From Symmetric Nets to Differential Equations exploiting Model Symmetries

MARCO BECCUTI[1], CHIARA FORNARI[1], GIULIANA FRANCESCHINIS[2], SAMI M. HALAWANI[3], OMAR BARUKAB[3], AB RAHMAN AHMAD[3], AND GIANFRANCO BALBO[13]

1. Dipartimento di Informatica, Università di Torino, Italy.
2. Dipartimento di Informatica, Università del Piemonte Orientale, Italy.
3. Faculty of Computing and Information Technology, Rabigh, King Abdulaziz University, Rabigh Branch, Kingdom of Saudi Arabia
Email: {beccuti,fornari,balbo}@di.unito.it, giuliana.franceschinis@di.unipmn.it, {halawani,obarukab,abinahmad}@kau.edu.sa

Stochastic Symmetric Nets (SSN) are a High-Level Stochastic Petri Net formalism which provides a parametric system description and an efficient analysis technique that exploit system symmetries to automatically aggregate its states. Even if significant reductions can be achieved in highly symmetric models, the reduced state space can still be too large to derive and/or solve the underlying stochastic process, so that Monte Carlo simulation and fluid approximation remain the only viable ways that need to be explored. In this paper, we contribute to this line of research by proposing a new approach based on fluid approximation to automatically derive from an SSN model a set of Ordinary Differential Equations (ODEs) which mimic the system behavior, and by showing how the SSN formalism allows to define an efficient translation method which reduces the size of the corresponding ODE system with an automatic exploitation of system symmetries. Additionally, some case studies are presented to show the effectiveness of the method and the relevance of its application in practical cases.

Keywords: Colored Petri Nets, Ordinary Differential Equations, symmetries, symbolic analysis

1. INTRODUCTION

. Stochastic Petri Nets (SPNs) are widely used for the performance analysis of complex Discrete Event Dynamic Systems [1, 2]. The stochastic process which underlies the behavior of an SPN is a Continuous Time Markov Chain (CTMC) whose state space is isomorphic to the reachability set of the SPN [3]. The evaluation of real systems often requires the construction of SPN models with large state spaces (state space explosion) that may hamper the practical relevance of these representations and which have motivated an impressive research effort to devise techniques capable of reducing the impact of this problem as highlighted by the following partial list of approaches.

- aggregation based methods, where states are grouped into classes, according to lumpability criteria (e.g. [4, 5, 6, 7]);
- composition/decomposition based methods (also defined modular or hierarchical), where an efficient representation of the whole state space is given in terms of system component's state spaces (e.g. [8]);
- partial order methods which decrease the state space by exploiting reductions of the possible interleavings of execution traces of concurrent processes(e.g. [9]);
- probabilistic methods where only a faction of the reachable states is really analyzed using the idea of storing only a partial description of visited states so that different states may be mapped onto the same representation (e.g. [10, 11]).
- methods based on the structured representations of the state space which allow to store and manipulate huge state spaces (e.g. [12, 13, 14]);
- out of core methods, which store the state space directly on the computer disk and propose efficiently algorithms to mitigate the disk access latency(e.g. [15, 16, 17]);
- parallel methods, where the state space is generated and encoded exploiting parallel algorithms and shared/distributed memory (e.g.[18, 19]);
- product-form methods which exploit the properties of certain classes of SPNs to compute their solution with algorithms that extend the well known Queuing Network *Product Form Solution* (PFS) to the PN (e.g. [20, 21, 22]).

When the model includes the representations of large groups of elements with similar behavior (e.g., Internet users, or human populations, or reagent molecules) the exploitation of lumpability properties may turn out to be insufficient and performance indices are estimated with Discrete Event Simulation [23, 24] instead of being evaluated with numerical methods. An alternative to the use of simulation is the approximation of the stochastic model with a deterministic one in which the time evolution of interesting quantities is represented with a set of Ordinary Differential Equations (ODEs) whose solution provides the expected values of the performance indices as functions of time (e.g. [25, 26, 27]). The convergence of the solution of the system of ODEs with respect to the expected values of the desired performance indices when the sizes of the involved populations grow very large has been studied by many researches which, starting from the work of Kurtz [28], have shown that the accuracy of the approximation is acceptable when the model can be considered as the representation of a system of interacting population quantities [29, 30]. The approach is based on the idea of replacing the integer vector representation of the markings of the SPN model with a vector of real values, thus implying a "fluidification" of the tokens of the model.

Cases in which this deterministic approximation represents a reasonable analysis approach often refer to real systems characterized by internal symmetries which can be conveniently exploited to devise compact representations. For this purpose, a particular type of Colored Stochastic Petri Nets, called Stochastic Symmetric Nets (SSN)[1] has been proposed which yield a representation of the system in a manner that is naturally suited to the construction of a lumped model [31]. Usually, this compact representation is used only at the level of the construction of the model [32] and is not exploited when the deterministic representation is desired. Indeed, in these cases the unfolding of the model is performed first, translating the compact colored net in a much larger (un-colored) SPN, and subsequently one ODE is written for each place of the resulting (detailed) model.

In this paper we describe how it is possible to automatically derive a reduced set of ODEs, which completely captures the behavior of the original model, exploiting the symmetrical properties of SSN representations not only at the construction level, but also at that of the solution. The idea is that of partitioning the set of ODEs, which can be derived from the original (and detailed) description of the model, into equivalent classes of ODEs which have indistinguishable solutions so that a representative equation is identified for each equivalent class. A reduced model may thus be defined collecting together only these representative equations so that a smaller computational effort is needed for the solution of the model. A similar result has already been discussed by Tschaikowsky and Tribastone in [33, 34] within the context of a fluid framework for PEPA [35], called GPEPA [36] where replicas of identical processes are grouped together and subsequently fluidified to define Fluid Process Algebra models which are exactly lumpable under certain conditions. Here we obtain the reduced set of ODEs following a completely different approach which has however many points in common with that of [33].

The rest of the paper is structured in the following manner. In Section 2, we provide an informal introduction to the SSN formalism with the description of a (relatively) simple model that will be used to help the reader to understand the details of the formalism and to play the role of a "running example" that will be used also to support the validity of our proposal. In Section 3 we recall the results characterizing the cases in which the fluidification of the model is possible and we show how this can be done on the unfolded version of an SSN model; in this same Section the construction of the reduced set of ODEs is also proposed, yielding the derivation of what we call "Symbolic" ODEs which are the representative elements of equivalent classes of basic ODEs. In Section 4 we show the use of our approach in the evaluation of several variants of the Running Example showing the effectiveness of the method both from the point of view of the sizes of the models that can be handled and of the consequent (much) smaller sizes of the systems of ODEs that need to be solved. Finally, in Section 5, we draw some conclusions on the relevance of the attained results and we highlight the research directions that we intend to pursue further to make the method applicable in a wider context.

2. STOCHASTIC SYMMETRIC NETS, SYMBOLIC MARKING AND FIRING

Before describing how to efficiently derive the ODE system from a Stochastic Symmetric Net (SSN) model, the SSN formalism is briefly introduced using the net in Fig. 1 and assuming that the reader has some basic knowledge of Stochastic Petri Net (SPN) [1] and of Colored Petri Nets (CPN) [37]. Then, the symbolic marking and firing notions for SSNs are recalled. A formal definition of both the formalism and the algorithms exploiting the symmetries can be found in [31].

2.1. SSN running example and informal definition

The SSN depicted in Fig. 1 is inspired by the epidemiological Susceptible-Infectious-Removed (SIR) model originally introduced in [38] and corresponds to the study of the progress of an epidemic in a large population as discussed in [39] using the SSN formalism. The model, which reflects this interpretation of the diffusion of an epidemic, assumes that there is an infected area where members of a large population are divided in three classes:

1. Class *Susceptible* groups all the population members that are not ill, but that are susceptible to the disease;
2. Class *Infectious* groups all the infected population members;
3. Class *Recovered* groups all the members of the population that were ill and that, recovering from the disease, are now immune.

Each member of the population typically progresses from susceptible to infectious and from infectious to recovered.

[1] SSN were previously known as *Stochastic Well-formed Nets* [31]; recently the untimed version of such formalism has been included in the High Level Petri Nets ISO standard **ISO/IEC 15909**, under the new name of Symmetric Nets (part 1, annex B.2).

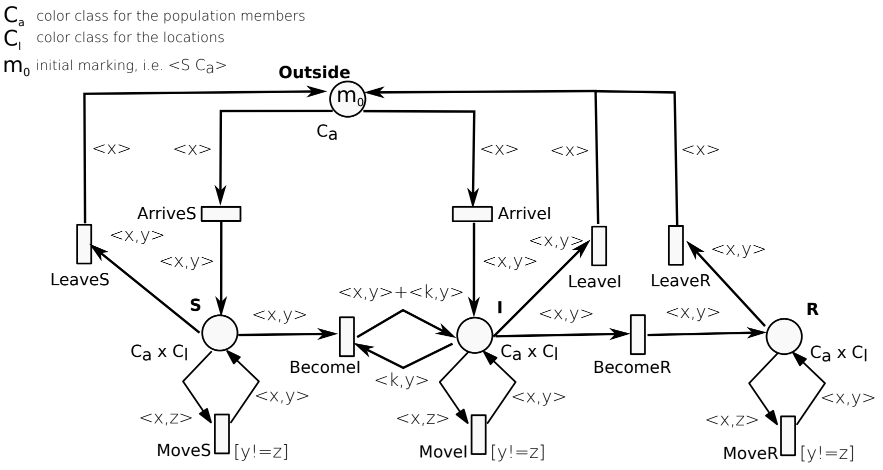


FIGURE 1. SSN net inspired to SIR model.

