862
20	10	3	53585	193890	236043

The result of the specification is shown in Fig. 8, where also one possible absolute priority assignment is shown. (By the way, the resulting net is a dpGSPN [20].) Now, the weights to allow computing probabilities in random switches would be required for the four resulting (nontrivial) priority classes. For instance, if one out of ten times a reaccess is needed, then $\mathbf{w}(t_{reaccess}) = 1$ and $\mathbf{w}(t_{exec}) = 9$; if 5 percent accesses are to write, then $\mathbf{w}(t_{req\_read}) = 5$ and $\mathbf{w}(t_{req\_read}) = 95$; if the probability to access the database for writing or reading is equal (unless there are k or more writers waiting), then $\mathbf{w}(t_{write}) = \mathbf{w}(t_{read}) = 1$; finally, if the LAN is allocated to any waiting process with equal probability, then $\mathbf{w}(t_{LAN\_w}) = \mathbf{m}(p_{wait\_LAN\_w})$ and $\mathbf{w}(t_{LAN\_r}) = \mathbf{m}(p_{wait\_LAN\_r})$. (See [6, pp. 103, 120] for the topic of marking dependency of transition weights. Following the proof of Theorem 5, the interested reader can observe that even immediate transition weights dependent on the $(t) \in [t]$ that is enabled can be safely allowed.)

This completes the (net-level) modeling of the system. Now, any performance index can be safely analyzed for any values of the parameters and with any analysis or simulation technique.

Compared to state space methods to check well-definition, first of all, it is worth noticing that, even in this small example, the state space becomes large as the different parameters grow. Table 1 shows in a few cases the number of tangible and vanishing markings, what gives an impression of the complexity that state space methods would face. Observe that, since many irrelevant vanishing markings are avoided by fixing an arbitrary ordering between nonrelated classes, the number of vanishing markings that are dealt with after applying net-level methods is smaller than that produced by purely state space-level methods (in Table 1 they appear in the last two columns, labeled "with pri" and "without pri," respectively).

Besides the state space size, which in some cases needs not be checked completely to analyze well-definition, there are several more reasons why state space methods can perform worse:

- Each time that a state space check finds a problem and some specification is added to solve it, the check should be restarted again because one cannot be sure that the new specification has no effect on the behavior previously explored.
- If problems (confusion, indirect conflict, etc.) can be effective, then they could be detected very late in the reachability graph generation process, and require to start all over again. Even worse, in the case of simulation, if the marking where the problem arises has low probability to be reached, then it could be detected very far away from the beginning of simulation.
- Even when problems are never effective, if there are many unrelated transitions that may be concurrently enabled in many markings (e.g., they are in different subnets of immediate transitions, a situation that is common in large examples), then the state-level methods would check these situations over and over again, leading to a significant overhead (especially in simulation).
- A state-space check would be required for any new initial marking because one cannot be sure that a specification that is complete for one marking is also for another, as it happens with the case $k = 1$ versus $k > 1$ in the example.

7 Discussion and Concluding Remarks

The method introduced in this paper guides the modeler in the task of defining the priorities (and weights) of immediate transitions in a GSPN model. The essential features of this method are:

- The resulting model is well-defined, i.e., the underlying stochastic process is unambiguosly defined. This is unfortunately not true for all models built according to the current net-level definition of GSPN, which is implemented in most tools.
- The method works at the net level, with the consequent efficiency and robustness, keeping (net based) modeling and (possibly state space based) analysis separate. Even in the cases where the state space is used for the analysis, the method is useful because it allows to reduce the number of vanishing markings by appropriately exploiting the independence of unrelated transitions.
- Modeling flexibility or expressiveness are not diminished; the modeler can even decide to break stochastic confusion by probabilities if the system under study requires it (the current net-level definition of GSPN forces to break confusion by priorities).

A potential disadvantage of net-level methods is the possibility of spurious warnings (i.e., requiring the specification of priorities between transitions when it is actually irrelevant). In some cases, those apparently spurious warnings are actually important (e.g., the case $k = 1$ versus $k > 1$ in Section 6). In any case, we believe that all warnings are somehow useful because they force the modeler to better understand the model and perhaps they serve as a suggestion to modify it so that transitions who are in fact independent appear to be so looking at the net. Anyway, net-level techniques considering also the initial marking can be applied to reduce the number of such spurious warnings, particularly taking into account mutual exclusion relations that can be deduced from net invariants, and which modify the conflict relations [8], [6]. (The case of $k = 1$ in Section 6 is an example.) We did not consider them in this work for simplicity, but there is no conceptual problem to do so.

