A Tool for the Automatic Derivation of Symbolic ODE from Symmetric Net Models

Marco Beccuti
Dipartimento di Informatica
Università di Torino
Torino, Italy
beccuti@di.unito.it

Lorenzo Capra
Dipartimento di Informatica
Università di Milano
Milano, Italy
capra@di.unimi.it

Massimiliano De Pierro
Dipartimento di Informatica
Università di Torino
Torino, Italy
depierro@di.unito.it

Giuliana Franceschinis
Dipartimento di Scienze e Innovazione Tecnologica
Università del Piemonte Orientale
Alessandria, Italy
giuliana.franceschinis@uniupo.it

Laura Follia
Dipartimento di Informatica
Università di Torino
Torino, Italy
follia@di.unito.it

Simone Pernice
Dipartimento di Informatica
Università di Torino
Torino, Italy
pernice@di.unito.it

***Abstract*—High-level Petri nets (HLPN) are an expressive formalism well supported by a number of tools that automate the editing and the interactive simulation of models and some kinds of analytical techniques, mainly based on state-space exploration. Structural analysis of HLPN is, however, a challenging task not yet adequately supported and it is often accomplished via the unfolding of an HLPN into a corresponding low-level Petri Net. An approach to derive a system of Ordinary Differential Equations (ODE) from a Stochastic Symmetric Net (SSN) has been proposed a few years ago, based on the net's unfolding and subsequent grouping of similar equations. This method has been recently improved by providing an algorithm that directly derives a compact ODE system (from a partially unfolded net) in a symbolic way, through algebraic manipulation of SSN annotations. In this paper, we present the automation of the calculus of Symbolic ODE (SODE) for SSN models as a new module of SNexpression, a recently developed tool for the symbolic structural analysis of Symmetric Nets. An application of the tool/technique to a variant of a SIRS epidemic model including antibiotic resistance is also described.**

***Index Terms*—High-Level Petri Nets, Symmetric Nets, Symbolic structural relations, Ordinary Differential Equations.**

I. Introduction

Symmetric Nets (SN) and their extension with stochastic timing[1], Stochastic SNs (SSNs) [11], are high-level Petri Net formalisms featuring a particular syntax for places, transitions, and arc annotations: such syntax has been devised to make the symmetries present in the modeled system's structure and behavior explicit. This formalism is thus convenient from the point of view of model representation as well as from that of its analysis. Efficient methods have been proposed to perform SSN state space based analysis [11], structural analysis [6], [8], [12], and analysis based on fluid approximations [3]. Many of these algorithms have been implemented in GreatSPN [1], whereas the most recent developments on structural analysis have been implemented in SNexpression (www.di.unito.it/~depierro/SNex), the tool described in this paper. A first version of this tool has been presented in [7]. In the last few years it has been extended with several features comprising the support to the theoretical advances in the symbolic calculus ([6], [9], [12]), and the algorithms to compute specific SN properties [8]. This paper focuses on a recent extension of the tool, an algorithm to derive the ODE of a SSN model in a symbolic way, avoiding the net unfolding.

In the literature there are a few examples of similar approaches, using symbolic manipulation of High-Level Petri Net arc expressions [14]: Extended Conflicts Sets computation is discussed in [13], but the method is applicable only to Unary Regular Nets; behavior preserving model reduction for Quasi Well-formed nets is proposed in [15]; symbolic Stubborn Sets computation is developed in [16]. However, to the best of our knowledge, the methods have not been implemented in a tool available to the research community; in particular we are not aware of any automatic tool capable of producing a set of symbolic ODE directly from the HLPN model.

Paper structure The paper is organized as follows: in the rest of this Section the SSN formalism is briefly introduced through an example; Section II provides a brief overview of the language for expressing SSN structural properties in symbolic form, that is the language on which SNexpression builds, and on the system of ODE that can be derived from a SSN. Section III defines the partial unfolding procedure. Section IV describes the specific functions required to automatically generate a reduced set of ODE that represent the whole behavior of the system, and illustrates a new method, improved w.r.t. that presented in [3], to derive such equations without completely unfolding the model. Section V describes the main functions available through a command line language recognized by the SNexpression CLI and the library implementing the calculus. In that section, the benefits of the method and its implementation are discussed with the help of the running example. Finally, Section VI outlines ongoing and foreseen

[1]Introduced with the name of Well-formed Nets, later renamed SNs.

future developments.

A. A primer on SSNs and introduction to the example model

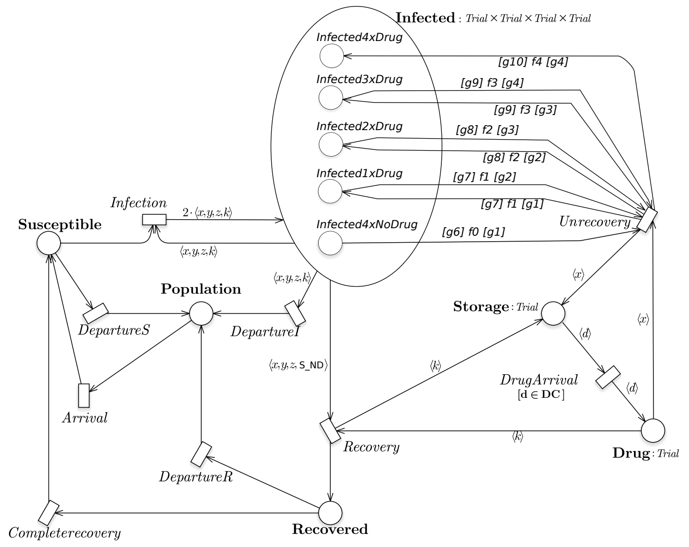
Let us now briefly introduce the SSN formalism and the language on which the calculus implemented by the SNexpression tool is based (the formal definition can be found in the Appendix). An SSN is a bipartite graph whose nodes are of two types, *places* and *transitions*. A place p denotes a system's local state and may hold tokens belonging to the *place color domain* $cd(p)$, defined as the Cartesian product of finite *basic color classes*; the multiset m of tokens stored in a place is called the *place marking*. A transition t represents an activity and its completion (called *firing*) represents an event modifying the system's state; each transition is parameterized with respect to a (arbitrarily ordered) set $var(t)$ of local variables, each one taking values on a color class: the set of possible assignments of t variables (which are elements of the Cartesian product of the variable's color classes) define the *transition color domain* $cd(t)$. For example, t with $var(t) = \{x, y, z\}$, $x, y : C$ and $z : C'$, is a transition with three parameters, x, y, z, taking values on sets C (x, y) and C' (z), respectively, hence $cd(t) = C \times C \times C'$. If there are n variables of the same type C, an indexed family of variable symbols $\{v_i\}$, $i : 1, .., n$, may be used: in the example above, we might use $\{x_1, x_2, z\}$, with $x_i : C$, $z : C'$.