At any point in time, members of the population can leave the infected area (independently of their individual state) to reach the "outside world"; moreover, members of the outside world can enter the infected area joining the groups of the susceptible or infected population members.

The dynamics of the epidemic in a large population is represented in our model observing that the population members may be in one of four states represented by the following four places:

- Place *Outside* containing the population in the environment outside the system under observation that could enter the system at some point and leave it later on;
- Place *S* containing all the members susceptible to the disease;
- Place *I* containing all the infected members;
- Place *R* containing all the recovered members that are immune to the disease.

Each member of the population typically progresses from susceptible to infectious (transition *BecomeI*) depending on the number of (already) infected members, and from infectious to recovered (transition *BecomeR*). The sojourn times of each single member in any one of such states are random variables with negative exponential distributions

The population in the infected area may grow and decrease due to transitions *ArriveI* and *LeaveS*, *LeaveI*, *LeaveR*.

To make the model more interesting, the infected area is assumed to be divided into several zones characterized by different infection and recovering rates, and the population to be made of members of different age levels that may present different immune responses. Hence the members of the population and the zones are identified by specific sets of colors. Each member of the population randomly chooses to enter one of the zones of the infected area at the moment of his arrival, and can randomly move among the zones (transitions *MoveS*, *MoveI* and *MoveR*).

In an SSN, as in any colored net formalism, *color domains* $cd()$ are 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*[2]. The enabling condition and the state change associated with each transition instance are specified by means of 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 SSNs are expressed by Cartesian products of *color classes* ($\mathcal{C} = \{C_1, \ldots, C_n\}$ is the set of all the classes) or by the neutral element (ε) consisting of a neutral color as in ordinary SPNs. Color classes 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), but only the colors within a static subclass are guaranteed to behave similarly (e.g. population members with the same immune response or age). Color classes may be ordered; a color class is ordered iff it is equipped with a successor function denoted by "!" defining a circular order on its elements.

In the net of Fig. 1 there are two color classes C_a and C_l modeling the age of the members of the population and the locations (zones) where they can be. The following static subclass partition could be defined, $C_a = C_a^{young} \cup \ldots \cup C_a^{old}$, identifying people of the same age possibly characterized by different recovery capabilities, and $C_l = C_l^{MZ_1} \cup C_l^{MZ_n}$ defining macro-zones with different contamination levels. The color domain of places *Outside* is C_a, while places *S, I, R* have color domain $C_a \times C_l$.

The color domains of a transition, are defined through a list of typed variables ($var(t)$), whose types ($type(var(t))$) are selected among the color classes. The variables associated with a transition appear in the functions annotating its arcs and can be interpreted as function parameters; a transition instance binds each variable to a specific color of proper type. A *guard* can be used to define restrictions on the allowed instances of a transition: it is

[2]All the instances of a transition enabled in a given state are allowed to work in parallel.

a Boolean expression defined on the color domain of the transition and taking the form of a *standard predicate*. The terms of a standard predicate are *basic predicates*, which allow one 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 *MoveS* is defined as $x : C_a; y, z : C_l$ meaning that the transition involves a population member (from color class C_a) and two locations (belonging to color class C_l); its guard $y \; != \; z$ means that only the instances which move a member 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 basic functions that are allowed in the formalism are:

- the *projection function* denoted by X_i and defined as:

  $$X_i(c_1, c_2, \dots, c_n,) \mapsto c_i$$

- the *successor function* denoted by $!X_i$ and defined as:

  $$!X_i(c_1, c_2, \dots, c_n) \mapsto !c_i$$

  where the color operator "!" (already introduced) is defined only for ordered color classes and implies a circular ordering on its elements.
- the *diffusion function* (also called synchronization function, depending on whether it annotates an output or an input arc), which is constant, is denoted by S_i and is defined as follows:

  $$S_i(c_1, c_2, \dots, c_n) \mapsto C_i$$

  The diffusion function can be restricted to a static subclass, denoted by $S_{i,j}$ and defined as follows:

  $$S_{i,j}(c_1, c_2, \dots, c_n) \mapsto C_{i,j}$$

Notice that in practice the symbols X_i used above to denote the projection function are substituted by names of transition variables (representing the transition parameters) in the models. The variable-based notation usually makes the model more readable.
Moreover, a linear combination of basic functions is also a basic function. Input, output and inhibitor arcs are denoted $I, O, H[p,t] : cd(t) \to Bag(cd(p))$.

The specification of the stochastic behavior is given by associating with any transition[3] a set of functions ω_t : $cd(t) \to \mathbb{R}$ expressed in the following form:

$$\omega(t) = \begin{cases} r_1 & cond_1 \\ r_2 & cond_2 \\ \dots & \\ r_n & cond_n \\ r_n + 1 & otherwise \end{cases}$$

[3]In this paper we do not consider immediate transitions [1].

where $cond_i$ is a boolean expression comprising standard predicates on the transition color instance and predicates on the marking (defined in terms of static subclasses) so that the firing rate r_i of a transition instance can depend only on the static subclasses of the objects assigned to the transition parameters, and not on the assigned objects themselves.

For instance, if we want to model a faster movement of the infected members among the zones in the same macro-zone, then we have to define $\omega(MoveI)$ as follows:

$$\omega(MoveI) = \begin{cases} r_1 & d(x) = d(y) \\ r_2 & d(x) != d(y) \end{cases}$$

with $r_1 > r_2$ and $x, y \in var(MoveI)$. This specification of the stochastic component of the model implicitly assumes that transition firing times are instances of negative exponential distributions and that the stochastic process driving the dynamic of model is a CTMC.

Having provided these basic definitions, let us introduce the concepts of *ordinary marking* and *transition instance enabling and firing*.

DEFINITION 2.1 (Ordinary Marking). *An ordinary marking* $\boldsymbol{m}$ *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 $C_a = \{a1, a2, a3\}$ and $C_l = \{l1, l2, l3\}$ is $\boldsymbol{m} = I(\langle a1, l2\rangle + \langle a2, l2\rangle + \langle a3, l3\rangle)$ representing the state where $a1, a2$ are infected in zone $l2$ and $a3$ is infected in zone $l3$.

Moreover, we define as a *symmetric* marking an ordinary marking where the marking in each place can be expressed as a sum of tuples of static subclasses (consistent with the color domain of such a place) multiplied by a integer coefficient.

DEFINITION 2.2 (Transition Instance enabling and firing). *An instance of a given transition* t *is an assignment of actual values to the transition variables* $var(t)$*. We use the notation* $\langle t, c\rangle$ *to denote an instance, where* c *is the assignment, also called* binding.

A transition instance $\langle t, c\rangle$ *is enabled and can fire in an ordinary marking* $\boldsymbol{m}$ *iff: 1) its guard evaluated on* c *is true; 2)* $\forall p \in P, \mathrm{I}[p,t](c) \leq m(p) \cap \mathrm{H}[p,t](c) > m(p)$*, where* $\leq$ *and* $>$ *are comparison operators among multisets.*

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

For example, consider the marking $\boldsymbol{m}$ = $Outside(\langle a1\rangle, \langle a2\rangle, \langle a3\rangle)$: it enables nine instances of both, transitions *ArriveS* and *ArriveI* with $x = a1$ or $x = a2$ or $x = a3$, and $y = l1$ or $y = l2$ or $y = l3$.

The constraints on the syntax of SSNs allow the automatic exploitation of the symmetries of the model through a symbolic representation of markings.

This idea is based on the existence of *Symbolic Markings* (SMs) which are compact representations for sets 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 2.3 (Symbolic marking). *An SM $\widehat{\boldsymbol{m}}$ 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 considering C_l partitioned in $C_l^1 = \{l1, l2\}$ and $C_l^2 = \{l3\}$, the two markings $\boldsymbol{m} = I(\langle a1, l2\rangle + \langle a2, l2\rangle + \langle a3, l3\rangle)$ and $\boldsymbol{m'} = I(\langle a1, l1\rangle + \langle a2, l1\rangle + \langle a3, l3\rangle)$ belong to the same SM -where there are two infected members $(a1, a2)$ in the same zone of the first macro-zone (C_l^1) and one $(m3)$ in the second macro-zone (C_l^2)- since we may obtain $\boldsymbol{m'}$ from $\boldsymbol{m}$ by applying the permutation that exchanges the colors $l1$ and $l2$, and vice versa. Observe that instead, marking $\boldsymbol{m''} = I(\langle a1, l3\rangle + \langle a2, l3\rangle + \langle a3, l1\rangle)$ does not belong to the same SM of $\boldsymbol{m}$ and $\boldsymbol{m'}$ since it is not possible to permute $l1$ and $l3$ because they belong to different static subclasses.

