Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

Efficient simulation of Stochastic Well-Formed Nets through symmetry exploitation

Marco Beccuti
Dipartimento di Informatica,
Università degli Studi di Torino, Torino, Italy

Giuliana Franceschinis
DiSIT
Università del Piemonte Orientale, Alessandria, Italy

ABSTRACT

Stochastic Well-Formed Nets (SWN) is a High-Level Stochastic Petri Net formalism supporting performability analysis. The symbolic marking and firing notions in SWNs allow to automatically aggregate states achieving significant reductions in highly symmetric models. If the reduced state space is still too large, simulation may be applied exploiting symbolic marking and firing to achieve more efficient handling of the Future Event List. This technique is implemented in the GreatSPN tool.

In this paper symmetry based simulation methods are presented, their strong and weak points are discussed, the issue of performance indices definition and computation is introduced, and an extension exploiting the most recent results on partial symmetries is proposed.

1 Introduction

The Stochastic Petri Nets (SPN) formalism and its generalizations (e.g. Generalized SPN - GSPN, Stochastic Well-Formed Nets - SWN) are powerful languages for the specification of stochastic processes (Continuous Time Markov Chains - CTMC), thus supporting performance and reliability analysis of systems modeled by means of such formalisms. State space based analysis methods may become impractical when the number of states is very large: the notions of symbolic marking and firing in SWN models have been introduced to automatically generate state aggregates, achieving a reduction in the state space size that may span orders of magnitude for highly symmetric models (Chiola, Dutheillet, Franceschinis, and Haddad 1993). Despite this, it may be the case that the size of the aggregate CTMC is still too large, and the simulation option remains the only viable way to go; moreover simulation may also be useful when the exponential distribution is not appropriate for modeling the duration of certain activities. The symbolic marking and firing notions turn out to be quite effective even in simulation: in fact they induce a more efficient handling of the Future Event List (FEL) thus producing a speed up in the simulation (Chiola, Franceschinis, and Gaeta 1992, Gaeta 1996): this technique has been implemented in the SWN symbolic simulator of the GreatSPN tool (Baarir, Beccuti, Cerotti, De Pierro, Donatelli, and Franceschinis 2009).

In this paper, after recalling the SWN formalism and the notation of symbolic marking and firing (Sec.2), the symmetry based efficient simulation methods are presented (Sec.3), and the issue of performance indices definition and computation in this framework is also discussed. Finally, the possibility of extending the method to inherit some of the most recent results on partial symmetries exploitation (Baarir, Beccuti, Dutheillet, Franceschinis, and Haddad 2011) is considered (Sec.4).

2 Stochastic Well-Formed Nets, symbolic marking and firing

Before describing the discrete event simulation of SWN models, the SWN formalism is intuitively introduced using the net in Fig. 1 assuming that the reader has some basic knowledge of SPN (Ajmone Marsan, Balbo, Conte, Donatelli, and Franceschinis 1995) and Colored Petri Nets (CPN)(Jensen 1997). Then the symbolic marking and firing notions for SWNs are recalled. A formal definition of both the formalism and the algorithms exploiting the symmetries can be found in (Chiola, Dutheillet, Franceschinis, and Haddad 1993).

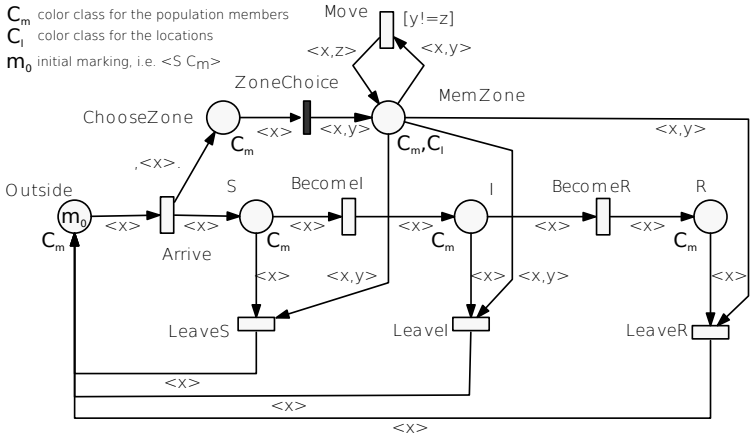


Figure 1: SWN net inspired to SIR model

2.1 SWN running example and informal definition

The SWN in Fig. 1 is inspired by the well-known epidemiological Susceptible-Infectious-Recovered (SIR) model, which describes the progress of an epidemic in a large population. The population members may be in one of four states: 1. *Outside* (place *Outside*) containing the people in the environment outside the system under observation that could enter the system at some point and leave it later on; 2. *Susceptible* (place *S*) containing all the people susceptible to the disease; 3. *Infectious* (place *I*) containing all the infected people; 4. *Recovered* (place *R*) containing all the recovered people that are immune to the disease. Each member of the population typically progresses from susceptible to infectious (transition *BecomeI*) and from infectious to recovered (transition *BecomeR*), and the sojourn time of each single subject in such states is a random variable with an exponential distribution.

The population in the observed system may grow and decrease due to transitions *Arrive* and *LeaveS*, *LeaveI*, *LeaveR*. The model also represents the fact that people are distributed among different zones: the marking of place *MemZone* encodes the mapping between members of the population in the system and their position. Each member of population randomly chooses his zone upon arrival (immediate transition *ZoneChoice*), and can randomly move among the zones (transition *Move*).

In an SWN, as in any colored net formalism, a *color domain* $cd()$ is associated with places and transitions. The color domain of a place defines the possible colors of the tokens that it contains, while the color domain of a transition defines its possible *instances*. The enabling conditions and the state change associated with each transition instance are specified through functions associated with arcs: given the color identifying an instance of the transition connected to the arc, the function provides the (multi)set of colored tokens that will be added to or removed from the place connected to the arc. Color domains in SWNs are expressed as Cartesian products of *color classes* ($\mathcal{C} = \{C_1, \ldots, C_n\}$ is the set of all classes), which may be seen as primitive domains and may be partitioned into *static subclasses*. The colors of a class represent entities of the same nature (e.g. population members) and may be ordered[1]. The colors within a static subclass behave similarly (e.g. population members with the same immune response).

In the net of Fig. 1 there are two color classes C_m and C_l modeling the population members and the locations (zones) where they can be. The following static subclass partition could be defined, e.g. $C_m = C_m^{strong} \cup C_m^{weak}$, identifying a different immune response among people, and $C_l = C_l^{Good} \cup C_l^{Bad}$ defining zones with different pollution levels. The color domain of places *S, I, R* and *Outside* is C_m, while place *MemZone* has color domain $C_m \times C_l$ (denoted C_m, C_l in the figure, as in GreatSPN syntax).