An assignment of values to $var(t)$ is said a *transition instance*: in the example, x=c_1, y=c_2, z=c', with $c_i \in C$, $c' \in C'$, is an instance of t, also denoted as a tuple $\langle c_1, c_2, c' \rangle \in cd(t)$ (this tuple is the transition instance's color)[2]. An explicit notation may be used, i.e., $(t, \langle c_1, c_2, c' \rangle)$. A transition t may have an associated *guard* $g(t)$, i.e., a predicate defined on $cd(t)$ (the default/implicit guard being the constant $true$): in that case the transition color domain $cd(t)$ is restricted to those instances of t verifying the guard.

Places and transitions are connected by directed, annotated arcs: the *expression* annotating an arc between t and p is a function of t's variables mapping every instance of t to a multiset[3] defined on $cd(p)$. The state (or marking) of an SSN is defined by the marking of all its places. An instance (t, c), with $c \in cd(t)$, is *enabled* in a marking m if each *input place* of t (a place connected to t by an arc pointing to t) holds at least as many tokens as the multiset yielded by the evaluation of the corresponding arc expression on color c. A state change may occur when an enabled instance *fires*, withdrawing a multiset of coloured tokens from each input place and adding a multiset of coloured tokens to each *output place*, according to the evaluation of the corresponding arc expressions on color c. The symbols $^\bullet t$, $t^\bullet$ denote the set of input/output places of t, respectively. The expressions on input/output arcs are denoted $I[p, t]$, $O[p, t]$.

[2]If $var(t) = \{\}$ then by convention $cd(t) = E = \{e\}$, E being the neutral color class

[3]A multiset on set D is a map $D \to \mathbb{N}$. $Bag[D]$ denotes the set of all multisets on D. If $m \in Bag[D]$, and $d \in D$, $m[d]$ is the multiplicity of d in m. The inclusion relation $\sqsubseteq$ is defined as: $m' \sqsubseteq m$ iff $\forall_d m'[d] \leq m[d]$.



Color class definition: $\text{Trial} = \text{ND} \cup \text{DC}$, $|\text{ND}| = 1, |\text{DC}| = 5$

Trans	guard	rate
Arrival	*true*	6e−8
CompleteRecovery	*true*	0.025
DepartureR	*true*	3.9e−8
DepartureS	*true*	3.9e−8
Infection	$g_1 \vee g_2 \vee g_3 \vee g_4 \vee g_5$	4.89e−7
DepartureI	$g_1 \vee g_2 \vee g_3 \vee g_4 \vee g_5$	3.9e−8
DrugArrival	$[d \in \text{DC}]$	1e−3
Unrecovery	$g_1 \vee g_2 \vee g_3 \vee g_4$	2e−5
Recovery	$g'_1 \vee g'_2 \vee g'_3 \vee g'_4$	1.98e−3
$g_6 = d_1 \in \text{ND} \wedge d_2 \in \text{ND} \wedge d_3 \in \text{ND} \wedge d_4 \in \text{ND}$		
$f_0 = \langle S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}} \rangle$		
$g_1 = x \in \text{DC} \wedge y \in \text{ND} \wedge z \in \text{ND} \wedge k \in \text{ND}$		
$g_7 = d_1 \in \text{DC} \wedge d_2 \in \text{ND} \wedge d_3 \in \text{ND} \wedge d_4 \in \text{ND}$		
$f_1 = \langle x, S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}} \rangle$		
$g_2 = x \in \text{DC} \wedge y \in \text{DC} \wedge z \in \text{ND} \wedge k \in \text{ND} \wedge x \neq y$		
$g_8 = d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{ND} \wedge d_4 \in \text{ND}$		
$f_2 = \langle x, y, S_{\text{ND}}, S_{\text{ND}} \rangle$		
$g_3 = x \in \text{DC} \wedge y \in \text{DC} \wedge z \in \text{DC} \wedge k \in \text{ND} \wedge$ $\wedge x \neq y \wedge x \neq z \wedge y \neq z$		
$g_9 = d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}$		
$f_3 = \langle x, y, z, S_{\text{ND}} \rangle$		
$g_4 = x \in \text{DC} \wedge y \in \text{DC} \wedge z \in \text{DC} \wedge k \in \text{DC} \wedge$ $\wedge x \neq y \wedge x \neq z \wedge x \neq k \wedge y \neq z \wedge y \neq k \wedge z \neq k$		
$g_{10} = d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{DC}$		
$f_4 = \langle x, y, z, k \rangle$		
$g_5 = x \in \text{ND} \wedge y \in \text{ND} \wedge z \in \text{ND} \wedge k \in \text{ND}$		
$g'_1 = x \in \text{ND} \wedge y \in \text{ND} \wedge z \in \text{ND} \wedge k \in \text{DC}$		
$g'_2 = x \in \text{DC} \wedge y \in \text{ND} \wedge z \in \text{ND} \wedge k \in \text{DC} \wedge x \neq y$		
$g'_3 = x \in \text{DC} \wedge y \in \text{DC} \wedge z \in \text{ND} \wedge k \in \text{DC} \wedge$ $\wedge x \neq y \wedge x \neq k \wedge y \neq k$		
$g'_4 = x \in \text{DC} \wedge y \in \text{DC} \wedge z \in \text{DC} \wedge k \in \text{DC} \wedge$ $\wedge x \neq y \wedge x \neq z \wedge y \neq z \wedge x \neq k \wedge y \neq k \wedge z \neq k$		

Fig. 1: A SIRS model with antibiotic resistance.

In Fig. 1 the SSN model used throughout the paper is depicted: it represents a Susceptible-Infected-Recovered-Susceptible (SIRS) model where members of a population (place *Population*) may join a community (transition *Arrival*) comprising infected individuals (place *Infected*) and become susceptible (place *Susceptible*), then by meeting infected community members the newcomers can get infected in turn (transition *Infection*). Place *Infected* is represented as a big ellipse containing five places: this will be used later to illustrate the partial unfolding procedure. For the moment let us just consider the big place as a whole, disregarding the enclosed places and the annotations on the arcs incident on them described on the table.

Infected individuals are treated wit a drug (e.g., an antibiotic is prescribed, possibly after an antibiogram analysis); the treatment may not be effective for various reasons, e.g., the antibiotic did not reach the infected tissue with enough concentration, or the patient did not accurately follow the prescription. Bacteria may develop resistance to an antibiotic as a result of an unsuccessful treatment, and this is modeled by recording in the token colors of place *Infected* the types of drugs (up to four) that did not lead to a full recovery. Transition *Unrecovery* represents the occurrence of unsuccessful treatment leading to a higher degree of resistance of the involved bacterium, up to a maximum level which cannot be treated with any drug. In the low and intermediate resistance levels a successful treatment may occur, represented by transition *Recovery*, leading to a complete recovery (passing through place *Recovered* and then back to the *Susceptible* state). Place *Drug* represents the available drugs that can be used for treatment. Periodically, a new supply of drugs is brought to the pharmacy from a *Storage* (transition *DrugArrival*).