Given two equivalent markings $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ (belonging to the same SM $\widehat{\boldsymbol{m}}$) it is possible to show that there is a 1:1 correspondence between the transition instances enabled in $\boldsymbol{m}_1$ and those enabled in $\boldsymbol{m}_2$. Indeed, if s is the permutation that allows one to obtain marking $\boldsymbol{m}_2$ from $\boldsymbol{m}_1$, i.e. $\boldsymbol{m}_2 = s.\boldsymbol{m}_1$, for each transition instance $\langle t, c\rangle$ enabled in $\boldsymbol{m}_1$ there exists a transition instance $\langle t, s.c\rangle$ enabled in $\boldsymbol{m}_2$ with the same rate. Moreover markings $\boldsymbol{m}_1'$ reached by firing $\langle t, c\rangle$ in $\boldsymbol{m}_1$, and $\boldsymbol{m}_2'$ reached by firing $\langle t, s.c\rangle$ in $\boldsymbol{m}_2$ are equivalent (i.e. they belong to the same SM $\widehat{\boldsymbol{m}}'$: indeed $\boldsymbol{m}_2' = s.\boldsymbol{m}_1'$). A consequence of this property is that the CTMC derived from the SSN model satisfies both the strong and exact lumpability conditions [31] with respect to the aggregation of equivalent states. If the equivalent states in the initial SM are equiprobable at time 0, then it is always true that the markings belonging to the same equivalence class remain equiprobable at any time: since in this paper we are considering only symmetric markings, then the precondition for equiprobability is always satisfied.

Let us now define the *static partition* of a place color domain: if $\widetilde{C_i}$ is the set of static subclasses of class C_i, the static partition of color domain $C_1 \times C_2 \times C3$ is $\tilde{C}_1 \times \tilde{C}_2 \times \tilde{C}3$; moreover, abusing notation, if $c = \langle c_1, c_2, c_3\rangle$ is a color in $C_1 \times C_2 \times C3$, let us denote $d(c) = \langle d(c_1), d(c_2), d(c_3)\rangle$ the static partition it belongs to.

Given a place p with color domain $cd(p)$, the average number of tokens of color $c \in cd(p)$ has the following property: $\forall c, c' : d(c) = d(c')$, $\bar{m}(p)(c) = \bar{m}(p)(c')$, i.e. the average number of colored tokens in p is the same for all colors in the same static partition element. This can be proved as follows:

$$\bar{m}(p)(c) = \sum_{m \in RS} \pi(m)\, m(p)(c) \qquad (1)$$

(where $\pi(m)$ is the probability of marking m, either at a given time t or in steady state). Let $c' : d(c) = d(c')$ be a color in the same static partition element: i.e. $c' = s.c$ for some permutation s preserving static partitions; now applying permutation s to equation (1) we obtain

$$\bar{m}(p)(s.c) = \sum_{s.m : m \in RS} \pi(s.m)\; s.m(p)(s.c) \qquad (2)$$

Due to the hypothesis of initial symmetric marking, $\{s.m | m \in RS\} = RS$, moreover $s.m$ is in the same SM as m and hence have the same probability, i.e. $\pi(m) = \pi(s.m)$, finally $s.m(p)(s.c) = m(p)(c)$ by definition of $s.m$.

An interesting feature of the SSN formalism is that it is possible to define a canonical symbolic representation for SMs, and to derive the set of reachable SMs by applying a symbolic firing rule. From the symbolic reachability graph it is possible to automatically derive the lumped Markov chain that could be constructed by exploiting the strong lumpability property of the ordinary Markov chain with respect to the state space partition into equivalence classes induced by the marking equivalence relation.

Unfolding procedure.

As we will see, the proof of correctness of our fluid approximation requires to translate an SSN model into its equivalent SPN. To explain how this is accomplished, we now briefly recall this procedure (more details can be found in [32, 40]).

The translation of an SSN model into an equivalent SPN is always possible by means of an unfolding procedure, which consists of replicating places and transitions as many times as the cardinalities of the corresponding color domains. Colors disappear in the unfolded model and the complex behavior due to color combinations, color arc functions, and color transition guards, is encoded with a net structure in which tokens are indistiguishable entities and new transitions, places, and arcs are introduced to account for the different actions performed by instances of the same transition on colored tokens. For what concerns places that contain tokens of different classes, the cross-product of the color domains are computed and each resulting element associated with the original name of the place becomes the name of a corresponding new place in the unfolded net. These place replicas are thus named $p_{c_1,\ldots,c_n}$. Similarly, for what concerns the transitions, the new names in the unfolded net start with the name of the transition in the SSN model, followed by a list expressing the binding. The possible bindings are derived accounting for the restrictions possibly imposed by the associated transition guards (when present). These transition replicas are thus named $t_{c_1,\ldots,c_n}$ where $\forall x_i \in var(t) x_i = c_i$; moreover, if an arc connecting t and p exists in the SSN model and is annotated with function $f(.)$, then $\forall c' \in f(c)$ an arc connecting $p_{c_1,\ldots,c_n}$ and $t_{c_1,\ldots,c_n}$ appears in the unfolding with weight $f(c)(c')$ (multiplicity of c' in multiset $f(c)$).

An example of unfolding is shown in Fig. 2, where we assume the color class C_1 defined as follows: $C = \{\alpha, \beta, \delta\}$. The unfolding of the SSN model in Fig. 2(left) is depicted in Fig. 2(right).

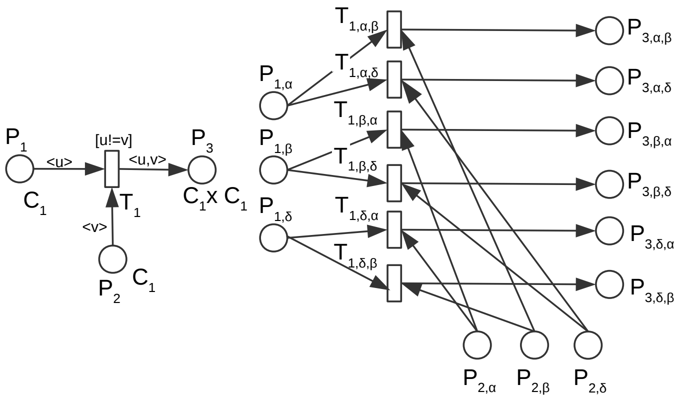


FIGURE 2. An example of unfolding.

3. FROM AN SSN MODEL TO THE CORRESPONDING ODE SYSTEM

One of the most standard approaches to analyze an SNN model consists in automatically deriving the (lumped) CTMC (which represents its underlying stochastic behavior [31]) and then performing its transient and/or steady-state analysis by analytic or numerical approaches [41]. However, there are cases in which this type of analysis is unfeasible, due to the size of the state space (i.e. state space explosion), which can be too large even when the model can be lumped (aggregated) on the basis of its symmetry properties. To cope with this problem, discrete event simulation [39, 24] or fluid approximation [42, 43, 27, 44] remain the only viable ways for studying the behavior of these type of systems.

In this paper we focus on the fluid approximation approach and we propose a fluidification method which exploits the symmetries of the models expressed with the SSN formalism to reduce the size of the system of Ordinary Differential Equations (ODE system) that can be used to obtain an approximate representation of the behavior of the model.

Before discussing such an approach, we recall how an SPN model can be evaluated by fluid approximation. Using the notation introduced in the previous Section, we can recall that an SPN is characterized by a set of functions $I, O, H[p,t] : t \to p$ that drive the evolution of the net from its initial marking $\boldsymbol{m}_0$. Given a marking $\boldsymbol{m}$, a transition t is enabled in $\boldsymbol{m}$ if $\forall p \in P, I[p,t] \leq m(p) \cap H[p,t] > m(p)$. Let $E(\boldsymbol{m})$ be the set of transitions enabled in marking $\boldsymbol{m}$, the enabling degree of transition t is defined as

$$\forall t \in E(\boldsymbol{m}) : e(t,\boldsymbol{m}) = \min_{j:I(t,p_j)\neq 0} \left\lfloor \frac{m(p_j)}{I[t,p_j]} \right\rfloor \tag{3}$$

The firing of transition t in marking $\boldsymbol{m}$ produces a new marking $\boldsymbol{m}' = \boldsymbol{m} + L[t]$, where $L[t] = O[t] - I[t]$. We can thus say that marking $\boldsymbol{m}'$ is *directly reachable* from $\boldsymbol{m}$ during the evolution of the net if $(\boldsymbol{m}' - \boldsymbol{m}) \in \Delta(\boldsymbol{m})$ where $\Delta(\boldsymbol{m}) = \{L[t] : t \in E(\boldsymbol{m})\}$. In the CTMC that underlies the behavior of an SPN, the change of marking of the SPN corresponds to the change of state which takes place with a rate that depends on the time parameter of the transition that fires. If we assume that all the transitions of the SPN use an infinite server firing policy [1] the rate of the CTMC corresponding to the firing of transition t is

$$q_{\boldsymbol{m},\boldsymbol{m}'} = \sum_{t:L[t]=(\boldsymbol{m}'-\boldsymbol{m})} \omega(t) e(t,\boldsymbol{m}) \tag{4}$$

Having recalled the relationship existing between the definition of an SPN and the corresponding CTMC, we can now proceed with the presentation of the definition and of the theorem originally derived by Kurtz [28], which we express in a form that is directly related to the definition of SPNs, providing a formal relation between the CTMC and its fluid approximation.