The transition color domains, are defined through a list of typed variables ($var(t)$), whose types ($type(var(t))$) are selected among the color classes. The variables of a transition appear in the functions annotating its arcs and can be interpreted as function parameters; a transition instance binds each variable

[1] A color class is ordered iff there exists a circular order on its colors defined through a successor function.

to a specific color of proper type. A *guard* can be used to define restrictions on the allowed instances of a transition: it is a Boolean expression defined on the transition color domain taking the form of a *standard predicate*. The terms of a standard predicate are *basic predicates*, which allow to compare colors assigned to variables of the same type ($x = y$), or to test whether a color element belongs to a given static subclass ($d(x) = C_{i,j}$), or to compare the static subclasses of the colors assigned to two variables ($d(x) = d(y)$).

For instance the color domain of transitions *Move* is defined as $x : C_m$, $y,z : C_l$; its guard $y! = z$ means that only the instances which move a subject from a zone (y) to a different zone (z) are allowed.

Arc functions are expressed as sums of tuples (the tuple syntax simply represents the Cartesian product of its elements), where each element in a tuple is chosen from a set of predefined *basic functions*, whose domains and codomains are respectively color classes and multisets (also called bags) on color classes. The allowed basic functions are the projection, the diffusion/synchronization, and the successor functions (only for ordered classes); a linear combination of basic functions is also a basic function. Input, output and inhibitor arcs are denoted $I,O,H[p,t] : cd(t) \rightarrow Bag(cd(p))$.

The specification of the stochastic behavior is given by associating (integer) priorities and (real) weights with the transitions. Transitions with priority zero, called *stochastic timed transitions*, fire after a random delay, with a negative exponential distribution; transitions with priority greater than zero are called *immediate transitions* and fire in zero time. The weight of a stochastic timed transition is interpreted as the rate of the corresponding firing time distribution, while that of an immediate transition allows to compute a probability distribution, to be used when the transition firing involves some conflict resolution.

Let us introduce the concepts of *ordinary marking* and *transition instance*.

Definition 1 (Ordinary Marking) An ordinary marking m is a function mapping each place p into a multiset on $cd(p)$, denoted as a weighted sum of color elements (indeed, a place can contain more than one token of a given color).

For instance an ordinary marking for the model in Fig.1 assuming the basic color classes $C_m = \{m1,m2,m3\}$ and $C_l = \{l1,l2,l3\}$ is $m = I(\langle m1\rangle + \langle m2\rangle + \langle m3\rangle) + MemZone(\langle m1,l2\rangle, \langle m2,l2\rangle, \langle m3,l3\rangle)$ representing the state where $m1,m2$ are infected in zone $l2$ and $m3$ is infected in zone $l3$.

Definition 2 (Transition Instance) A transition instance is an assignment of actual values to the transition variables $var(t)$. We use the notation $\langle t,c\rangle$ for an instance of transition t, where c represents the assignment of actual values to the transition parameters.

For example, consider the marking $m = Outside(\langle m1\rangle, \langle m2\rangle)ChooseZone(\langle m3\rangle)$: it enables three instances of immediate transition *ZoneChoice*, with $x = m3$ and $y = l1$ or $y = l2$ or $y = l3$ (no instance of *Arrive* is enabled because, being timed, it has lower priority).

A transition instance $\langle t,c\rangle$ is enabled and can fire in a ordinary marking m iff: 1) its guard evaluated on c is true; 2) $\forall p \in P, I[p,t](c) \leq m(p) \cap H[p,t](c) > m(p)$, where $\leq$ and $>$ are comparison operators between multisets: 3) there is no other transition instance with higher priority satisfying points 1) and 2).

The firing of enabled transition instance $\langle t,c\rangle$ in m produces a new marking m' such that $\forall p \in P, m'(p) = m(p) - I[p,t](c) + O[p,t](c)$

The constraints on the syntax of SWNs allow the automatic exploitation of the behavioral symmetries of the model through a symbolic representation of markings and transition instances. This idea is based on the *Symbolic Marking* (SM) a compact representation for a set of equivalent ordinary markings, where the actual color of tokens is abstracted away, but the ability to distinguish tokens with different colors and to establish their static subclass is retained.

Definition 3 (Symbolic marking) An SM is an equivalence class of ordinary markings; two markings are equivalent if one can be obtained from the other by applying a color permutation preserving static subclasses.

For instance the two markings $m = I(\langle m1\rangle + \langle m2\rangle + \langle m3\rangle) + MemZone(\langle m1,l2\rangle, \langle m2,l2\rangle, \langle m3,l3\rangle)$ and $m' = I(\langle m1\rangle + \langle m2\rangle + \langle m3\rangle) + MemZone(\langle m2,l2\rangle, \langle m3,l2\rangle, \langle m1,l3\rangle)$ belong to the same SM (where there

are three infected members two in one zone and one in another zone) since we may obtain m' from m by applying the permutation that exchanges the colors $m1$ and $m3$, and viceversa.

A SM can be represented by means of *dynamic subclasses* (denoted $Z^j_{C_{i,k}}$ where j is the index of the dynamic subclass), a color class partition which groups together the tokens appearing in the same set of places with the same multiplicity and belonging to the same static subclass. A dynamic subclass representing n different color elements has cardinality n (denoted $|Z^j_{C_{i,k}}| = n$). An example of SM corresponding to six ordinary markings, where there are three infected members two in one zone and one in another zone, is $\widehat{\mathbf{m}} = I(\langle Z^0_m \rangle + \langle Z^1_m \rangle) + MemZone(\langle Z^0_m, Z^I_l \rangle + \langle Z^I_m, Z^0_l \rangle)$ with $|Z^0_m| = 1$, $|Z^1_m| = 2$, $|Z^0_l| = 1$, $|Z^1_l| = 1$, $Z^2_l| = 1$, where a generic member is into the dynamic subclass Z^0_m, two are grouped into Z^1_m and the zones are associated with $Z^i_l, i = 1,2,3$.

A *symbolic firing rule* can be directly applied on SM, introducing the concept of *symbolic instance*

Definition 4 (Symbolic Transition Instance) A symbolic transition instance is a binding of dynamic subclasses to the transition variables $var(t)$. We use the notation $\langle t, \widehat{c} \rangle$ for a symbolic instance of transition t, where $\widehat{c}$ represents the assignment of actual dynamic subclasses to the transition parameters.