The place color domains of this example are built of a single basic color class, $Trial$, composed of two *static subclasses*: ND ($= \{nd\}$), and DC, of cardinality $n \geq 4$ (in the experiments $n = 5$). Place *Infected* has color domain $Trial \times Trial \times Trial \times Trial$, and thus contains 4-tuples of elements of class $Trial$ representing the achieved antibiotic resistance of the bacterium of an infected individual. Places *Drug* and *Storage* have color domain $Trial$, and actually hold only tokens with colour in subclass DC. The other places have a "neutral" color domain (i.e., they contain "black" tokens) composed, by convention, of the neutral color class $E = \{e\}$. The initial marking we are considering has 890000 black tokens in place *Susceptible*, 50 tokens with color $\langle nd, nd, nd, nd\rangle$ in place *Infected*, 100 tokens of color $\langle drug_i\rangle$ for all $drug_i \in$ DC, 9950 black tokens in place *Population*. The arc expressions are rather simple: syntactically they appear as tuples (1-tuple $\langle x\rangle$ and 4-tuples $\langle x, y, z, k\rangle$, or $\langle x, S_{\mathrm{ND}}, S_{\mathrm{ND}}, S_{\mathrm{ND}}\rangle$) whose elements are projections (taking the form of variables as x, y, z, k, d) or constant functions (e.g., S_{ND}).

Let us consider the arcs connecting transition *Unrecovery* to the places enclosed by *Infected*, which result from a partial unfolding procedure that will be explained later: in the original net there are only one input and one output arc, annotated by $I[Infected, Unrecovery] = f_0[g_1] + f_1[g_2] + f_2[g_3] + f_3[g_4]$ and $O[Infected, Unrecovery] = f_1[g_1] + f_2[g_2] + f_3[g_3] + f_4[g_4]$, respectively, where the definition of f_i and g_i are given in Fig. 1. A term like $f_i[g_j]$ evaluates to $f_i(c)$ if $g_i(c) = true$, otherwise it results in $\emptyset$.

The transition color domains are $Trial$ (*DrugArrival*, with parameter d); $Trial \times Trial \times Trial \times Trial$ (*Unrecovery*, *Infection*, *Recovery*, *DepartureI*); E (*Arrival*, *DepartureS*, *DepartureR*, *Completerecovery*).

Each transition has an associated *guard*, defining its valid instances, and a *rate* parameter ($w(trans.instance)$) that concurs to the definition of the transition firing speed (also called *transition intensity*): the transition guards and rates for this model are listed on the table included in Fig. 1. Guards involve basic predicates restricting the possible values of variables to a given subset of colors, or enforcing a given pair of variables to take the same, or different, values.

Transition intensities (characterizing the stochastic delay between transition enabling and firing) are derived from the transition rate and the marking according to one among several possible *service semantics*: in this paper we shall consider *infinite server semantics* (IS) and a semantics based on the *mass action law* (MA). The transition intensity of (t, c) in marking m according to the IS semantics is defined as the product of the rate $w(t, c)$ and the enabling degree of (t, c) in m ($ed(t, c, m)$ is the largest value k satisfying $\forall_{p \in {}^\bullet t} m(p) \sqsupseteq kI[p, t]$). When the MA semantics is used, the transition intensity is given by the product of $w(t, c)$ and $\prod_{p \in {}^\bullet t} \prod_{c' \in I[p,t](c)} m(p)[c]^{I[p,t](c)[c']}$.

Both arc functions and guards must be expressed with a syntax that allows symmetries to be exploited: this means that if a transition instance (t, c) is enabled in marking m, then a similar instance c' obtained by applying a permutation[4] of colors on c is enabled in marking m' obtained from m by applying the same permutation of colors, whose firing produces a similar reachable marking, up to the same permutation of colors. This symmetric behavior is the key to symbolic structural analysis. Deriving the structural properties of SSN models in a symbolic way is the main function of the SNexpression tool; exploiting this feature to enhance the efficiency of other solution methods is another important application of the tool: in this paper symbolic structural properties are exploited to derive a compact system of ODE for the transient analysis of average measures on SSN models.

A SSN with an initial marking is an executable model, it can be simulated or its state space can be generated and analysed to compute interesting performance indices. When the state space is very large, provided that suitable hypothesis on the model are verified [2], the average marking of each place p at time t can be evaluated, without exploring the state space, by generating and solving a system of Ordinary Differential Equations (ODE). A procedure to automatically obtain the System of ODE from a Stochastic Petri Net (SPN) model has been implemented in GreatSPN: this can be applied to a SSN after its *unfolding*, however this may lead to a very large number of ODE (as many as the number of places in the unfolded SPN model), including a large number of terms (proportional to the number of arcs in the model).

II. Symbolic manipulation of SSN arc functions

In this section some background concepts required to explain the new method for deriving the set of Symbolic ODE (SODE) from a SSN are introduced. The method builds upon the ability to solve in a symbolic way expressions whose terms

[4]The permutations must preserve the partition into static subclasses.

are the elements of a language $\mathcal{L}$, and involving a specific set of functional operators (in this context, the *difference* and the *transpose*). The elements of $\mathcal{L}$ are an extension of the SN arc functions, and have got the following syntax:

$$\sum_j \lambda_j [g'_j] T_j [g_j],\ \lambda_j \in \mathbb{N}$$

where T_j is a tuple of class-functions while $[g_j]$ and $[g'_j]$ are called respectively *guard* and *filter*. These expressions denote functions $D \to Bag[D']$; D and D' are in turn defined as Cartesian products of color classes. The components in a tuple T_j are in one-to-one correspondence with the elements in the Cartesian product D': they are *intersections* ($\cap$) of basic class functions $D \to Bag[C]$ from set $BS = \{v, S - v, S_C, S_{C_k}\}$, where C is one of the basic color classes in D', v is a variable of type C, and C_k is a static subclass of C. When class C occurs n times in D' a family $\{v_i\}$, $i : 1,..,n$, of type C variables may be used to highlight that v_i (also called *projection*) refers the i^{th} type C element on a tuple of D'. Symbols S, S_{C_k} are constants mapping to a whole (sub)class. To keep the presentation simple, ordered classes are not considered here, but the presented results extend to models including them. The functions in BS are a subset of SN class functions (see the Appendix), however, any SN class function can be expressed[5] in terms of BS. A function tuple T_j, individually considered, maps to multisets with multiplicities ≤ 1 (i.e., *sets*) of colors. Filters and guards are boolean expressions whose terms are basic predicates: variable symbols occurring on the guard g_j and on the tuple T_j have exactly the same meaning, whereas a variable $v_i : C$ on g'_j refers the i^{th} type C element of any color tuple represented by T_j.