DEFINITION 3.1. *Let $X_\eta(v)$ be a parametric family of Markov chains with $\eta \in \mathbb{N}$, and state spaces $S_\eta \subset \mathbb{Z}^k$, such a family is called density dependent iff there exists a continuous function $f(y,l)$, $y \in \mathbb{R}^k, l \in \Delta(y)$ such that the non-diagonal entries of the infinitesimal generator corresponding to $X_\eta(v)$ can be written in the form:*

$$q_{k,k+l} = \eta f\left(\frac{k}{\eta}, l\right) \tag{5}$$

and the initial state of the chain is $\eta x_0, x_0 \in \mathbb{Z}^k$, with probability one.

Let $X(v)$ be a deterministic process denoting the solution of the ODE system

$$\frac{dX(v)}{dv} = \sum_{l \in \Delta(X(v))} f(X(v), l) \tag{6}$$

with initial condition $X(0) = x_0$, meaning that the initial value of the deterministic process $X(v)$ is equal to the initial state of the parametric family of density dependent Markov chains $X_\eta(v)$ divided by η.

In [28] Kurtz has showed that under relatively mild conditions on function f the following relation holds between the function $X(v)$ and a trajectory of the CTMC $X_\eta(v)$:

$$\forall \delta > 0 : \lim_{\eta\to\infty} P\left\{ \sup_{u \leq v} \left| \frac{1}{\eta} X_\eta(u) - X(u) \right| > \delta \right\} = 0. \tag{7}$$

Informally this means that if we consider a sequence of CTMCs with increasing initial state (i.e., the sequence of initial states is $x_0, 2x_0, 3x_0, ..., \eta x_0$) and if we assume that the infinitesimal generators corresponding to this sequence satisfy the conditions expressed by Eq. 5, then, as η becomes large, the behavior of the CTMC converges to the solution of the ODEs given in Eq. 6. In this situation, the probability of finding differences between a trajectory of the CTMC and the solution of the ODEs in a finite time horizon $(0, v)$ that are larger than a predefined arbitrarily small threshold, is zero.

Using the same reasoning presented in [29, 30] to prove that modeling interacting "populations/quantities" in process algebra leads to a family of density dependent

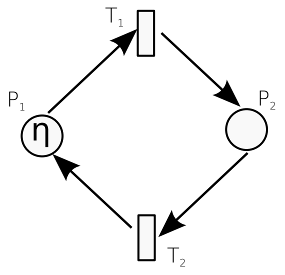


FIGURE 3. A simple SPN model used to explain fluidification.

CTMCs, then it is straightforward to show that the CTMCs constructed from the Petri nets modeling the same type of systems are density dependent too. Hence any SPN model, representing the interaction among "populations/quantities" growing to infinity in the same manner, can be approximated by ODE systems. Observe that in [42] the authors show how this class of models can be enlarged considering only some constraints on the net structural level. (i.e. deadlock trap properties).

Now we describe the ODEs which provide the fluid approximation of an SPN. Denote with x the state of the system as a vector of real numbers and with t_i an infinite server transition, which moves "fluid" tokens from a state x to a state x' with speed:

$$s(t_i,x) = \omega(t_i) \min_{j:I(p_j,t_i)\neq 0} \frac{x_j(v)}{I[p_j,t_i]}, \tag{8}$$

Such speed depends on the rate of the transition, $\omega(t_i)$, and on its *enabling degree* which is a function of the quantities of tokens present in its input places.

The number of tokens x_i in the i-th place evolves according to the following ODE:

$$\frac{dx_i(v)}{dv} = \sum_{j=1}^{m} s(t_j,x(v))(O[p_i,t_j]-I[p_i,t_j]) \tag{9}$$

so that if place p_i is an input (output) place of transition t_j then transition t_j is removing (adding) tokens from (to) place p_i according to the multiplicity given by function I (O).

The model behavior is then represented by one equation for each individual place, so that the computational complexity grows linearly with the number of places, while the size of the underlying CTMC grows exponentially.

Now, we provide an example of fluid approximation for the simple SPN model in Fig. 3 where its I and O matrices are:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad O = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and the initial marking $\boldsymbol{m}_0(P_1) = \eta$ ($\eta = 100$).

Then, from equation 9 the following ODE system can be written:

Time	CTMC	ODE	Error$_{\%}$
0.0001	99.9919	99.990001	0.0018991
0.0002	99.9838	99.980004	0.0037966
0.0011	99.8949	99.890121	0.0047840
0.0019	99.8142	99.810361	0.0038461
0.0027	99.7336	99.730728	0.0028796
0.0035	99.6532	99.651222	0.0019848
0.0116	98.8558	98.853353	0.0024753
0.0197	98.0712	98.068304	0.0029529
0.0278	97.2992	97.295871	0.0034214
0.0359	96.5397	96.535851	0.0039869
0.0849	92.1901	92.191678	0.0017116
0.134	88.2473	88.245389	0.0021655
0.183	84.6731	84.67514	0.0024092
0.2321	81.4327	81.431891	0.0009934
0.4008	72.4317	72.430531	0.0016139
0.5204	67.6567	67.658602	0.0028112

TABLE 1. Average number of tokens in the place P_1 at time t with $\eta = 100$.

$$\begin{aligned}
\frac{dx_{P_1}}{dv} &= -\omega(T_1)(\min[x_{P_1}]) + \omega(T_2)(\min[x_{P_2}]) \\
\frac{dx_{P_2}}{dv} &= +\omega(T_1)(\min[x_{P_1}]) - \omega(T_2)(\min[x_{P_2}])
\end{aligned}$$

where $x_{P_1}(0) = 1$ and $x_{P_2}(0) = 0$

In Table 1 the average number of tokens in the place P_1 at time t, computed by solving the underlying CTMC with standard methods and with the ODE system, are reported (i.e. second and third column respectively). Observe that the results computed by the ODE system reported in the third column are multiplied by η. Finally, the last column reports the percentage errors (i.e. $\frac{|v_{CTCM}-v_{ODE}|}{v_{CTCM}} * 100$) which are very small.

3.1. Fluidification of the unfolded model

The conceptually simpler way of representing the limit behavior of a SSN model with an ODE system consists of translating the SSN model into a standard SPN via an *unfolding* procedure, and then of applying the method discussed above when writing one Ordinary Differential Equation for each place of the resulting model [32]. As we will see shortly with an extensive discussion of the method applied in the case of an extremely small example, this approach yields a redundant set of ODEs which could be reduced suggesting the possibility of devising a method that may automatically produce such smaller set of ODEs.

Having described at the end of the previous sub-section how it is possible to provide a fluid approximation of an SPN, we can easily define a fluid approximation of an SSN model as follows: first the SSN model is translated into an equivalent SPN using the unfolding operator (defined in Sec. 2), then the fluid approximation discussed before is applied on the derived SPN model.

To show how this approach can be followed and to get suggestions on ways to optimize this process, let us study

$$\begin{aligned}
\frac{dx_{P_{1,\alpha}}}{dv} &= -\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}]) - \lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\delta}}]) \\
\frac{dx_{P_{1,\beta}}}{dv} &= -\lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\alpha}}]) - \lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\delta}}]) \\
\frac{dx_{P_{1,\delta}}}{dv} &= -\lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\alpha}}]) - \lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\beta}}]) \\
\\
\frac{dx_{P_{2,\alpha}}}{dv} &= -\lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\alpha}}]) - \lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\alpha}}]) \\
\frac{dx_{P_{2,\beta}}}{dv} &= -\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}]) - \lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\beta}}]) \\
\frac{dx_{P_{2,\delta}}}{dv} &= -\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\delta}}]) - \lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\delta}}]) \\
\\
\frac{dx_{P_{3,\alpha,\beta}}}{dv} &= +\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}]) \\
\frac{dx_{P_{3,\alpha,\delta}}}{dv} &= +\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\delta}}]) \\
\frac{dx_{P_{3,\beta,\alpha}}}{dv} &= +\lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\alpha}}]) \\
\frac{dx_{P_{3,\beta,\delta}}}{dv} &= +\lambda(\min[x_{P_{1,\beta}}, x_{P_{2,\delta}}]) \\
\frac{dx_{P_{3,\delta,\alpha}}}{dv} &= +\lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\alpha}}]) \\
\frac{dx_{P_{3,\delta,\beta}}}{dv} &= +\lambda(\min[x_{P_{1,\delta}}, x_{P_{2,\beta}}])
\end{aligned}$$

TABLE 2. Full set of ODEs for the model of Fig. 2.

the behavior of the simple example of Fig. 2(a), where C contains the color class $C_1 = \{\alpha, \beta, \delta\}$ with only one static subclass $C_{1,1}$.