A symbolic transition instance represents a set of ordinary transition instances which can be obtained by valid assignments of colors to the dynamic subclasses appearing in $\widehat{c}$. When the same dynamic subclass Z^j_i with cardinality greater than 1 is bound to more than one variable in $var(t)$, an additional specification is required, indicating whether the variables are bound to the same or different elements of Z^j_i. An example of symbolic instance is shown in the next section.

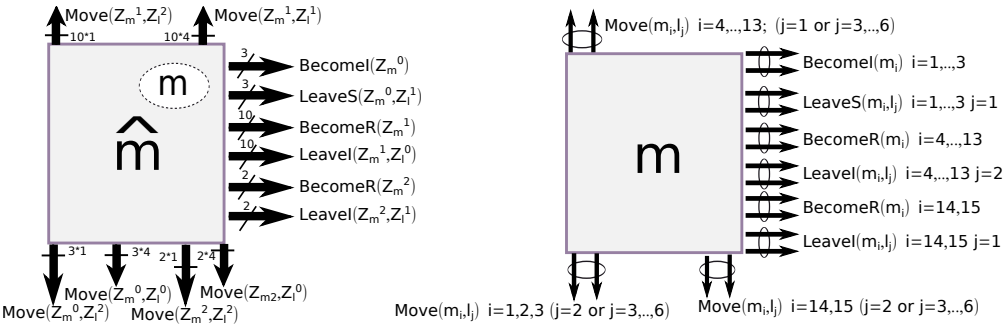
3 Discrete event simulation based on symbolic marking and firing

The classical discrete event simulation approach consists in generating, from a stochastic model specification, n execution traces. This is obtained running an event scheduling and time-advance algorithm, following the time-ordered sequence of (stochastic) events that may occur in each state. The model status at each point in (simulated) time corresponds to the SWN marking, the events correspond to the firings of transition instances, and the state change corresponds to the effect of firing a transition instance.

Typical performance measures to be collected along the simulation can be defined through cumulative and instantaneous reward functions of markings and transition firings. Petri Nets simulation can take advantage of several optimizations: in fact the FEL update required after each state change, may be efficiently performed by taking advantage of the model structure (and of the structural conflict and causal connection relations between transitions); moreover, a considerable saving in time may derive from a properly designed enabling test for (colored) transition instances (taking advantage of the specific structure of the SWN arc functions) (Capra 2007, Liu and Heiner 2010). Last but not least, the peculiar characteristics of the SWN formalism which favors the automatic exploitation of the model behavioral symmetries, may lead to a substantial saving in the FEL management. This aspect will be discussed in the next subsection.

The simulation approach allows more freedom in the type of stochastic process to be analyzed than other methods (usually based on numerical analysis). Specifically, it is possible to consider general firing time distributions. The only constraint that must be satisfied to still exploit the SWN symmetries is that all instances of a given transition share the same type of distribution, moreover the parameters of such distribution must be the same for all instances involving colors that belong to the same static subclass (i.e. only a limited color dependency is allowed). Hereafter we also assume that the distributions are such that there is a null probability that two events are scheduled to occur at the same time in any simulation run.

Observe that when general transition delay distributions are allowed, the complete specification of the stochastic process behavior requires to choose the type of memory policy (age vs. enabling memory, i.e. choose whether to keep memory of the remaining firing time of a transition when it is disabled by another transition firing). In case a non exponential transition t with age memory has also multiple server semantics then it is necessary to choose one among several possible descheduling/rescheduling policies to be applied



$\widehat{\mathbf{m}} = S(\langle Z_m^0 \rangle) + I(\langle Z_m^1 \rangle + \langle Z_m^2 \rangle) + MemZone(\langle Z_m^0, Z_l^1 \rangle + \langle Z_m^1, Z_l^0 \rangle + \langle Z_m^2, Z_l^1 \rangle)$ with $|Z_m^0| = 3,\ |Z_m^1| = 10,\ |Z_m^2| = 2,\ |Z_l^0| = 1,\ |Z_l^1| = 1, |Z_l^2| = 4$

$\mathbf{m} = S(\langle m_1 \rangle + \langle m_2 \rangle + \langle m_3 \rangle) + I(\langle m_4 + ... + m_{15} \rangle) + MemZone(\langle m_1, l_1 \rangle + ... + \langle m_3, l_1 \rangle + \langle m_4, l_2 \rangle + ... + \langle m_{13}, l_2 \rangle \langle m_{14}, l_1 \rangle + \langle m_{15}, l_1 \rangle)$

Figure 2: Enabled symbolic instances and an example of their instantiation.

as the enabling degree of the transition instances of t decreases or increases (possible choices are first/last event inserted into the FEL, or event with earliest/latest scheduled or remaining firing time).

3.1 Efficient simulation of SWN models

The stochastic process describing the behavior of an SWN model with immediate and exponentially distributed timed transitions is a CTMC: it is actually an exactly lumpable CTMC (Chiola, Dutheillet, Franceschinis, and Haddad 1993), hence its states may be grouped into equivalence classes. The adoption of the symbolic marking representation instead of the ordinary one corresponds to choosing an higher abstraction level for the state description. The interesting property of SWNs is that such higher abstraction level does not introduce any approximation in the model behavior: indeed any simulation trace that can be observed in the *ordinary simulation* of an SWN model is stochastically equivalent to a corresponding simulation trace in the *symbolic simulation* and viceversa.

In the rest of this subsection we discuss how the symbolic marking and firing representation can be exploited to improve the simulation efficiency. In the next subsection some experimental results are reported showing that the symbolic simulation is very convenient in case of highly symmetric systems, while in other cases it does not achieve a significant gain.

The symbolic marking representation efficiency is based on two observations: (1) due to the type of functions and guards (and transition delay distributions) associated with the SWN arcs and transitions it is possible to forget about the actual color identities while retaining the ability to decide whether any two colors are equal or different, and which static subclass they belong to, (2) often the colored tuples contained in a place may be more efficiently expressed as Cartesian products of color subsets. Dynamic subclasses are just disjoint subsets of colors (belonging to the same static subclass) that are "anonymized" (since they may enable equivalent event sequences), but remain distinguishable. The number of dynamic subclasses in each static subclass partition must be minimal, but must allow to express the marking of each place as a sum of Cartesian products of dynamic subclasses. This induces a factorization in the marking representation which is particularly effective as the cardinality of color classes grows larger. This aspect is not so relevant in simulation as it is when generating the whole state space.

The other important observation that has an impact on simulation efficiency is that the symbolic instance definition induces a factorization in the event notices that may influence the FEL management. Let us illustrate this on our running example, considering the SM $\widehat{\mathbf{m}}$ shown in Fig. 2. It represents a situation were three healthy (Z_m^0) plus two infected (Z_m^2) persons are in one zone (Z_l^1), while there are ten infected persons (Z_m^1) in a different zone (Z_l^0). The other four zones (Z_l^2) do not host any person.