Language $\mathcal{L}$ is closed with respect the *transpose* and the *difference* operators (formally defined in the Appendix); SNexpression implements the rules for symbolically treating these operators.

The language, with its operators and properties, allows the SODE characterizing an SSN model to be defined without net's unfolding. In particular, the difference and transpose operators (the latter is denoted ^{t}) allow the relations $\mathcal{R}$ and $\mathcal{A}$ to be defined and expressed in a symbolic form. Assuming that nodes p and t are connected, function $\mathcal{R}(t,p)$, called *Removed By*, defines which instances (t, c') withdraw tokens of color $c \in cd(p)$ from place p. Function $\mathcal{A}(t,p)$, called *Added By*, defines which instances (t, c') put tokens of color $c \in cd(p)$ into p. Both functions map to multisets: $\mathcal{R}(t,p)(c)[c']$ and $\mathcal{A}(t,p)(c)[c']$ are the number of tokens of color c withdrawn/added by an instance (t, c') from/to p.

$$\mathcal{R}(t,p) : cd(p) \to Bag[cd(t)];\ \mathcal{R}(t,p) = (I[t,p] - O[t,p])^t$$

$$\mathcal{A}(t,p) : cd(p) \to Bag[cd(t)];\ \mathcal{A}(t,p) = (O[t,p] - I[t,p])^t$$

Let us show a few examples on the SIRS SSN depicted in Fig. 1. The table below refers to places $Infected3\text{x}Drug$ and *Susceptible*, and contains the expressions of relations $\mathcal{A}$ and $\mathcal{R}$ for all transitions they are connected to. These symbolic expressions are the basis for the construction of the ODE associated with place $Infected3\text{x}Drug$, which contains positive terms corresponding to $\mathcal{A}$, and negative terms corresponding to $\mathcal{R}$, representing the flow of tokens into and out of the place, respectively. Analogously for place *Susceptible*.

$\mathcal{A}(Infection, Infected3\text{x}Drug) =$ $1\langle d_1, d_2, d_3, d_4\rangle[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge$ $d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}]$
$\mathcal{R}(Recovery, Infected3\text{x}Drug) =$ $1\langle d_1, d_2, d_3, S - d_1 \cap S - d_2 \cap S - d_3 \cap S_{\text{DC}}\rangle$ $[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge d_1 \in \text{DC} \wedge$ $d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}]$
$\mathcal{A}(Unrecovery, Infected3\text{x}Drug) =$ $1\langle d_1, d_2, d_3, S_{\text{ND}}\rangle[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge$ $d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}]$
$\mathcal{R}(Unrecovery, Infected3\text{x}Drug) =$ $1\langle d_1, d_2, d_3, S - d_1 \cap S - d_2 \cap S - d_3 \cap S_{\text{DC}}\rangle$ $[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge d_1 \in \text{DC} \wedge$ $d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}]$
$\mathcal{R}(DepartureI, Infected3\text{x}Drug) =$ $1\langle d_1, d_2, d_3, d_4\rangle[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge$ $d_1 \in \text{DC} \wedge d_2 \in \text{DC} \wedge d_3 \in \text{DC} \wedge d_4 \in \text{ND}]$
$\mathcal{R}(Infection, Susceptible) =$ $1\langle S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}}\rangle + 1\langle S_{\text{DC}}, S_{\text{ND}}, S_{\text{ND}}, S_{\text{ND}}\rangle$ $+1[d_1 \neq d_2]\langle S_{\text{DC}}, S_{\text{DC}}, S_{\text{ND}}, S_{\text{ND}}\rangle$ $+1[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3]\langle S_{\text{DC}}, S_{\text{DC}}, S_{\text{DC}}, S_{\text{ND}}\rangle$ $+1[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_1 \neq d_4 \wedge d_2 \neq d_3 \wedge d_2 \neq d_4 \wedge d_3 \neq d_4]$ $\langle S_{\text{DC}}, S_{\text{DC}}, S_{\text{DC}}, S_{\text{DC}}\rangle$
$\mathcal{A}(Arrival, Susceptible) = 1\langle S_E\rangle$
$\mathcal{A}(CompleteRecovery, Susceptible) = 1\langle S_E\rangle$
$\mathcal{R}(DepartureS, Susceptible) = 1\langle S_E\rangle$

Place $Infected3\text{x}Drug$ comes from the partial unfolding of $Infected$: this replica holds four-tuples of colors whose first three are of type DC and all different, and the fourth one is of type ND (the singleton $\{nd\}$). This is clearly indicated by the guards of all corresponding $\mathcal{A}$ and $\mathcal{R}$ terms. The symbolic expression of $\mathcal{A}(Infection, Infected3\text{x}Drug)$ is a template of those instances of $Infection$ putting tokens of color $\langle d_1, d_2, d_3, d_4\rangle$ into the place[6]: this simple template indicates that (only) a transition instance $x = d_1$, $y = d_2$, $z = d_3$, $k = d_4$ puts exactly *one* (according to the weight of the unique term of $\mathcal{A}$) such a token into the place. Observe that despite there is input arc from $Infected$ to $Infection$ (which after the partial unfolding is replicated for each of the five unfolded places, including $Infected3\text{x}Drug$), $\mathcal{R}(Infection, Infected) = \emptyset$.

The expression of $\mathcal{R}(Recovery, Infected3\text{x}Drug)$ is a bit more complex: it represents those instances of $Recovery$ withdrawing tokens of color $\langle d_1, d_2, d_3, d_4\rangle$ from $Infected3\text{x}Drug$. The expression's unique term means that an instance $x = d_1$, $y = d_2$, $z = d_3$, and with k bound to *any* color (of type DC) other than d_1, d_2, d_3, removes exactly *one* such token from the place. According to the term cardinality,

[5]This can be obtained through the intersection, which is not included in the SSN arc expressions syntax: see the Appendix for the details.

[6]Recall that $cd(Infected) = Trial \times Trial \times Trial \times Trial$, hence symbol d_i denotes the value of the i^{th} element in the four-tuple associated with each token on the place.

there are $|\mathrm{DC}|-3$ such instances.

The symbolic expression of $\mathcal{R}(Infection, Susceptible)$ is built of five terms. As a whole, it represents which instances of $Infection$ may remove a *black token* from place $Susceptible$: they are 1) those with $x \in \mathrm{ND}$, $y \in \mathrm{ND}$, $w \in \mathrm{ND}$, $k \in \mathrm{ND}$, which corresponds to $x = y = z = k = nd$, 2) those with $\forall x \in \mathrm{DC}$, $y \in \mathrm{ND}$, $w \in \mathrm{ND}$, $k \in \mathrm{ND}$ (there are $|\mathrm{DC}|$ such instances), 3) those with $\forall x \in \mathrm{DC}$, $\forall y \in \mathrm{DC}$, $x \neq y$, $w \in \mathrm{ND}$, $k \in \mathrm{ND}$ (there are $|\mathrm{DC}|(|\mathrm{DC}|-1)$ such instances), and so forth. In this case *filters* appear in the expression, allowing specific color instances to be selected from the (parametric) set represented by the following tuple (e.g., those satisfying $x \neq y$ in the third term): as said, a variable symbol d_i in a filter explicitly refers the i^{th} element of a given type (in our example, coinciding with the i^{th} position) in the tuple [7].