Applying the unfolding operator on this SSN, we derive the SPN model of Fig. 2(b), where the two input places of the colored model are replaced by the six input places that account for the cardinality of color class C_1 and where the only transition of the colored model is replaced with six transitions identified by the combinations of colored tokens that the transition in the colored model may withdraw from the input places upon its firing. Transitions $T_{\alpha,\alpha}, T_{\beta,\beta}$, and $T_{\delta,\delta}$ connected to places $P_{3,\alpha,\alpha}, P_{3,\beta,\beta}$ and $P_{3,\delta,\delta}$ are not explicitly represented in the unfolded net (Fig. 2(b)), since they cannot fire due to the guard $[x! = y]$, thus implying that their output places would be constantly empty.

From the unfolded model in Fig. 2(b), and observing that $\omega(T_1) = \lambda$ and $\forall p \in P, I(p, T_{1,*,*}) \leq 1$ we can derive the system of 12 ODEs shown in Table 2 where, to avoid unneeded complexity, we use variables of the type $x_{P_{1,\alpha}}$ where we should write $x_{P_{1,\alpha}}(v)$ to explicitly express their dependence on v.

These 12 ODEs (which are organized to highlight their correspondence with the different places of the original SSN model) contain quite a lot of redundancy. Indeed, referring for a moment to the three equations that describe the dynamics of place P_1, we can observe that any permutation of the members of the static color sub-class $C_{1,1}$ applied to the elements of the first equation maps it into the other two (for instance, if we replace $\{\alpha, \beta, \delta\}$ in the first equation with $\{\beta, \alpha, \delta\}$ we obtain the second one)[4]. Moreover, if we assume that the initial marking $X(0)$ is *symmetric*, meaning that colors of a static subclass are equally numerous in all the net places[5], it is possible to see that the evolution in time of the three quantities described by the three differential equations are identical. Referring again to the first of the three equations we are considering in this moment, this has an important effect on the evaluation of the right-hand-side of this equation since it allows us to (freely) substitute $\min[x_{P_{1,\alpha}}(v), x_{P_{2,\beta}}(v)]$ with $\min[x_{P_{1,\alpha}}(v), x_{P_{2,\delta}}(v)]$, or even with $\min[x_{P_{1,\alpha}}(v), x_{P_{2,\alpha}}(v)]$. Based on this observation, we can rewrite this same equation in the following manner

$$\frac{dx_{P_{1,\alpha}}}{dv} = -\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}]) - \lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}])$$

which means

$$\frac{dx_{P_{1,\alpha}}}{dv} = -2\lambda(\min[x_{P_{1,\alpha}}, x_{P_{2,\beta}}])$$

This last result says that, as long as the permutation remains within the static subclass constraints mentioned before, the identity of the tokens belonging to a static sub-class becomes inessential. This is consistent with the symbolic marking concept discussed in the previous section, thus suggesting that a single symbolic differential equation can be written for each component of the symbolic marking to describe the temporal evolution of this representative quantity.

Recalling the same idea behind the definition of symbolic marking, we can define the notion of "symbolic" ODE: a compact representation for a set of equivalent ODE, where the actual color identity is abstracted away, but the ability to distinguish different colors and to establish their static subclass is retained.

DEFINITION 3.2 (Symbolic ODE). *A symbolic ODE (denoted $\widehat{ODE}$) is an equivalence class of ODEs, such that two ODEs are equivalent if one can be obtained from the other by applying a color permutation preserving static subclasses.*

A symbolic ODE representation can be formalized as follows:

DEFINITION 3.3 (Symbolic ODE representation). *A representation $\mathscr{R}$ of a symbolic ODE is a tuple:*

$$\mathscr{R} = \langle \widetilde{c}, \widetilde{s}, \mathscr{M} \rangle$$

where

- $\widetilde{c} : C \to \mathbb{N}$ *is a mapping from a color to the index of its corresponding color class.*

[4]A similar reasoning applies also to the equations of the other groups.

[5]If $n^i_\alpha(0)$ is the number of instances of tokens of color α in place i in the initial marking, then also the number of instances of tokens of the other colors in the same place must be the same, i.e., $n^i_\beta(0) = n^i_\delta(0) = n^i_\alpha(0)$.

- $\widetilde{s}: C \to \mathbb{N}$ *is a mapping from a color to the index of its corresponding static subclass.*
- $\mathscr{M} : \{x\} \to \{\widehat{x}\}$ *is mapping from an ODE variable to the corresponding symbolic variable in $\widehat{ODE}$ such that:*

$$\mathscr{M}(x_{p_{i,c_1,\dots,c_n}}) = \widehat{x}_{p_{i,Z^1_{\widetilde{c}(c_1),\widetilde{s}(c_1)},\dots,Z^k_{\widetilde{c}(c_n),\widetilde{s}(c_n)}}}$$

where $\forall Z^j_{\widetilde{c}(c),\widetilde{s}(c)}, Z^k_{\widetilde{c}(c'),\widetilde{s}(c')}\ k = j \Leftrightarrow c = c'$

For instance, the $\widehat{ODE}$ corresponding to the first ODE in Table 2 is:

$$\frac{d\widehat{x}_{P_{1,Z^1_{1,1}}}}{dv} = -\lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^2_{1,1}}}) - \lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^3_{1,1}}})$$

Now we can observe that symbolic variables representing the distribution of equivalent tokens (w.r.t. a colored permutation preserving static subclasses) in a place are the same so that we can substitute all these symbolic variables with a representative one. This leads automatically to a reduction in the number of $\widehat{ODEs}$.

$$\frac{d\widehat{x}_{P_{1,Z^1_{1,1}}}}{dv} = -\lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^1_{1,1}}}) - \lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^1_{1,1}}})$$

$$\frac{d\widehat{x}_{P_{2,Z^1_{1,1}}}}{dv} = -\lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^1_{1,1}}}) - \lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^1_{1,1}}})$$

$$\frac{d\widehat{x}_{P_{3,Z^1_{1,1},Z^2_{1,1}}}}{dv} = +\lambda(\min(\widehat{x}_{P_{1,Z^1_{1,1}}}, \widehat{x}_{P_{2,Z^1_{1,1}}})$$

The symbolic "representative" ODE can be directly derived by any ODE applying the algorithm in Fig. 1. It takes in input all the variables of an ODE and replaces them with the corresponding representative symbolic variables. In details, this algorithm considers the following structured (record) types:

```
typedef struct {
        string place;
        vector ⟨string⟩ color;          //vector of string
} ODE_variable;

typedef struct {
        string color_class;
        string subclass;
        int index;
} Z_var;

struct sym_ODE_variable {
        string place;
        vector ⟨Z_var⟩ sym_color;       //vector of struct Z_var
};
```

Type *ODE_variable* encodes an ODE variable $x_{p_{c_1,\dots,c_n}}$, so that the field *place* stores the corresponding place name (e.g. p) and *color* the list of the associated color (e.g. $c_1,\dots,c_n$). In the same way, type *sym_ODE_variable* encodes an $\widehat{ODEs}$ variable, so that the field *Z_var* encodes for each $Z^k_{i,j}$, their indexes i,j, and k (i.e. *color*, *subclass* and *index* respectively).

Algorithm 1 Variable translation

```
 1: function TRANSLATION(vector ⟨ODE_variable⟩ O)
      O = it is vector encoding all the variables of an ODE
      S = it is vector encoding all the symbolic variables of an $\widehat{ODEs}$
 2:     vector ⟨sym_ODE_variable⟩ S;
 3:     for (int $i = 0$; $i <$ O.size(); i++) do
 4:         S[i].Init(O[i]);
 5:         list ⟨string,int⟩ M=NULL;
 6:         int $j = 0$;
 7:         for ( int $k = 0$; $k <$ O[i].color.size(); k++) do
 8:             string C=getColor(O[i].color[k]);
 9:             string SS=getSS(O[i].color[k]);
10:             S[i].sym_color[k].color_class = C;
11:             S[i].sym_color[k].subclass = SS;
12:             int index = M.Search(O[i].color[k]);
13:             if (index == -1) then
14:                 M.Insert(O[i].color[k],j);
15:                 S[i].sym_color[k].index = j;
16:                 j++;
17:             else
18:                 S[i].sym_color[k].index = index;
19:             end if
20:         end for
21:     end for
22:     return S;
23: end function
```

For each ODE variable, in vector O, the algorithm derives the corresponding symbolic variable and stores it in S. Hence, method Init() initializes the field *place* and the size of *Z_var* according to its input ODE variable (i.e. $O[i]$). For each color of an ODE variable (i.e. vector O[i].*color*) the corresponding entry in $S[i]$.sym_color is updated. In details, functions getColor() and getSS() take in input a color and return its color class and static subclass as well. Finally, the index is defined using a counter (i.e. variable j) and the list M to insure that same colors will have same indices.

3.2. Symbolic fluidification

We now prove that the evolution in time of the quantities described by ODEs belonging to the same $\widehat{ODE}$ are identical.

The essence of the result provided by the last set of differential equations is that in the SSNs considered in this paper, if initialized with a symmetric marking, they evolve in time (and in the average) in the same manner since as shown in Section 2 the average number of tokens in a place is the same for all colors in the same static subclass. The proof of this result can be also done via "mathematical induction" expressing the left-hand-side of the differential equations as limits of their *difference quotients*.