Fig. 2 shows the symbolic instances enabled in $\widehat{\mathbf{m}}$ and the ordinary instances enabled in one ordinary marking $\mathbf{m}$ included in $\widehat{\mathbf{m}}$: e.g. the symbolic instance $\langle LeaveS, x = Z_m^0, y = Z_l^1 \rangle$ represents ($|Z_l^1| \cdot |Z_m^0| = 3$) three ordinary instances enabled in $\mathbf{m}$: $\langle LeaveS, x = m_i, y = l_1 \rangle, i = 1, \ldots, 3$. Similarly, the symbolic instances

Table 1: Mean length of FEL varying C_m cardinality for the three sets of experiments

	$\|C_l\| = 5$ and all trans. rate 1		$\|C_l\| = 5$, LeaveS rate 10 and BecomeI rate 0.01		$\|C_l\| = 10$ and all trans. rate 1	
$\|C_m\|$	$\|Sym.FEL\|$	$\|Ord.FEL\|$	$\|Sym.FEL\|$	$\|Ord.FEL\|$	$\|Sym.FEL\|$	$\|Ord.FEL\|$
100	13.067366	106.207794	7.171052	103.540184	18.499599	104.478897
300	14.176353	310.145691	8.212779	305.355957	22.029955	306.801819
500	14.565784	513.060730	8.471622	506.676910	23.028669	508.701172
700	14.817933	715.552795	8.523383	707.779175	23.513765	710.343689
900	14.857899	917.777344	8.571803	908.749207	23.795662	911.795166
1100	14.893694	1119.715454	8.609436	1109.621948	24.135056	1113.204346
1300	15.098781	1321.607910	8.643605	1310.431885	24.412901	1314.378662
1500	15.152643	1523.244629	8.673923	1511.180420	24.684221	1515.488525

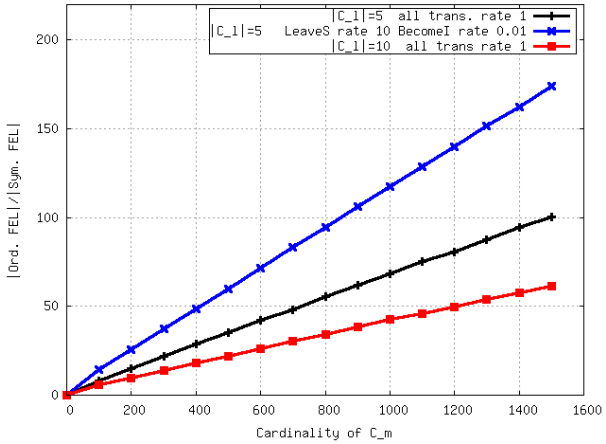


Figure 3: Reduction factor achieved by the first three experiments

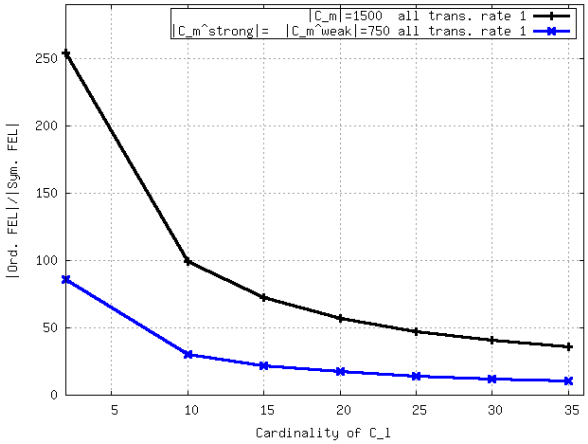


Figure 4: Reduction factor achieved by the last two experiments

$\langle BecomeR, x = Z_m^j \rangle, j = 1,2$ represent respectively ten ($|Z_m^1|$) and two ($|Z_m^2|$) ordinary instances. Six symbolic instances of transition *Move* (representing a total of 75 ordinary ones) are also enabled. Each enabled symbolic instance $\langle t, \hat{c} \rangle$ has a corresponding event notice in the FEL, and $|\langle t, \hat{c} \rangle|$ scheduled firing times (one for each represented ordinary instance); an optimization can be applied in the case of exponentially distributed firing times, i.e. storing only the earliest firing time for each symbolic instance (exploiting the memoryless property of the exponential, and the fact that the minimum among k exponentially distributed random variables has an exponential distribution with rate equal to the sum of the k rates).

3.2 Experimental results

In this section some experimental results are presented for the model described in section 2, which were performed with the simulation tool integrated in the GreatSPN framework. Five different experiments (in terms of color class cardinality, partition in static subclasses and transition rates) are introduced to show how the symbolic simulation can efficiently exploit the model symmetries to reduce the mean FEL length.

The results of the first three experiments are summarized in Table 1, where the Mean Length of FEL computed by the Symbolic simulation ($|Sym.FEL|$) is compared with the one computed by the Ordinary simulation ($|Ord.FEL|$) varying the cardinality of C_m (first column) and considering C_l and C_m not partitioned in static subclasses. Indeed, the second and third columns correspond to $|Sym.FEL|$ and $|Ord.FEL|$ when $|C_l| = 5$ and all the transitions have rate 1; while the next two columns show the same measures when $|C_l| = 5$, *BecomeI* and *LeaveS* have rates 0.01 and 10 and all the other transitions 1. Finally, the last two columns report $|Sym.FEL|$ and $|Ord.FEL|$ when $|C_l| = 10$ and all the transitions have rate 1.

All these experiments show that the symbolic approach achieves for this model a good reduction factor (i.e. $\frac{|Ord.FEL|}{|Sym.FEL|}$), which depends on the cardinality of C_m. For instance, for the first experiment the reduction

factor is ~ 100.5 when $|C_m| = 1500$; while it is ~ 48 when $|C_m| = 700$. However, the different reduction factors achieved by three experiments and plotted in Fig. 3 show clearly that other parameters may affect this measure too. For instance, comparing the first experiment (first line in Fig. 3) with the second one (second line in Fig. 3) we observe that the choice of transition rates can affect the reduction factor. Indeed, in the second experiment the rates chosen for *BecomeI* and *LeaveS* causes an increase of the probability that a population member is outside (place *Outside*) or susceptible (place *S*). Hence, during the simulation it is more likely to visit SMs where the members are more concentrated in the same few states, leading to a higher level of aggregation. On the other hand, comparing the first experiment (first line in Fig. 3) with the third one (third line in Fig. 3) we observe that reduction factor is inversely proportional to the cardinality of C_l. Indeed increasing the number of zones induces a higher spread in the population members, so that during the simulation it is more likely to visit SMs with a lower level of aggregation.