A. The system of ODE

The approach for deriving the SODE for a given place p of a (partially unfolded) SN requires to compute $\mathcal{R}(t,p)$ for each transition t such that $p \in {}^\bullet t$, and $\mathcal{A}(t,p)$ for each t, $p \in t^\bullet$. Let $\mathbf{x}[p,c]$ be the (average) number of c-colored tokens in place p at time ν (to keep notation simpler we will omit time dependency). The differential equation for place p and color $c \in cd(p)$ is defined (in a non symbolic way) as:

$$\frac{d\mathbf{x}[p,c]}{d\nu} = \sum_{(t,c'):p\in t^\bullet, c'\in\mathcal{A}(t,p)(c)} \varphi(\mathbf{x}(\nu),t,c')\cdot\mathcal{A}(t,p)(c)[c'] - \sum_{(t,c'):p\in {}^\bullet t, c'\in\mathcal{R}(t,p)(c)} \varphi(\mathbf{x}(\nu),t,c')\cdot\mathcal{R}(t,p)(c)[c'] \quad (1)$$

Each sum spans over all instances (t,c') that add (positive terms) or withdraw (negative terms) tokens of color c to/from p. The *intensity* φ of (t,c') -which depends on the distribution of colored tokens in t's preset- multiplied by the number of tokens of color c added to or withdrawn from p by (t,c') ($\mathcal{A}(t,p)(c)[c']$ or $\mathcal{R}(t,p)(c)[c']$) gives the actual flow of tokens into or out of p.

Due to SN symmetries, the above procedure can be done for just an *arbitrary* color $c \in cd(p)$. The two summations over the color domain of t in (1) may be compressed by folding instances with "similar" intensity and moving the same number of tokens in, or out of, place p. This is directly achieved by expressing the ODE in a *symbolic* way.

Let us show how the ODE looks like for two places in the SIRS model: let us consider place *Susceptible* and place $Infected3\mathrm{x}Drug$, for which the $\mathcal{A}$ ad $\mathcal{R}$ have been presented earlier in this section.

$$\frac{d\mathbf{x}[\mathrm{Susceptible},e]}{d\nu} = +\varphi(\mathbf{x},Arrival,e)\cdot 1$$
$$+\varphi(\mathbf{x},CompleteRecovery,e)\cdot 1 - \varphi(\mathbf{x},Departure,e)\cdot 1$$
$$-\textstyle\sum_{\langle d_1,d_2,d_3,d_4\rangle\in D_0}\varphi(\mathbf{x},Infection,\langle d_1,d_2,d_3,d_4\rangle)\cdot 1$$
$$-\textstyle\sum_{\langle d_1,d_2,d_3,d_4\rangle\in D_1}\varphi(\mathbf{x},Infection,\langle d_1,d_2,d_3,d_4\rangle)\cdot 1$$
$$-\textstyle\sum_{\langle d_1,d_2,d_3,d_4\rangle\in [g_2]D_2}\varphi(\mathbf{x},Infection,\langle d_1,d_2,d_3,d_4\rangle)\cdot 1$$
$$-\textstyle\sum_{\langle d_1,d_2,d_3,d_4\rangle\in [g_3]D_3}\varphi(\mathbf{x},Infection,\langle d_1,d_2,d_3,d_4\rangle)\cdot 1$$
$$-\textstyle\sum_{\langle d_1,d_2,d_3,d_4\rangle\in [g_4]D_4}\varphi(\mathbf{x},Infection,\langle d_1,d_2,d_3,d_4\rangle)\cdot 1$$

where D_i are domains and g_i are filters:

$D_0 = \mathrm{ND}\times\mathrm{ND}\times\mathrm{ND}\times\mathrm{ND}$
$D_1 = \mathrm{DC}\times\mathrm{ND}\times\mathrm{ND}\times\mathrm{ND}$
$g_2 = d_1 \neq d_2$
$D_2 = \mathrm{DC}\times\mathrm{DC}\times\mathrm{ND}\times\mathrm{ND}$
$g_3 = d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3$
$D_3 = \mathrm{DC}\times\mathrm{DC}\times\mathrm{DC}\times\mathrm{ND}$
$g_4 = d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_1 \neq d_4 \wedge d_2 \neq d_3 \wedge d_2 \neq d_4 \wedge d_3 \neq d_4$
$D_4 = \mathrm{DC}\times\mathrm{DC}\times\mathrm{DC}\times\mathrm{DC}$

and a filter prefixing a domain restricts it to the tuples satisfying the predicate. The first term derives from $\mathcal{A}(Arrival, Susceptible)$, and stands for a summation with only one element: in fact the neutral color class E contains a single element e, and the neutral transition *Arrival* has only one instance with intensity $\varphi(\mathbf{x},Arrival,e) = w(Arrival)\mathbf{x}[Population]$ and the coefficient 1 is that appearing in $\mathcal{A}(Arrival, Susceptible)$. Similar arguments justify the second and third terms (the third term is subtracted because it derives from $\mathcal{R}(Departure, Susceptible)$). The following five terms instead refer to several instances of *Infection* that withdraw tokens from different unfolded places replacing the original place *Infected.* The first term for transition *Infection* comprises only one instance (since $|D_0| = 1$) and the intensity of such instance is $\varphi(\mathbf{x}, Infection, \langle d_1,d_2,d_3,d_4\rangle) = w(Infection)\mathbf{x}[Susceptible]\mathbf{x}[Infected4\mathrm{x}NoDrug]$ (observe that the tokens in place $Infected4\mathrm{x}NoDrug$ have all color $\langle nd,nd,nd,nd\rangle$). The second term for transition *Infection* comprises $|\mathrm{DC}|$ instances (since $|D_1| = |\mathrm{DC}|$) and the intensity of such instances is $\varphi(\mathbf{x}, Infection, \langle d_1,d_2,d_3,d_4\rangle) = w(Infection)\mathbf{x}[Susceptible]\mathbf{x}[Infected1\mathrm{x}Drug]$. Assuming $n = |\mathrm{DC}| \geq 4$, the next three terms comprise respectively $n\cdot(n-1)$, $n\cdot(n-1)\cdot(n-2)$ and $n\cdot(n-1)\cdot(n-2)\cdot(n-3)$ instances and their intensities depend on the (average) marking of *any color*[8] in place $Infected2\mathrm{x}Drug$, $Infected3\mathrm{x}Drug$ and $Infected4\mathrm{x}Drug$, respectively.