Consider two differential equations of the type described before, both elements of the same equivalence class (i.e.

belong to the same $\widehat{ODE}$)

$$\frac{dx_{P_{i,c_1,\ldots,c_n}}(v)}{dv} = \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x_k(v)}{I[p_j,t_i]} (O[p_i,t_j]-I[p_i,t_j])$$

$$\frac{dx_{P_{i,c'_1,\ldots,c'_n}}(v)}{dv} = \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x'_k(v)}{I[p_j,t_i]} (O[p_i,t_j]-I[p_i,t_j])$$

and rewrite them introducing their difference quotients

$$x_{P_{i,c_1,\ldots,c_n}}(v+\Delta) = x_{P_{i,c_1,\ldots,c_n}}(v) + \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x_k(v)}{I[p_j,t_i]} (O[p_i,t_j]-I[p_i,t_j])$$

$$x_{P_{i,c'_1,\ldots,c'_n}}(v+\Delta) = x_{P_{i,c'_1,\ldots,c'_n}}(v) + \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x'_k(v)}{I[p_j,t_i]} (O[p_i,t_j]-I[p_i,t_j])$$

We now prove that the evolution in time of the quantities described by ODEs belonging to the same $\widehat{ODE}$ are identical.

THEOREM 3.1. *Let $dx_{P_{i,c_1,\ldots,c_n}}/dv$ and $dx_{P_{i,c'_1,\ldots,c'_n}}/dv$ be the evolution in time of the quantity of the two colored tuples $\langle c_1,\ldots,c_n\rangle$ and $\langle c'_1,\ldots,c'_n\rangle$ in place P_i defined by two ODEs belonging to the same $\widehat{ODE}$, then: $x_{P_{i,c_1,\ldots,c_n}}(v) = x_{P_{i,c'_1,\ldots,c'_n}}(v) \Rightarrow \forall c, c', \tilde{c}(c) = \tilde{c}(c') \wedge \tilde{s}(c) = \tilde{s}(c')$.*

PROOF (by mathematical induction).

For the basis, $x_{P_{i,c_1,\ldots,c_n}}(0) = x_{P_{i,c_1,\ldots,c_n}}(0)$ is verified by the *"initial marking"* assumption. Indeed, since we consider only *symmetric* initial markings (meaning that the initial marking of each place can be expressed as a weighted sum of Cartesian products of static subclasses), we can state that, initially, the colors of a static subclass are equally numerous in all the net places.

For the inductive step, let us assume that the theorem holds at time v, i.e. $x_{P_{i,c_1,\ldots,c_n}}(v) = x_{P_{i,c_1,\ldots,c_n}}(v)$; we show that it still holds at time $v+\Delta$. This implies to prove that the two following equations for the two quantities derived by Eq. 10 are identical.

$$x_{P_{i,c_1,\ldots,c_n}}(v) + \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x_k(v)}{I[p_j,t_i]} (O[p_i,t_j]-I[p_i,t_j])$$

$$x_{P_{i,c'_1,\ldots,c'_n}}(v) + \sum_{j=1}^{m} \omega(t_j) \min_{k:I[t_j,p_k]\neq 0} \frac{x'_k(v)}{I[p_j,t_i]} (O[p'_i,t_j]-I[p'_i,t_j])$$

We can observe that: $x_{P_{i,c_1,\ldots,c_n}}(v)$ and $x_{P_{i,c'_1,\ldots,c'_n}}(v)$ are identical by assumption, since they are defined by ODEs of the $\widehat{ODE}$; the $\omega(t_i)$ are identical since they depend only on the static subclass involved in the transition firing; the matrices I and O are symmetric w.r.t. the color of a static subclass by definition of SSN (i.e. I and O are identical for colors belonging to the same static subclass); $x_j(v)$ and $x'_j(v)$ are identical by assumption, since they represent quantities defined by ODEs belonging to the same $\widehat{ODE}$. Indeed, since $x_j(v)$ and $x'_j(v)$ appear respectively in the two above ODEs which belong to the same $\widehat{ODE}$ by definition, then a color permutation between them preserving static subclasses must exist meaning that these quantities are also defined by two ODEs of a same $\widehat{ODE}$. Then, the two equations are identical. □

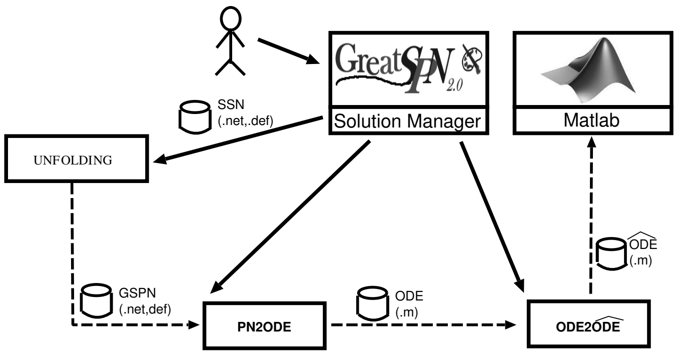


FIGURE 4. Framework architecture.

4. EXPERIMENTAL RESULTS

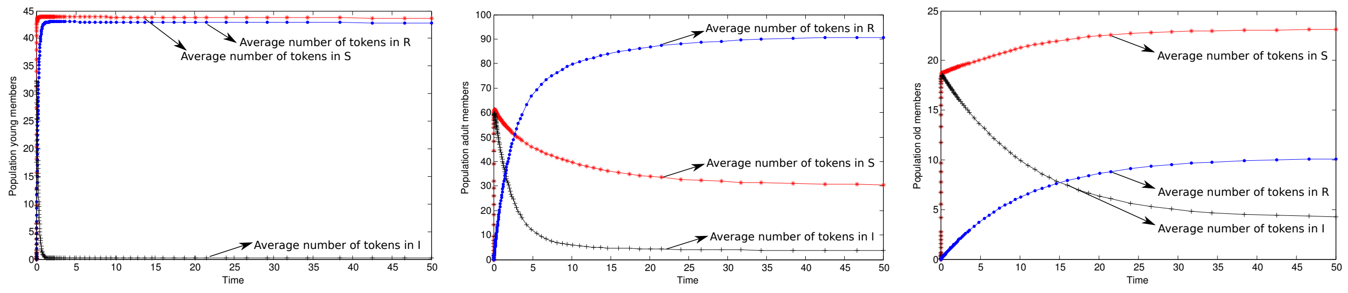
In this section we report a set of experiments on the SIR model (Fig. 1), which highlights how our approach is able to reach a high reduction factor (i.e. number of ODEs divided by number of corresponding $\widehat{ODEs}$) automatically exploiting the behavioral symmetries of this model. These experiments have been computed with a prototype implementation based on *GreatSPN* [45] for the generation of the $\widehat{ODE}$ system from an SSN model, and *Matlab* [46] for solving the differential equations.

The architecture of the prototype is depicted in Fig. 4 where framework components are shown by rectangles, component invocations by solid arrows, and models/data exchanges by dotted arrows. *GreatSPN* is used as graphical interface, for constructing the model, and as solution manager for activating the solution process. The solution manager executes in the correct order the framework components, and manages the models/data exchanges between them. The solution process comprises three steps:

1. *Unfolding* derives the unfolded GSPN model from an SNN model;
2. *PN2ODE* generates the ODE system from a GSPN model;
3. *ODE2$\widehat{ODE}$* derives $\widehat{ODE}$ system from the input ODE system.

The generated $\widehat{ODE}$ system is then solved using Matlab.

The first experiment that we are discussing is the simplest one and derives from a color specification that makes the model completely symmetric. In this first case, the color classes C_a and C_l are not partitioned into static subclasses.



(a) Epidemic diffusion in a generic zone for the young members of the population.

(b) Epidemic diffusion in a generic zone for the adult members of the population.

(c) Epidemic diffusion in a generic zone for the old members of the population.

FIGURE 6. Temporal behavior of the epidemic in the presence of population ages with different infection rates

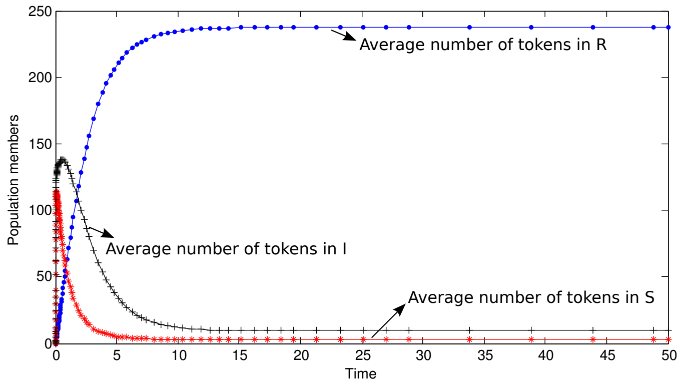


FIGURE 5. Epidemic diffusion in a generic zone.

Transition	rate
ArriveS	0.5
ArriveI	0.5
BecomeI	1.0
BecomeR	0.5
LeaveS, *LeaveR*	0.02
LeaveI	0.1
MoveS, *MoveI*, *MoveR*	1.0

TABLE 3. Transition rates.