Finally, the last two experiments show how the reduction factor decreases when increasing the cardinality of C_l given that C_m has a unique static subclass (first line in Fig. 4) or is partitioned in two static subclasses: $C_m = C_m^{strong} \cup C_m^{weak}$, with $|C_m^{strong}| = |C_m^{weak}| = 750$ (first line in Fig. 4). The two lines have similar trend, but the partition of the population members in two subset of equal size (C_m^{strong} and C_m^{weak}) reduces substantially the efficiency of symbolic simulation with respect to the ordinary one.

In summary, highly symmetric models may take advantage of symbolic representation of the marking and transition instance, and significant reduction can be achieved in particular when symmetric transitions have exponentially distributed delay. When symmetric transitions are few or states where they are enabled are rarely reached, the increased complexity in handling symbolic markings and symbolic firing may outperform the FEL reduction advantage. In Section 4 the issue of how to recover the lost advantage in partially symmetric systems is discussed.

3.3 Performance indices computation in symbolic simulation

As already observed, a SM represents a set of equivalent ordinary markings and corresponds to a more abstract state representation: the two abstraction levels (ordinary vs. symbolic markings) identify two different, but strictly related stochastic processes. As already pointed out, any sequence in the ordinary state space has a stochastically equivalent sequence in the symbolic state space. The following statements clarify the relation between the possible behaviors (i.e. possible simulation traces) at the ordinary and symbolic level: 1) any ordinary trace can be mapped onto a symbolic trace, any symbolic trace can be mapped onto the set of ordinary traces it represents; 2) all ordinary traces that are represented by the same symbolic trace are stochastically equivalent, and are equally likely to be undertaken during a simulation experiment; 3) the probability of a symbolic trace is equal to the sum of the probabilities of all ordinary traces it represents.

Any performance measure that is compatible with both abstraction levels (i.e. that can be expressed in terms of the SM) has the same distribution in the two processes, and can be estimated through symbolic simulation: for instance one performance index in this category is the *disease propagation rate*, corresponding to the overall throughput of all instances of transition *BecomeI* (it may possibly be refined taking into account static subclasses), which can be computed at the SM level. But what happens if one is interested in computing a color-dependent performance measure? In (Chiola, Dutheillet, Franceschinis, and Haddad 1993) it has been proven that in SWN with exponentially distributed transition delays, as a consequence of the above stated properties, the stochastic process describing the behavior in time of the model is a CTMC which satisfies the exact lumpability conditions with respect to the (ordinary markings) aggregation induced by the SM. The interesting consequence of the exact lumpability property is that the ordinary markings belonging to any reachable SM are equiprobable (both in transient and in steady state), provided that the set of initial ordinary markings are exactly those represented by the initial SM, and the associated initial probability distribution is uniform. Since it is possible to compute the set of ordinary markings contained into any SM, and due to the equiprobability of the represented markings, it is possible to compute also color dependent performance indices that can be defined as a function of the ordinary markings probability

distribution, while still performing symbolic simulation. For instance the *average number of persons in state I within a specific zone l_j* may be computed exploiting the above property. First for any symbolic marking $\widehat{\mathbf{m}}$ with place I not empty and representing n ordinary markings, we should derive all corresponding ordinary markings which have zone l_j associated with some, say $k > 0$, infected population members. Then we derive the reward of $\widehat{\mathbf{m}}$ as the sum over all such ordinary markings of the corresponding values k, divided by n. In practice, for most performance indices of interest, it is possible to compute the desired reward without actually instantiating all ordinary markings in each SM, but rather using combinatorial arguments.

When transition delays with general probability distribution are admitted, it is still possible to compute an index specifically referring to a given color element c by choosing an intermediate abstraction level: this is obtained by separating c into a new static subclass and computing the required performance index using the SM on the new model. The idea of "tagging" a specific color is also the basis for any path based performance index computation (Balbo, Beccuti, De Pierro, and Franceschinis 2011). Since all color elements within the same static subclass behave in exactly the same way (see properties at the beginning of this subsection), then the performance figures computed for one of them is the same also for all other elements in the same static subclass and with equivalent initial marking. For instance the *average (or the distribution of the) time spent by a specific population member m_i in states S, I or R, at each passage in the system* (i.e. the time between arrival and departure of a member m_i) can be computed by isolating m_i in a separate static subclass. The computed index can be generalized to any m_j with similar characteristics.

4 Symbolic simulation based on Extended SM

In this section we propose a new method for symbolic simulation of SWN, to reach a higher aggregation level in systems with mostly symmetric behavior and occasional local asymmetric behavior (called *partially symmetric systems*). Our example model may be considered as an instance of this class of systems when there are two types of members that have different behaviors with respect to recovery time, so that C_m has to be partitioned accordingly, e.g. $C_m^{strong} = \{m1, m2\}$ and $C_m^{weak} = \{m3\}$ and the rate of *BecomeR* instances should be set to model the dependency on the static subclass of the involved member.

In these partially symmetric systems, the FEL length reduction based on the SM may be less effective due to the need of maintaining a state detail level that is adequate when the behavioral differences arise, but it is redundant elsewhere.

In the context of state space based methods, to cope with this limitation the Extended SRG (ESRG) (Baarir, Beccuti, Dutheillet, Franceschinis, and Haddad 2011) has been introduced, based on the notion of *Extended* SM (ESM): the basic idea is to refine the state description detail level only when this is really needed. In many cases the state aggregation induced by the ESM can be significantly higher than that induced by the SM. In the rest of the paper, we discuss how ESMs can also be used to further improve the efficiency of symbolic simulation of partially symmetric systems.

4.1 ESM approach: definition and notation

Before describing how the ESM can be used to improve the performance of simulation, we have to formalize the notion of ESM. Intuitively, an ESM still represents a set of ordinary markings, but it considers a coarser notion of equivalence than that used by the SM.

Definition 5 (Extended Symbolic Marking) An ESM $\widehat{\widehat{\mathbf{m}}}$ is a pair $\langle SR, E \rangle$ where SR represents a standard SM *ignoring the partition of classes into static subclasses*, and where E is a set of *eventualities*, namely a set of SM obtained by instantiation of SR with respect to the actual static subclass partition. A specific eventuality of $\widehat{\widehat{\mathbf{m}}}$ is denoted $\langle SR, e \rangle$ with $e \in E$. E may not contain all possible instantiations of SR w.r.t. the static subclasses partition: if E contains all the possible instantiations then it called *saturated* and the eventualities specification can be left implicit (they can be automatically derived from the SR).