The ODE for place $Infected3\mathrm{x}Drug$ is derived similarly:

$$\frac{d\mathbf{x}[Infected3\mathrm{x}Drug,\mathrm{d}_{D_3}]}{d\nu} =$$
$$+w_1\cdot\mathbf{x}[Susceptible]\mathbf{x}[Infected3\mathrm{x}Drug,\mathrm{d}_{D_3}]\cdot 1$$
$$-w_2\cdot\mathbf{x}[Infected3\mathrm{x}Drug,\mathrm{d}_{D_3}]\cdot 1$$
$$+w_3\cdot\mathbf{x}[Infected2\mathrm{x}Drug,\mathrm{d}_{D_3}]\mathbf{x}[Drug]\cdot 1$$
$$-\textstyle\sum_{\langle d1,d2,d3,d4\rangle\in[g_3']D3} w_3\cdot\mathbf{x}[Infected3\mathrm{x}Drug,\mathrm{d}_{D_3}]\mathbf{x}[Drug]\cdot 1$$
$$-\textstyle\sum_{\langle d1,d2,d3,d4\rangle\in[g_3']D3} w_4\cdot\mathbf{x}[Infected3\mathrm{x}Drug,\mathrm{d}_{D_3}]\mathbf{x}[Drug]\cdot 1$$

where: $w_1 = w(Infection)$, $w_2 = w(DepartureI)$, $w_3 = w(Unrecovery)$, $w_4 = w(Recovery)$; $\mathrm{d}_{D_3} = \langle \mathrm{a},\mathrm{b},\mathrm{c},nd\rangle$ represents an arbitrary fixed tuple in $[g_3]D3$ (where a, b, c, are arbitrary distinct colors in DC representing the drugs that were already tried and failed to fight the infection while the last element is $nd \in \mathrm{ND}$) and $g_3' = g_3 \wedge d_1 = \mathrm{a} \wedge d_2 = \mathrm{b} \wedge d_3 = \mathrm{c}$. The last two summations range over $|\mathrm{DC}|-3$ instances: those with $d_1 = a \wedge d_2 = b \wedge d_3 = c$ and $d_4 \in \mathrm{DC}$ and different

[7] In the SNexpression tool all variables name meet this syntax so that their order is a priori defined.

[8] The reason why it is possible to choose the number of tokens of any color in the (average) place marking lies in the model symmetry, that guarantees that such number is surely the same for each color in the place color domain.

from a, b, c.

III. Partial unfolding of a SSN

A preliminary, partial unfolding of some nodes of the original SSN may be required to compute the set of SODE: the reason is that the method assumes that all instances of any transition have the same (base) rate and the (average) number of tokens of each color c in place p at time ν is the same. Transitions are unfolded only if their rates are color dependent: assuming that the rate of t, with an associated guard g_t, is expressed as a set of k pairs $\{\langle g_i, w_i\rangle\}$ (with $w_i \in \mathbb{R}^+$, $j \neq i \Rightarrow \neg(g_i \wedge g_j)$, and $\bigvee_i g_i \equiv g_t$), after the unfolding there will be k copies of t, denoted $t_{[g_i]}$, with same input and output arcs as t, with guard g_i, and rate w_i.

Instead, places are unfolded in two steps: the first step is based on the static partition of color classes[9]. The second step involves those places that have the same static subclass repeated in their color domain, after the first unfolding step. Place unfolding exploits the possibility of prefixing the functions on the arcs incident on an unfolded place instance with *filters* ensuring that the tokens yielded by evaluating the functions match the color pattern associated with that instance.

Let $\widetilde{C_i} = \{C_{i,h}, h : 1 \ldots |\widetilde{C_i}|\}$ be the set of static subclasses of color class C_i. The number of places resulting from the unfolding of a place p with color domain[10] $cd(p) = C_{i_1}^{n_1} \times \ldots \times C_{i_k}^{n_k}$ is given by the product $\prod_{j=1}^{k} |\widetilde{C_{i_j}}|^{n_j}$. Each unfolded place is labelled with $p_{[g]}$, where g is a conjunction of membership clauses $c_{i_j,q} \in C_{i_j,h}$, $\forall j : 1 \ldots k$, $q : 1 \ldots, n_j$, associating a static subclass to each element of the Cartesian product $cd(p)$. Each place $p_{[g]}$ is connected to the same transitions as the original place p, with the corresponding arc functions prefixed by filter $[g]$.

The second place unfolding step involves all those places p that have $n\ (> 1)$ repetitions of a color class C_i in $cd(p)$. The unfolding can be done iteratively, considering one class C_i and one static subclass $C_{i,h} \in \widetilde{C_i}$ at a time. Let $p_{[g]}$ be a place obtained after the first unfolding step, and assume that filter $[g]$ includes $m\ (> 1)$ predicates in the form $c_{i,q} \in C_{i,h}$, then the place is unfolded into as many places as the number np of partitions of a set of cardinality m into at most $|C_{i,h}|$ parts. For each such a partition $\rho_j, j : 1 \ldots np$ (which denotes a partition of the set of symbols $\{c_{i,q}\}$), the corresponding unfolded place p_{ρ_j} is associated with a predicate g'_j, which is the conjunction of a number of (in)equalities: g'_j contains equalities for the variables $c_{i,q}$ that are in the same subset of partition ρ_j, and an inequality for each pair $c_{i,q1}, c_{i,q2}$ belonging to different subsets in partition ρ_j. Each unfolded place, labelled with $p_{[g \wedge g'_j]}$, is connected to the same transitions as $p_{[g]}$, with the corresponding arc functions prefixed by filter $[g \wedge g']$ (in some cases this may result in a null arc). This procedure is not yet included within SNexpression, but its implementation is planned to be released shortly.

[9]In the case a class is not partitioned, we consider it as having only one static subclass.

[10]The notation $C_{i_j}^{n_j}$ means that class C_{i_j} occurs n_j times in the Cartesian product. The variable $c_{i_j,q}$ refers the q^{th} occurrence of C_{i_j}

Observe that, as long as we consider unordered color classes and the static subclasses have cardinality greater than or equal to the maximum number of repetitions of the corresponding class in any place color domain, the partial unfolding of a SSN is independent on static subclass cardinalities. The number of copies of a partially unfolded place may be reduced by taking into account the form of the arc functions appearing on the arcs connected to the place.