In this way our lumping approach which corresponds to the definition of the symbolic marking allows to reach the highest reduction level; for instance, considering $|C_a| = 1$ and $|C_l| = 40$, the corresponding ODE system comprises 121 equations, while the $\widehat{ODE}$ is made of only 4 differential equations, thus yielding a reduction factor equal to 30.25).

The rates of the transitions specified in this first model are reported in Table 3. Assuming that 10,000 people are initially put in place *Outside*, the temporal behavior (between 0 to 50 time units) of the population in a generic zone is shown in Fig.5. Given the complete symmetry of the model with respect to the infection zones, due to the identical values of the rates associated with transitions *MoveS*, *MoveI*, and *MoveR*, the population members are equally distributed over the different infection zones and behave identically with respect to the dynamics that expresses the change of state within the *Susceptible*, *Infectious*, and *Recovered* classification. In this context, the infection peak is reached at time 1, while the number of infected members returns to a normal level at time 10, and then remains stable.

Before discussing the results of the second experiment, it is important to observe that the temporal behavior depicted in Fig. 5 derives from the solution of the reduced set of $\widehat{ODE}$ equations and is identical to that obtained by solving the corresponding ODE system derived from the unfolding of the model. Moreover, this same observation remains valid also for all the results obtained within the other experiments that we are discussing next.

In the second experiment we consider the population divided into three subclasses $C_a = C_a^{young} \cup C_a^{adult} \cup C_a^{old}$ to model different immune responses associated with the people's ages. Indeed, this C_a partition allows us to define parametric rates for the transitions *BecomeI* and *BecomeR* as follows:

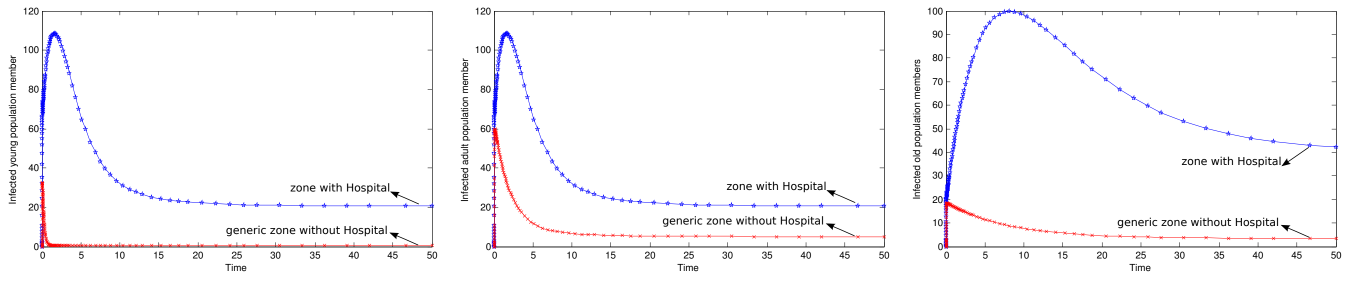
$$\omega(BecomeI) = \begin{cases} 0.01 & d(x) = C_a^{young} \\ 0.1 & d(x) = C_a^{adult} \\ 1 & d(x) = C_a^{old} \end{cases}$$

and

$$\omega(BecomeR) = \begin{cases} 5 & d(x) = C_a^{young} \\ 0.5 & d(x) = C_a^{adult} \\ 0.05 & d(x) = C_a^{old} \end{cases}$$

.

In this case, assuming that the population comprises 3,500 young members, 5,000 adults and 1,500 old subjects, and maintaining the subdivision of the infected area into 40 zones, the reduction factor remains equal to 30.25, but the numbers of ODEs and of $\widehat{ODE}$ increase to 363 and 12, respectively. Again, the dynamics of the infection on the different zones remain identical and Fig. 6 reports the temporal behaviors of the epidemic among young, adult, and old members of the population. Actually, the evolution of the epidemic is quite different for the three age classes considered in this model and this is made clear by Fig. 7 which reports the time evolution of the percentages of the infected population members for each age group. Indeed, Fig. 7 shows that old members entering an infected area become quickly ill in high percentage and later-on stabilize to a percentage level that is in any case much higher than that of adult and young members. From this experiments it is clear how the introduction of static sub-classes allows



(a) Epidemic diffusion for generic young members assuming the presence of two macro-zones.
(b) Epidemic diffusion for generic adult members assuming the presence of two macro-zones.
(c) Epidemic diffusion for generic old members assuming the presence of two macro-zones.

FIGURE 8. Temporal behavior of the epidemic in the presence of population ages with different infection rates and of a hospital.

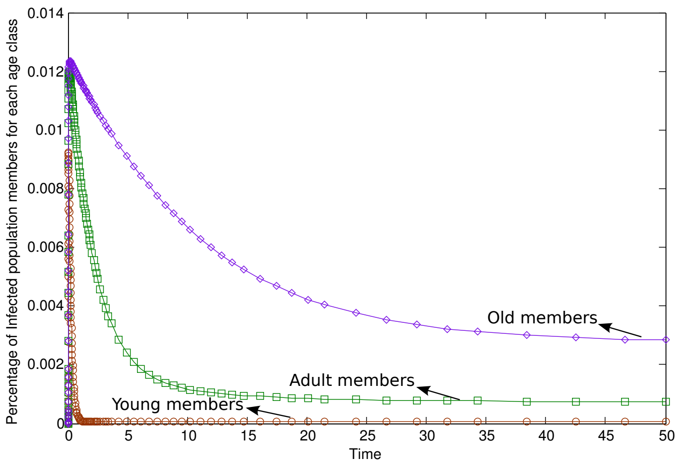


FIGURE 7. Percentage of infected population members for each considered class age.

us to "easily" consider the different infection propagation associated with the different population age.

Finally, in the last experiment we enrich our model by considering the presence of a hospital which makes one of the 40 zones considered in the previous experiments different from the others. This is reflected in the specification of our model by assuming that the infected area is sub-divided in two macro-zones so that $C_l = C_l^{MZ_1} \cup C_l^{MZ_2}$ and $|C_l^{MZ_1}| = 1$, $|C_l^{MZ_2}| = |C_l| - 1)$. With this subdivision we assume further that, when needed, infected members of the population can move into the macro-zone MZ_1 where the hospital is present to get specific additional help. This is reflected in our model by the modification of the specification of transition *MoveI* which is now controlled by a guard defined in the following manner: $[d(y) = C_l^{MZ_1}]$.

This partition of color classes reduces the symmetry of our model and thus implies a smaller reduction factor that now becomes 17.28. Indeed, the number of ODEs is still 363; while the number of $\widehat{ODE}$s becomes now 21. In Fig. 8 the temporal behavior of infected young/adult/old population members in the unique zone in MZ_1 (that where the hospital is located) is compared with the corresponding behavior in a generic zone in MZ_2 assuming the rate of the transition *MoveI* is equal to 0.04. From this plot we observe, as expected, that the average number of infected members is higher in the zone where a hospital is present. Moreover, this difference is more evident when we consider the member's age.

5. CONCLUSION AND FUTURE WORK

In this paper we have shown that models expressed with the SSN formalism and characterized by large initial markings which produce huge reachability sets, can be conveniently analyzed with the solution of a reduced set of ODEs. The possibility of representing the behavior of these types of nets with deterministic models is known as "fluidification" of the model and relies on the results expressed by Kurtz theorem which establishes the relationship existing between the solution of the system of ODEs and the average behavior of the stochastic model. A result similar to ours has already been presented in [33, 34] within the context of GPEPA [36] where replicas of identical processes are grouped together and subsequently fluidified to obtain a model which satisfies the "exact Lumpability" criteria [8, 7] and can thus be represented by a set of ODEs corresponding to the macro-states of the lumped model. The original contribution of this paper is represented by the exploitation of the symmetrical properties of SSN models to automatically construct a reduced set of ODEs that we call Symbolic Ordinary Differential Equations and which completely capture the behavior of the original model.

The power of this result is supported by the study of an epidemiological model that is represented by a relatively simple SSN, but which includes already many details that make the complexity of the analysis non-trivial. The analysis of the model is supported by a prototypical implementation that, starting from the SSN model constructed with the GreatSPN analysis tool, derives the reduced set of differential equations and prepares the specification that is subsequently passed to Mathlab for the solution of the system of ODEs. This preliminary implementation constructs also the unfolded model and the corresponding set of ODEs used for the validation of the results obtained with the reduced deterministic representation.

We are currently working at extending our approach to directly construct the reduced set of ODEs from the SSN model without performing the unfolding step: in this way symbolic ODE system shall be immediately derived from the SSN model without generating the un-reduced deterministic process.

After including this new and final step in our transformation algorithm, our future work will address the possibility of extending the method to cases in which the models exhibit only partial symmetries, and of representing these systems by means of Stochastic Ordinary Differential Equations to account for the cases in which the deterministic approximation is not suited.

Moreover, we will conduct a deeper study of the method developed by Tschaikowsky and Tribastone [33, 34] to perform a comparison among the two techniques in order to discover equivalences and differences which mighth make them better suited for the analysis of specific applications,

6. ACKNOWLEDGMENTS

We wish to thank the anonymous referees for their valuable comments that helped to improve the quality of our paper. This work has been supported in part by project "AMALFI - Advanced Methodologies for the Analysis and management of Future Internet" sponsored by Università di Torino and Compagnia di San Paolo, and by project grant Nr. 10-15-1432/HICI from the King Abdulaziz University of Saudi Arabia.