The idea behind the ESM is to remain at a higher abstraction level (the SR level) as long as possible, and take into account the static subclasses partition only when unavoidable. An eventuality defines an

association of the SR dynamic subclasses with the static subclasses (which may require a refinement of the dynamic subclass partition). Now, we define the function $inst()$ that returns the SM corresponding to a given specific eventuality $\langle SR, e\rangle$ obtained by instantiating the SR dynamic subclasses as specified in e.

An example of ESM for the model in Fig. 1 is $\widehat{\widehat{\mathbf{m}}} = Outside(\langle Z^1_m\rangle) + I(\langle Z^2_m\rangle) + MemZone(\langle Z^2_m, Z^1_l\rangle)$ with $|Z^1_m| = 2$, $|Z^2_m| = 1$ and $|Z^1_l| = 1$. This ESM represents a marking with two members outside the system and an infected member in a generic zone. An example of eventuality for this ESM is $e : Z^1_m = Z^1_{C^{strong}_m} \cup Z^1_{C^{weak}_m}$, $Z^2_m = Z^2_{C^{strong}_m}$, $Z^1_l = Z^1_{C_l}$ which represents the following SM $\widehat{\mathbf{m}} = Outside(\langle Z^1_{C^{strong}_m}\rangle + \langle Z^1_{C^{weak}_m}\rangle) + I(\langle Z^2_{C^{strong}_m}\rangle) + MemZone(\langle Z^2_{C^{strong}_m}, Z^1_l\rangle)$.

This two level representation of an ESM also leads to different types of transition instances: the *instantiated instance* and the *generic instance*.

Definition 6 (Instantiated instance and generic instance)
A *instantiated instance* $\langle t, \widehat{c}\rangle$ of a transition t referring to the eventuality $\langle SR, e\rangle$ corresponds to a transition instance defined on $inst(\langle SR, e\rangle)$. A *generic instance* $\langle t, \widehat{\widehat{c}}\rangle$ of a symmetric transition t referring to a given ESM $\widehat{\widehat{\mathbf{m}}}$ is a transition instance defined on the *SR*.

The instantiated instance $\langle t, \widehat{c}\rangle$ referring to eventuality $inst(\langle SR, e\rangle)$ is an instantiation of a generic instance $\langle t, \widehat{\widehat{c}}\rangle$ referring to the *SR* iff $\langle t, \widehat{\widehat{c}}\rangle$ can be derived from $\langle t, \widehat{c}\rangle$ by ignoring the partition of color classes into static subclasses. As a consequence, when dealing with ESM based simulation, both *generic* and *instantiated firings* will be considered.

Finally let us introduce the notions of *symmetric/asymmetric* transition and ESM, and of Uniform ESM.

Definition 7 (Symmetric transition) Transition t is symmetric if no static subclass appears in its arc functions, no clauses such as $d(x) = d(y)$ or $d(x) \neq d(y)$ or $d(x) = C_i$ or $d(x) \neq C_i$ appear in its guard and its rate does not depend on any static subclass.

The transitions that do not meet this definition are called *asymmetric*. This distinction leads to a partition of the ESMs into two sets: *symmetric* ones, enabling only symmetric transition instances, and *asymmetric* ones having one or more eventualities $e \in E$ which enable one or more asymmetric transition instances. Observe that while it is possible to define the enabling of a symmetric transition at the level of the *SR*, asymmetric transition enabling (and firing) can be defined only at the level of the eventualities.

Definition 8 (Uniform ESM.) An ESM is uniform iff it represents only one SM.

An example of Uniform ESM in our model is the ESM where all the members are outside the system.

4.2 Simulation algorithm based on ESM

In this section the simulation algorithm based on ESM is presented. The key idea is to exploit the ability of ESMs to remain at a higher abstraction level as long as the eventualities can be kept implicit: assuming to start from an initial symmetric saturated or uniform ESM, until the behavior remains locally symmetric only the *SR* marking representation is used to describe the state, and the generic transition instances (potentially representing several instantiated instances) are scheduled in the FEL.

This of course is possible only until the simulation does not reach an asymmetric ESM: when this happens, the algorithm has to probabilistically choose one eventuality e among those represented by such ESM (exploiting the fact that until symmetry is not broken the probability distribution of the represented ordinary markings is uniform, and the eventuality probability is proportional to the number of ordinary markings they represent); then all the instantiated transition instances are checked for enabling in $inst(\langle SR, e\rangle)$ and scheduled in the FEL. In other words asymmetric ESMs force the algorithm to go down at the level of SMs and instantiated transition instances (i.e. all the generic instances must be instantiated). Specifically, in our example model with two classes of members having different recovery time, the algorithm does not reach any asymmetric ESM until there is at least one infected member. Once the symmetry is broken because an asymmetric ESM is reached, the simulator must work at the eventuality (i.e. SM) level until

a "regeneration marking" is met, i.e. until a marking is reached where the influence of the asymmetric behavior stops. Uniform ESMs are indeed regenerative since they represent only one SM.

The pseudo-code of the simulation engine is reported in Algorithm 4.2, where the initial ESM $\widehat{\widehat{\mathbf{m}}}_0$ is passed as input parameter. It can be divided into two phases: the initialization [lines 1-7] and the simulation step [lines 8-32]. In the initialization phase, $\widehat{\widehat{\mathbf{m}}}_0$ is assigned to the current ESM variable $\widehat{\widehat{\mathbf{m}}}$, then its type is saved in the variable *Type* through the method *getType()*. The list of enabled generic and instantiated instances are computed calling the methods *compGenericFiring()* and *compInstatiatedFiring()* respectively. Observe that this code assumes for simplicity that initial ESM can be only a saturated symmetric or uniform ESM, nevertheless it could be extended to consider asymmetric initial marking.

The simulation step is repeated until the current simulation time (i.e. variable *CT*) is smaller than the end simulation time (i.e. *EndTime*) or the FEL is empty. For each step the method *pop()* removes the first event *ev* from the list, and returns it. Then the simulation time is incremented and a new event is processed [lines 10-23]. Three cases are possible according to the type of the current value of *Type*:
1) If the value of Type is 1 or 3, i.e. symmetric saturated or uniform ESM, then *ev* can be only a generic transition instance, hence the algorithm can immediately fire *ev* and update consistently the current ESM (method *updateMarking()*) and its type (method *getType()*)[lines 12-13]
2) If the value of Type is 2, i.e. saturated asymmetric ESM, then the algorithm has to probabilistically choose one SM among those represented by the current ESM [line 15-17]. In particular the method *getEventuality()* returns one eventuality among those in which *ev* is enabled, and the method *Inst()*, taking in input an eventuality, computes the corresponding SM. The chosen SM is updated, and the type of the newly reached SM is tested: if it is 3 then the corresponding uniform ESM is computed [lines 19-21].
3) If the value of Type is 4, then *ev* can be only a instantiated transition instance. Hence the algorithm can immediately fire *ev* and update consistently the current SM [line 18]. Then the type of the newly reached SM is tested: if it is 3 then the corresponding uniform ESM is computed [lines 19-21]. This case is different by the second case since the marking which enables *ev*, is a specific eventuality of an ESM.