In our example model the partial unfolding is limited to places *Drug*, *Storage* and *Infected*. The first unfolding step would produce 2 copies of the first two places and 2^4 copies of the third place, the latter should be further unfolded to take into account the case where elements belonging to the same static subclass are equal or different. Finally some unfolded copies can be eliminated if they remain isolated when considering transition guards. In this example the number of non isolated copies is relatively small: *Drug* and *Storage* are connected only to transitions moving tokens in subclass DC; considering place *Infected* there are only five copies because ND has cardinality 1 (hence all elements in that class are surely equal), and the elements in DC are surely different due to the guards of any transition connected to the place, moreover the guards introduce also restrictions on the way elements belonging to the two static subclasses are interleaved in the tuples. The five copies of *Infected* correspond to the number of acquired antibiotic resistance, which in this model spans from 0 to 4. The arcs connecting transitions *Infection* and *Recovery* to place *Infected* are replicated for each of the five unfolded copies: the arc expressions are prefixed with the filter appropriate for the specific replica of *Infected*. Observe that increasing the size of $DrugChoice$ does not change the structure of the partially unfolded net (while increasing the size of ND would change it). The model in Fig. 1 shows the five instances of place *Infected* and the connections between the unfolded places and transition *Unrecovery* after the unfolding: the filters g_6 to g_{10} appearing in the arc expressions show the static subclasses of each element of the tuples withdrawn from or added to the place and the relation (equality or inequality) between tuple elements belonging to the same static subclass.

IV. The symbolic ODE generation method

In this section, we describe the algorithm (implemented in `SNexpression`) that, starting from a partially unfolded SN, directly derives the associated set of symbolic ODE. The algorithm, essentially based on calculating the structural relations $\mathcal{A}$ and $\mathcal{R}$, operates in a purely symbolic way. It may be logically divided into three steps: referring to a given place p, 1) computation of a compact symbolic expression representing the ODE for the place; 2) refinement of the expression above to take into account transition firing *intensity*, i.e. the firing speed, which may be computed according to an infinite server semantics or a mass action law (in both cases, it is a marking dependent function); 3) automatic derivation of the ODE format ready to be solved through the `R` tool.

In a preliminary version of the algorithm [3], a more complex method to implement step 2) has been proposed (involving also the composition operator, not required in the current method): the new algorithm proposed in this paper is much simpler, more efficient and has been implemented in the tool, hence the generation of the ODE system from an SSN model is now completely automatic.

In the rest of this section the new method for step 2), based on a rewriting of the input arc functions, is presented in detail. The description focuses on the preliminary rewriting of input functions used on 2). Step 1) instead is only summarized: the reader may refer to [3], [4] for the technical details.

A. Preliminary rewriting of input arc functions

A subtle point in the derivation of SODE is the possibility that a transition's instances, that (after the partial unfolding) are assumed to have the same base rate, have different firing intensities, according to the adopted marking dependency. The reason is that an input arc function may take different shapes when evaluated on different instances of a transition.

A rewriting of input arc functions may be required, so that function shapes implicitly match the intensity-based logical partitioning of the set of transition instances.

Specifically, functions $I[p,t] : cd(t) \to Bag[cd(p)]$ have to be expressed as sums $\sum_j \lambda_j F_j, F_j = [g_j]T_j[g'_j]$, where:

a) $j_1 \neq j_2 \Rightarrow (F_{j_1} \cap F_{j_2}) = \emptyset$ b) $g'_{j_1} \neq g'_{j_2} \Rightarrow (g'_{j_1} \wedge g'_{j_2}) =$ false (terms must be pairwise disjoint and the associated guards are either equal or mutually exclusive).

In SNexpression there is an option to enforce such a rewriting: when needed, this preliminary step can be performed once and for all (as the SN partial unfolding), and the requirement is the same independently of the chosen law for transition intensities. Thus the impact of this preprocessing on the method's efficiency is negligible.

Arc functions[11] $I[p,t]$ can thus be partitioned into subsums $\sum_1 + \ldots \sum_m$ of (disjoint) terms having the same guard, i.e., $\sum_h = \left(\sum_{j_h} \lambda_{j_h}[g_{j_h}]T_{j_h}\right)[g'_h]$. Guards $\{g'_h\}$, with the additional complementary one $g'_0 = \neg(\bigvee_h g'_h) \wedge g(t)$, define a *parametric partition* of $cd(t)$: all t's instances matching a given g'_h withdraw the same multiset of colors (up to a symmetry-preserving, color permutation) from p. The predicate g'_0 represents instances of t (if there are any) not withdrawing tokens from p.

Let $p \in {}^\bullet t$: $G_{t,p} = \{g'_0, \ldots, g'_m\}$ denote the intensity-based partition of t instances with respect to p. Let us define the map $\mu[t,p] : G_{p,t} \to \mathbb{N}$ as:

$$\begin{cases} \mu[t,p][g'_0] = 0 \\ \mu[t,p][g'_h] = \eta_h \quad h \neq 0 \end{cases}$$

where the value η_h depends on the law used to compute the transition intensities, e.g., $\eta_h = max(\{\lambda_{j_h}\})$ for infinite-server, $\eta_h = \sum \lambda_{j_h}$ for mass-action.

If $|{}^\bullet t| > 1$, the partitions related to all t's input places $p \in {}^\bullet t$ must be combined in order to get a final one. This is done by calculating the "Cartesian product" $G_t = \otimes_{p \in {}^\bullet t} G_{t,p}$, which results in a set of tuples $\langle g_{p_1}, \ldots, g_{p_n}\rangle$, each interpreted as a conjunctive form (resulting $false$ elements are erased). By construction, G_t (as $G_{t,p}$) represents a partition of $cd(t)$. The map $\mu[t] : G_t \to \otimes_{p \in {}^\bullet t}\mathbb{N}$ associates intensity-equivalent classes of t's instances with corresponding tuples of coefficients. It is defined as follows: let ${}^\bullet t = \{p_1, \ldots, p_n\}$, $g_{p_i} \in G_{t,p_i}$

$$\mu[t](g_{p_1} \wedge \ldots \wedge g_{p_n}) = \langle \mu[t,p_1](g_{p_1}), \ldots, \mu[t,p_n](g_{p_n})\rangle.$$

In the special case ${}^\bullet t = \{p\}$ then $\mu[t] = \mu[t,p]$. Guards mapping to the same tuple of coefficients may be proficiently replaced by a single equivalent "OR" term. The set G_t of guards will be used to refine the ODE symbolic expression on the basis of the intensity of t's instances. In the particular (but not rare) case $G_t = \{g(t)\}$ (meaning that all color instances of t have the same enabling degree), no refinement is needed.

If in $\mu[t]$ there were an element mapping to a tuple with all zeroes, it would mean there are some instances of t not having any input place: this should be considered as a modeling error. *Example* To illustrate the method we refer to the second SSN model used in [4] (Figg. 3-5), which despite an apparent simplicity allows to illustrate some non trivial aspect of the method. The net comprises a single transition t and ${}^\bullet t = \{p_0, p_1\}$, $t^\bullet = \{p_2\}$. Input places have color domain C, where $C = C_1 \cup C_2$ is a basic color class partitioned in two subclasses. The set $var(t)$ includes two type C variables x, y, moreover t which has the guard $[x \in C_1]$, hence $cd(t) = C \times C$. The input arc functions, both of arity $C \times C \to Bag[C]$, are:

$$I[p_0,t] := 2\langle x\rangle + \langle y\rangle \qquad I[p_1,t] := \langle y\rangle$$

The partial unfolding causes the duplication of both input places, to take into account the partition of C: unfolded instances are denoted $p_{0,i}, p_{1,i}$, $i : 1, 2$. Input arc functions are accordingly prefixed by filters $[c \in C_i]$, $i : 1, 2$.