REFERENCES

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

[2] Balbo, G. (2001) Introduction to Stochastic Petri Nets. In Brinksma, E., Hermanns, H., and Katoen, J.-P. (eds.), *Formal Methods and Performance Analysis, LNCS Vol. 2090*, May, pp. 84–155. Springer-Verlag, Berlin, Germany.

[3] Molloy, M. K. (1982) Performance analysis using stochastic petri nets. *IEEE Transactions on Computers*, **31**, 913–917.

[4] Baarir, S., Beccuti, M., Dutheillet, C., Franceschinis, G., and Haddad, S. (2011) Lumping partially symmetrical stochastic models. *Performance Evaluation*, **68**, 21 – 44.

[5] Gilmore, S., Hillston, J., and Ribaudo, M. (2001) An efficient algorithm for aggregating PEPA models. *Software Engineering, IEEE Transactions*, **27**, 449–464.

[6] Sanders, W. H. and Meyer, J. F. (2006) Reduced base model construction methods for stochastic activity networks. *IEEE J.Sel. A. Commun.*, **9**, 25–36.

[7] Kemeny, J. and Snell, J. (1960) *Finite Markov Chains*. Van Nostrand, Princeton, NJ.

[8] Buchholz, P. (1993) Aggregation and reduction techniques for hierarchical GCSPNS. *Proc. of 5th International Workshop on Petri Nets and Performance Models*, Toulouse, France, oct., pp. 23–32. IEEE Computer Society.

[9] Valmari, A. (1991) Stubborn Set of Colored Petri Nets. *ICATPN 91*, Gjern, Denmark, June, pp. 102–121. IEEE Computer Society.

[10] Holzmann, G. J. (1998) An Analysis of Bitstate Hashing. *Form. Methods Syst. Des.*, **13**, 289–307.

[11] Knottenbelt, W., Harrison, P., Mestern, M., and Kritzinger, P. (2000) A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation*, **39**, 127 – 148.

[12] Ciardo, G., Zhao, Y., and Jin, X. (2012) Ten Years of Saturation: A Petri Net Perspective. *T. Petri Nets and Other Models of Concurrency*, **5**, 51–95.

[13] Hermanns, H., Kwiatkowska, M., Norman, G., Parker, D., and Siegle, M. (2003) On the use of MTBDDs for Performability Analysis and Verification of Stochastic Systems. *Journal of Logic and Algebraic Programming: Special Issue on Probabilistic Techniques for the Design and Analysis of Systems*, **56**, 23–67.

[14] Davies, I., Knottenbelt, W. J., and Kritzinger, P. S. (2002) Symbolic methods for the state space exploration of gspn models. *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02), volume 2324 of LNCS*, London, UK, April, pp. 188–199. Springer.

[15] Boudewijn, A. B., Bell, E., and Haverkort, B. R. (2001) Serial And Parallel Out-Of-Core Solution of Linear Systems arising from Generalised Stochastic Petri Nets. *In Proc. High Performance Computing 2001*, Hyderabad, India, Dec., pp. 22–26. Springer.

[16] Deavours, D. D. and Sanders, W. H. (1997) An efficient disk-based tool for solving very large Markov models. *Proc. 9th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, Saint Malo, France, June, pp. 58–71. Springer-Verlag.

[17] Kwiatkowska, M. Z. and Mehmood, R. (2002) Out-of-Core Solution of Large Linear Systems of Equations Arising from Stochastic Modelling. *Proceedings of the Second Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, London, UK, UK, July PAPM-PROBMIV '02, pp. 135–151. Springer-Verlag.

[18] Allmaier, S. C., Kowarschik, M., and Horton, G. (1997) State Space Construction and Steady-State Solution of GSPNs on a Shared-Memory Multiprocessor. *in Proc. IEEE Int. Workshop Petri Nets and Performance Models (PNPM '97*, Saint Malo, France, June, pp. 112–121. IEEE Computer Society.

[19] Caselli, S., Conte, G., and Marenzoni, P. (1995) Parallel state space exploration for GSPN models. In De Michelis, G. and Diaz, M. (eds.), *Application and Theory of Petri Nets 1995*, Lecture Notes in Computer Science, **935**, pp. 181–200. Springer Berlin Heidelberg, Turin, Italy.

[20] Coleman, J. L., Henderson, W., and Taylor, P. G. (1996) Product form equilibrium distributions and a convolution algorithm for Stochastic Petri nets. *Performance Evaluation*, **26**, 159–180.

[21] Boucherie, R. J. (1994) A characterisation of independence for competing Markov chains with applications to stochastic Petri nets. *IEEE Transactions on Software Engineering*, **20**, 536–544.

[22] Balbo, G., Bruell, S., and Sereno, M. (1994) Arrival theorems for product-form stochastic Petri Nets. *ACM SIGMETRICS*, Nashville, Tennessee (USA), May, pp. 87–97. ACM.

[23] Fishman, G. S. (1978) *Principles of Discrete Event Simulation*. John Wiley & Sons, Inc., New York, NY, USA.

[24] Gaeta, R. (1996) Efficient Discrete-Event Simulation of Colored Petri Nets. *IEEE Transactions on Software Engineering*, **22**, 629–639.

[25] Bobbio, A., Gribaudo, M., and Telek, M. (2008) Analysis of large scale interacting systems by mean field method. *Quantitative Evaluation of Systems, 2008. QEST '08. Fifth International Conference on*, Saint Malo, France, Sept., pp. 215–224. IEEE Computer Society.

[26] Bradley, J. T., Hayden, R., Knottenbelt, W. J., and Suto, T. (2008) Extracting Response Times from Fluid Analysis of Performance Models. *SIPEW'08, SPEC International Performance Evaluation Workshop, Darmstadt, 27-28 June 2008*, Darmstad, Germany, June, Lecture Notes in Computer Science, **5119**, pp. 29–43. Springer.

[27] Silva, M., Julvez, J., Mahulea, C., and Vazquez, C. (2011) On fluidization of discrete event models: observation and control of continuous Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, **21**, 497.

[28] Kurtz, T. G. (1970) Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, **1**, 49–58.

[29] Tribastone, M. (2010) Scalable differential analysis of large process algebra models. *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, Williamsburg, Virginia, USA, sept. 307. IEEE Computer Society.

[30] Tribastone, M., Gilmore, S., and Hillston, J. (2012) Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, **38**, 205–219.

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

[32] Liu, F., Heiner, M., and Yang, M. (2012) An efficient method for unfolding colored Petri nets. *Winter Simulation Conference*, Berlin, Germany, Dec. 295. IEEE Computer Society.

[33] Tschaikowski, M. and Tribastone, M. (2012) Exact fluid lumpability for markovian process algebra. *Proceedings of the 23rd International Conference on Concurrency Theory: CONCUR 2012*, Newcastle upon Tyne, UK, sept., pp. 380–394. Springer.

[34] Tschaikowski, M. and Tribastone, M. (2013) Tackling continuous state-space explosion in a markovian process algebra. *Theoretical Computer Science* , **?**, –.

[35] Hillston, J. (1996) *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA.

[36] Hayden, R. A. and Bradley, J. T. (2010) A fluid analysis framework for a markovian process algebra. *Theoretical Computer Science*, **411**, 2260 – 2297.

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

[38] Kermack, W. and McKendrick, A. (1927) A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A*, **115**, 700–721.

[39] Beccuti, M. and Franceschinis, G. (2012) Efficient simulation of stochastic well-formed nets through symmetry exploitation. *Proceedings of the Winter Simulation Conference*, Los Alamitos, California, USA, Dec. WSC '12, pp. 296:1–296:13. IEEE Computer Society.

[40] Teruel, E., Franceschinis, G., and Silva, M. (1998) Untimed Petri Nets. In Balbo, G. and Silva, M. (eds.), *Performance Models for Discrete Event Systems with Synchronisations: Formalisms and Analysis Techniques*, pp. 27–75. Editorial KRONOS, Zaragoza, Spain.

[41] Stewart, W. J. (1995) *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, New Jersey, USA.

[42] Heiner, M., Mahulea, C., and Silva, M. (2010) On the Importance of the Deadlock Trap Property for Monotonic Liveness. *Int. Workshop on Biological Processes and Petri Nets (BioPPN), A satellite event of Petri Nets 2010*, Braga, Portugal, June, pp. 39–54.

[43] Hillston, J. (2005) Fluid flow approximation of PEPA models. *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, pp. 33–42.

[44] Silva, M. and Recalde, L. (2004) On fluidification of Petri Nets: from discrete to hybrid and continuous models. *Annual Reviews in Control*, **28**, 253 – 266.

[45] Babar, J., Beccuti, M., Donatelli, S., and Miner, A. S. (2010) Greatspn enhanced with decision diagram data structures. *Proceedings of Applications and Theory of Petri Nets, 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21-25,*, Los Alamitos, California, USA, June, pp. 308–317. IEEE Computer Society.

[46] MATLAB webpage. `http://www.mathworks.com/`.