After the update of the current marking according with *ev*, the FEL must be updated: all the newly enabled transition instances are inserted and all the transition instances which are no longer enabled are removed [line 24-31]. Two cases are considered: if the value of Type is 4 then only instantiated transition instances are scheduled; otherwise both generic and instantiated transition instances are scheduled.

Some preliminary experiments, computed thanks to a prototype implementation of this algorithm, have confirmed that in case of partially symmetric systems the simulation based on ESM allows a higher reduction of FEL size w.r.t. the one based on SM. For instance, if we consider the asymmetry introduced in the beginning of this section where the rate of *BecomeR* depended on the type of the involved member, then we obtain a reduction factor of 1.87 when C_m is partitioned in two static subclasses and a reduction factor of 3.81 when it is partitioned in four static subclasses.

An optimization for the simulation algorithm based on ESM. The efficiency of the pseudo-code presented in Algorithm 4.2 can be improved in case of saturated asymmetric ESM. Indeed, when a saturated asymmetric ESM is reached through a generic transition firing and all its eventualities are characterized by the same output rate; then the instantiation of the ESM can be delayed after the choice of which enabled transition instance has the earliest scheduled firing time: if a generic transition instance wins the race, then the instantiation may be postponed.

In our example model, this optimization could be applied in all the reached saturated asymmetric ESMs where the rate out of the eventualities is the same, this may happen for example in a state where there is at least one infected member (which may be either strong or weak, or a mix of the two), so that there are two enabled asymmetric transition instances, but the *sum* of the rates of such transitions $BecomeR(x) + LeaveI(x,y)$ is independent on the static class of x (y is the zone, and in this section class C_l

Algorithm 1 Pseudo-code of simulation algorithm based on ESRG

1: **procedure** ESRG_SIM($\widehat{\widehat{\mathbf{m}}}_0$)

FEL = Future Event list, it maintains the list of simulation events.
CT = current time.
$\widehat{\widehat{\mathbf{m}}}_0$ = initial ESM
$\widehat{\widehat{\mathbf{m}}}$ = current ESM.
$\widehat{\mathbf{m}}$ = current SM.
Type = type of current ESM - e.i. {1 saturated symmetric, 2 saturated asymmetric, 3 uniform, 4 eventuality}.
$T^G_{\widehat{\widehat{\mathbf{m}}}}$ = a list containing all the generic enabled transition instances in $\widehat{\widehat{\mathbf{m}}}$.
$T^I_{\widehat{\widehat{\mathbf{m}}}}$ = a list containing all the instantiated enabled transition instances in $\widehat{\widehat{\mathbf{m}}}$. $T^G_{\widehat{\widehat{\mathbf{m}}}} \cap T^I_{\widehat{\widehat{\mathbf{m}}}} = \emptyset$
$T_{\widehat{\mathbf{m}}}$ = a list containing all the enabled transition instance in $\widehat{\mathbf{m}}$.

2: CT:=0; ▷ /*Initialization*/
3: $\widehat{\widehat{\mathbf{m}}} := \widehat{\widehat{\mathbf{m}}}_0$;
4: *Type*:= $\widehat{\mathbf{m}}$.*getType();* ▷ *//type of* $\widehat{\widehat{\mathbf{m}}}_0$ *cannot be 4;*
5: $T^G_{\widehat{\widehat{\mathbf{m}}}} := \widehat{\mathbf{m}}.compGenericFiring()$;
6: $T^I_{\widehat{\widehat{\mathbf{m}}}} := \widehat{\widehat{\mathbf{m}}}.compInstatiatedFiring()$;
7: FEL.insertEvent($T^G_{\widehat{\widehat{\mathbf{m}}}} \cup T^I_{\widehat{\widehat{\mathbf{m}}}}$); ▷ */*End Initialization*/*
8: ***while*** *(CT* $<$ *EndTime) and (*!FEL.empty()*)* ***do***
9: ev *:=FEL.pop();*
10: CT:= ev.getTime()
11: ***if*** *((Type==1) or (Type==3))* ***then*** ▷ */*Marking update*/*
12: $\widehat{\widehat{\mathbf{m}}}$:= $\widehat{\widehat{\mathbf{m}}}$.updateMarking(ev);
13: *NewType:=* $\widehat{\mathbf{m}}$.*getType();*
14: ***else***
15: ***if*** *(Type==2)* ***then***
16: $\widehat{\mathbf{m}}$:=$\widehat{\widehat{\mathbf{m}}}$.Inst(ev.getEventuality());
17: ***end if***
18: $\widehat{\mathbf{m}}$:= $\widehat{\mathbf{m}}$.updateMarking(ev);
19: ***if*** ($\widehat{\mathbf{m}}$.*checkType()==3)* ***then***
20: $\widehat{\widehat{\mathbf{m}}}$=$\widehat{\mathbf{m}}$.getUniform();
21: ***end if***
22: *NewType:=*$\widehat{\mathbf{m}}$.*getType()* ;
23: ***end if*** ▷ */*End Marking update*/*
24: ***if*** *(NewType == 4)* ***then*** ▷ */*FEL update*/*
25: $T_{\widehat{\mathbf{m}}} := \widehat{\mathbf{m}}.compFiring()$;
26: FEL.updateEvent($T_{\widehat{\mathbf{m}}}$);
27: ***else***
28: $T^G_{\widehat{\widehat{\mathbf{m}}}} := \widehat{\widehat{\mathbf{m}}}.compGenericFiring()$;
29: $T^I_{\widehat{\widehat{\mathbf{m}}}} := \widehat{\widehat{\mathbf{m}}}.compInstatiatedFiring()$;
30: FEL.updateEvent($T^G_{\widehat{\widehat{\mathbf{m}}}} \cup T^I_{\widehat{\widehat{\mathbf{m}}}}$);
31: ***end if*** ▷ */*End FEL update*/*
32: *Type:=NewType;*
33: ***end while***
34: ***end procedure***

is assumed to have only one static subclass). This allows to decide whether a symmetric or an asymmetric transition shall win the race, before instantiating the ESM eventualities.