Also, for simplicity, we considered GSPN without inhibitor arcs, although the method can be extended to this case by appropriate modification of the basic conflict and causal relations, as is done in [8], [6]. Trying to extend the method to apply it also to the high level, or colored, extensions of GSPN [9] is a direction of immediate future work.

We propose to apply this method *whenever* GSPN (with immediate transitions) is used to model systems from *whichever* application domain. Therefore, the method should be automatically supported by modeling tools, a task we are currently undertaking. A prototype implementation (in Java) to be integrated within the GreatSPN package has been developed. It loads a net designed with GreatSPN that is initially considered as having no defined priorities. The tool then shows on the model the transitions pairs for which a priority specification is required, and then the modeler can specify them graphically. As a required arc is specified, the algorithm is called, and the new required arcs are computed and visualized. When all required arcs have been specified by the user, the net can be saved, and the tool chooses a total ordering among transitions consistent with the partial order specified by the modeler. Currently, the tool allows stochastic confusion to be resolved using probabilities, although, in this case, the solution algorithms of GreatSPN cannot be used (updating these solution modules is also in progress).

Concerning the relative versus absolute definition of priorities, we have already pointed out that, once the model with relative priorities is well-defined, absolute priorities can be assigned in any coherent way to get a standard GSPN to input to an existing tool, and to reduce the number of vanishing markings due to irrelevant interleavings. In any case, relative priorities appear to be better suited to compositional modeling, where well-definition problems are particularly severe [13], as it was discussed in [20]. Therefore, another direction of future work is the development of compositional modeling methodologies based on the new net-level definition of GSPN presented here.

Acknowledgments

The authors are grateful to the reviewers of a previous version of the manuscript. Their comments and questions helped to significantly improve the work. This work was supported by the Italy-Spain exchange programme HI1998-0196. The work of E. Teruel was also supported by projects CICYT TAP98-0679 and TIC2001-1819. The work of M. De Pierro and G. Franceschinis has been partially supported by the MIUR through the local research funds (60 percent project) and the FIRB project "Perf" (n.RBNE019N8N). During the development of the work, M. De Pierro was graduate student at the Dipartimento di Informatica, University di Torino.

REFERENCES

[1] M.H.T. Hack, "Decidability Questions for Petri Nets," PhD thesis, MIT, Cambridge, Mass., Dec. 1975.

[2] J.L. Peterson, *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, 1981.

[3] E. Best and M. Koutny, "Petri Net Semantics of Priority Systems," *Theoretical Computer Science,* vol. 96, pp. 175-215, 1992.

[4] F. Bause, "On the Analysis of Petri Nets with Static Priorities," *Acta Informatica,* vol. 33, no. 7, pp. 669-686, 1996.

[5] M. Ajmone Marsan, G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems," *ACM Trans. Computer Systems,* vol. 2, no. 2, pp. 93-122, 1984.

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

[7] E. Smith, "On the Border of Causality: Contact and Confusion," *Theoretical Computer Science,* vol. 153, nos. 1-2, pp. 275-270, 1996.

[8] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte, "Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications," *IEEE Trans. Software Eng.,* vol. 19, no. 2, pp. 89-107, Feb. 1993.

[9] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "Stochastic Well-Formed Colored Nets for Symmetric Modelling Applications," *IEEE Trans. Computers,* vol. 42, no. 11, Nov. 1993.

[10] Y. Li and C.M. Woodside, "Complete Decomposition of Stochastic Petri Nets Representing Generalized Service Networks," *IEEE Trans. Computers,* vol. 44, no. 8, Aug. 1995.

[11] S. Donatelli and P. Kemper, "Integrating Synchronization with Priority into a Kronecker Representation," *Performance Evaluation,* vol. 44, nos. 1-4, pp. 73-96, 2001.

[12] O. Botti and L. Capra, "A GSPN Based Methodology for the Evaluation of Concurrent Applications in Distributed Plant Automation Systems," *Euromicro J. Systems Architecture,* vol. 42, pp. 503-530, 1996.