5 Related work

The idea most closely related to that presented in this paper, allowing an efficient handling of the FEL in the simulation of symmetric Petri net models, has been proposed in (Sanders and Freire 1993) for the Stochastic Activity Network (SAN) formalism. The SAN feature which allows to model the presence of symmetries in the model is the composition of models from atomic (sub)models, by means of two operators: *Replicate* and *Join*. The composed model is represented by a tree-like structure whose leaves are atomic models, and whose intermediate nodes specify how atomic models are glued together; the model structure may be reflected in the marking representation, which can be expressed as a hierarchy of submarkings.

Since the submodels under a *Replicate* operator are identical, they all enable exactly the same set of transitions when they have exactly the same (sub)marking; this naturally leads to a *factorization* in the representation of the submarkings of the identical submodels under a *Replicate* operator, as well as a *factorization* of the list of enabled transitions in these submodels. Indeed the marking of a set of n identical submodels may be represented as a set of $n' \leq n$ submarkings, each of which weighted with the number of submodels sharing that same state (of course the n' weights must sum up to n). This representation is more abstract (and efficient) than identifying uniquely each single submodel in the replicate subtree and recording the submarking of each of them separately. When a marking comprises k replicated submodels with the same submarking sm_j, each of these submodels enables the same set of (local) transitions: it is thus possible to include into the FEL only one representative for each transition in this set, and to keep for each of them an ordered list of firing times (a further optimization is possible if firing time distribution is negative exponential, allowing to generate only the earliest firing time in each set).

Our approach is however more general since the expressive power of the (possibly nested) replication operator is not equivalent to that of colors in SWNs, in particular the possibility of defining color domains as Cartesian products of color classes allows to model dynamic associations of elements from different color classes and make the model behavior depend on it. In the SAN formalism this may be partially represented using nested replicate operators, but in this case the association is static. On the other hand the interaction of elements sharing at least one color component is natural in SWN and the state aggregation automatically adapts to it, while the interaction of corresponding subnets from different replicate operators must be explicitly managed by the modeler (through C++ code) and may prevent the automatic aggregation of states. Finally, the SAN formalism does not provide an easy way of handling subnets with similar structure but different parameters which would allow to represent and exploit partial symmetries.

6 Conclusion and future work

In this paper we have presented the symmetry based simulation methods for SWNs discussing through an example the possible improvement in simulation performance that can be achieved on highly symmetric nets with respect to ordinary simulation of the same nets. Efficient computation of performance measures through this method has also been discussed.

Having observed that the efficiency may significantly decrease in case of partially symmetric systems, a new algorithm based on the ESM representation has been proposed to cope with the limitation of the SM based symbolic approach; some results have been derived from a prototype implementation to provide an indication of the possible improvements, an optimization of the method is also discussed. The complete implementation of the new algorithm in the GreatSPN framework is planned as a future work to experiment and evaluate its efficiency on several benchmark models.

REFERENCES

Ajmone Marsan, M., G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. 1995. *Modelling with Generalized Stochastic Petri Nets*. New York, NY, USA: J. Wiley.

Baarir, S., M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis. 2009. "The GreatSPN tool: recent enhancements". *SIGMETRICS Performance Evaluation Review, Special Issue on Tools for Performance Evaluation* 36 (4): 4–9.

Baarir, S., M. Beccuti, C. Dutheillet, G. Franceschinis, and S. Haddad. 2011. "Lumping partially symmetrical stochastic models". *Perform. Eval.* 68 (1): 21–44.

Balbo, G., M. Beccuti, M. De Pierro, and G. Franceschinis. 2011. "Computing first passage time distributions in stochastic well-formed nets". In *ICPE*, edited by S. Kounev, V. Cortellessa, R. Mirandola, and D. J. Lilja, 7–18: ACM.

Capra, L. 2007. "Integrating symmetries and symbolic enabling test for efficient simulation of SWNs.". In *21st EUROPEAN Conference on Modelling and Simulation (ECMS 2007), Prague, Czech Republic.*

Chiola, G., C. Dutheillet, G. Franceschinis, and S. Haddad. 1993, nov. "Stochastic well-formed coloured nets for symmetric modelling applications". *IEEE Trans. on Computers* 42 (11): 1343–1360.

Chiola, G., G. Franceschinis, and R. Gaeta. 1992. "A symbolic simulation mechanism for well-formed coloured Petri nets". In *Annual Simulation Symposium*, 192–201: IEEE Computer Society.

Gaeta, R. 1996. "Efficient Discrete-Event Simulation of Colored Petri Nets". *IEEE Trans. Software Eng.* 22 (9): 629–639.

Jensen, K. 1997. *Coloured Petri nets. Basic Concepts, Analysis Methods and Practical Use (vol.1,2,3)*. New York, NY, USA: Springer Inc.

Liu, F., and M. Heiner. 2010, October. "Computation of Enabled Transition Instances for Colored Petri Nets". In *Proc. 17th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2010)*, Volume 643 of *CEUR Workshop Proceedings*, 51–65. CEUR-WS.org.

Sanders, W. H., and R. S. Freire. 1993, July. "Efficient Simulation of Hierarchical Stochastic Activity Network Models". *Discrete Event Dynamic Systems: Theory and Applications* 3 (2/3): 271–299.

AUTHOR BIOGRAPHIES

MARCO BECCUTI is assistant professor at the Università di Torino. He obtained his Ph.D in 2008 at the Department of Computer Science in the Università del Piemonte Orientale, Alessandria, Italy and Università di Torino, Torino, Italy in "cotutela" with the Université Paris Dauphine, Paris. His research activity is mainly focused on Petri net and Markov decision process theory and applications, performance evaluation, discrete-event simulation. His email address is beccuti@di.unito.it and his web page is http://www.di.unito.it/∼beccuti.

GIULIANA FRANCESCHINIS received her PhD in computer science from the Università di Torino in 1992. From 1992 to 1998, she was an assistant professor at the Computer Science Department of the same University. From 1998 to 2002, she was an associate professor at the Università del Piemonte Orientale at Alessandria, Italy. Since 2002 she is full professor in the same University. Her research interests are in the areas of dependability and performance evaluation of systems through stochastic processes modeling and analysis, in particular by means of stochastic Petri nets and their extensions. Her contributions in these areas are published in several papers in international conference proceedings and journals. She has been involved in several national and international research projects and cooperation initiatives and has been visiting professor in international research institutions. Her email address and web page are giuliana.franceschinis@di.unipmn.it and http://people.unipmn.it/giuliana/.