[13] S. Donatelli and G. Franceschinis, "The PSR Methodology: Integrating Hardware and Software," *Application and Theory of Petri Nets 1996,* J. Billington and W. Reisig, eds., 1996.

[14] A. Mazzeo, N. Mazzocca, S. Russo, and V. Vittorini, "A Method for Predictive Performance of Distributed Programs," *Simulation Practice and Theory,* pp. 65-82, Jan. 1997.

[15] M. Ajmone Marsan and R. Gaeta, "Modeling ATM Systems with GSPNs and SWNs," *Performance Evaluation Rev.,* vol. 26, no. 2, pp. 28-37, 1998.

[16] M. Ajmone Marsan, C. Casetti, R. Gaeta, and M. Meo, "Performance Analysis of TCP Connections Sharing a Congested Internet Link," *Performance Evaluation,* vol. 42, nos. 2-3, pp. 109-127, 2000.

[17] M. Ajmone Marsan, R. Gaeta, and M. Meo, "Accurate Approximate Analysis of Cell-Based Switch Architectures," *Performance Evaluation,* vol. 45, no. 1, pp. 33-56, 2001.

[18] M. Silva and E. Teruel, "Petri Nets for the Design and Operation of Manufacturing Systems," *European J. Control,* no. 3, pp. 182-199, 1997.

[19] I. Mura and A. Bondavalli, "Hierarchical Modelling and Evaluation of Phased-Mission Systems," *IEEE Trans. Reliability,* vol. 48, no. 4, 1999.

[20] E. Teruel, G. Franceschinis, and M. De Pierro, "Clarifying the Priority Specification of GSPN: Detached Priorities," *Proc. Int'l Workshop Petri Nets and Performance Models (PNPM '99),* pp. 114-123, 1999.

[21] G. Ciardo and R. Zijal, "Well-Defined Stochastic Petri Nets," *Proc. Fourth Int'l Workshop Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS '96),* pp. 278-284, 1996.

[22] M.A. Qureshi, W.H. Sanders, A.P.A. van Moorsel, and R. German, "Algorithms for the Generation of State-Level Representations of Stochastic Activity Networks with General Reward Structures," *IEEE Trans. Software Eng.,* vol. 22, no. 9, pp. 603-614, Sept. 1996.

[23] F. Bause, "Analysis of Petri Nets with a Dynamic Priority Method," *Application and Theory of Petri Nets 1997,* P. Azéma and G. Balbo, eds., 1997.

Enrique Teruel received the Industrial Engineering degree from the University of Zaragoza, Spain, in 1990, and the PhD degree in systems engineering and computer science from the same University in 1994. He works in the Departamento de Informática e Ingeniería de Sistemas, where he was an assistant professor from October 1992 to December 1997, and is currently an associate professor. He is in charge of courses on systems theory and automatic control in the Centro Politécnico Superior of Zaragoza. He has coauthored more than 20 papers in refereed journals, international conferences, and research monographs on the modeling and analysis of concurrent discrete event dynamic systems, using Petri net-based formalisms, and developing structural reasoning and techniques, with applications in diverse domains from industrial engineering to computer science.

Giuliana Franceschinis received the Laurea degree in computer science from the University of Torino, Italy, in 1986, and the PhD degree in computer science from the same University in 1992. From November 1992 to October 1998, she has been an assistant professor in the Computer Science Deptartment at the University of Torino. From November 1998 to October 2002, she was an associate professor at the University of Piemonte Orientale at Alessandria. Since November 2002, she has been a full professor in the same university, where she teaches computer architecture, operating systems, and simulation. Her current research interests are in the area of dependability and performance evaluation of computer and communication systems and in stochastic Petri nets theory and applications. She is a coauthor of the book *Modelling with Generalized Stochastic Petri Nets* (John Wiley and Sons, 1995). She is member of the IEEE Computer Society.

Massimiliano De Pierro received the Laurea degree in computer science from the University of Torino, Italy, in 1999. After his university studies, he worked for a period of one year at the Italian ENEL research center, investigating fault tolerance in embedded systems. He is currently working on his PhD thesis, which is devoted to the application of structural analysis techniques to Stochastic Well-formed Petri Nets (SWN's).