We focus on $p_{0,1}$, the instance of p_0 that holds tokens of color C_1. The function $I[p_{0,1}, t]$ in fact needs to be rewritten as (t's guard $x \in C_1$ is implicit):

$$3\langle x\rangle[x = y] + (2\langle x\rangle + \langle y\rangle)[x \neq y \wedge y \in C_1] + 2\langle x\rangle[y \in C_2]$$

This expression points out that an instance $\langle c, c\rangle$ of t withdraws three tokens of color c, whereas an instance $\langle c, c'\rangle$, with $c \neq c'$, withdraws two tokens of color c and, if $c' \in C_1$, one of color c'. This affects the transition instance intensity that depends on the maximum number of tokens of a certain color withdrawn from a place.

The input functions of the other places do not require any further manipulation: their *simplified* forms, implicitly taking t's guard into account, are:

$$I[p_{0,2}, t] := \langle y\rangle[y \in C_2] \qquad I[p_{1,1}, t] := \langle y\rangle[y \in C_1]$$
$$I[p_{1,2}, t] := \langle y\rangle[y \in C_2]$$

[11]over which the incident transition's guard $g(t)$ implicitly spans

The map G_t is reported below in a matrix form, considering two laws defining the transition instances intensity (the membership clause $x \in C_1$ is implicit): infinite-server (IS) and mass-action (MA). Note that, when the MA law applies, some t's instances are brought together (the case $y \in C_1$ includes the sub-cases in which $x = y$ or $x \neq y$).

$$G_t^{IS} = \begin{array}{l} \\ \{x \neq y \wedge y \in C_1, \\ y \in C_2, \\ x = y\} \end{array} \begin{array}{c} \begin{array}{cccc} p_{0,1} & p_{0,2} & p_{1,1} & p_{1,2} \end{array} \\ \begin{pmatrix} 2 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

$$G_t^{MA} = \begin{array}{l} \\ \{y \in C_1, \\ y \in C_2\} \end{array} \begin{array}{c} \begin{array}{cccc} p_{0,1} & p_{0,2} & p_{1,1} & p_{1,2} \end{array} \\ \begin{pmatrix} 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

B. Symbolic representation of ODE

The terms of the SODE corresponding to place p are based on the symbolic expressions $\mathcal{A}(t,p)$ and $\mathcal{R}(t,p)$, formally represented as weighted sums of *tuples* $\sum_i \lambda_i F_i$, $\lambda_i \in \mathbb{N}$, $F_i = [g_i]T_i[g'_i]$, $\forall c \in cd(p), \forall c' \in cd(t)\ F_i(c)[c'] \leq 1$. A term of $\mathcal{R}$ and $\mathcal{A}$ can be seen as a *parametric* set of t's instances, each one withdrawing/putting λ_i tokens of color c from/to p. Hence we need to compute the *cardinality* of these parametric sets, which may depend on $c \in cd(p)$.

Definition 1 (Constant-size function): A function $F[g] : D \rightarrow Bag[D']$ is constant-size if and only if $\exists k \in \mathbb{N} : \forall c \in D, g(c) \Rightarrow |F(c)| = k$.

The above definition includes the particular case $g \equiv true$. The *cardinality* $|F[g]|$ of a constant-size function is equal to $|F(c)|$, for any c s.t. $g(c)$ is true.

A function tuple $T[g] \in \mathcal{L}$ is constant-size if and only if, for each T's component (class function) f, $f[g]$ is constant-size. A syntactical characterization of constant-size tuples (possibly prefixed by a filter) $[g']T[g] \in \mathcal{L}$ (based on g''s form and T's components) is given in [3].

Property 1: Any expression $e \in \mathcal{L}$ can be rewritten as a weighted sum of constant-size tuples $[g'_i]T_i[g_i]$.

The default normalization command of `SNexpression` tool exploits Property 1 and results in expressions whose terms are constant size.

Example The term of $\mathcal{R}(Unrecovery, Infected3\text{x}Drug)$:

$$\langle d_1, d_2, d_3, S - d_1 \cap S - d_2 \cap S - d_3 \cap S_{\mathrm{DC}} \rangle$$
$$[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \wedge d_1 \in \mathrm{DC} \wedge d_2 \in \mathrm{DC} \wedge$$
$$d_3 \in \mathrm{DC} \wedge d_4 \in \mathrm{ND}]$$

has a guard ensuring that the intersection in the tuple is constant-size: its cardinality (as well as that of the tuple) is $|\mathrm{DC}| - 3$.

Example The term of $\mathcal{R}(Infection, Susceptible)$:

$$[d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3]\langle S_{\mathrm{DC}}, S_{\mathrm{DC}}, S_{\mathrm{DC}}, S_{\mathrm{ND}} \rangle$$

has a filter, that has to be taken into account: letting $n = |\mathrm{DC}|$, $m = |\mathrm{ND}|$, the term's cardinality is expressed by the formula $n(n-1)(n-2)m$. The product $n(n-1)(n-2)$ is the number of triplets with all different elements (chosen from a set of n). The procedure for computing the cardinality of a constant-size function tuple is explained in [4].

`SNexpression` can be instrumented so that terms $F_i = [g_i]T_i[g'_i]$ appearing in $\mathcal{R}$ or $\mathcal{A}$ are not only *constant-size* but also *pairwise disjoint*. Thus, according to the transpose semantics, one such term represents a *set* of $n_i = |F_i|$ colour instances of t, each withdrawing/adding *exactly* λ_i (the term's coefficient in the weighted sum) tokens from/to place p (these instances satisfy the predicate g'_i).

If, in addition, all t colour instances matching $[g_i]T_i[g'_i]$ had the same intensity (denoted by $\varphi(\mathbf{x}(\nu), t)$), we could directly express the SODE relating place p as:

$$\frac{d\mathbf{x}[p,c]}{d\nu} = \sum_{t:p \in t^\bullet, F_i\ in\ \mathcal{A}(t,p)} \lambda_i n_i \varphi(\mathbf{x}(\nu), t) - \sum_{t:p \in {}^\bullet t, F_j\ in\ \mathcal{R}(t,p)} \lambda_j n_j \varphi(\mathbf{x}(\nu), t) \quad (2)$$

Each term in the SODE is a product of four factors: the cardinality of the expression identifying a set of (n_i) homogeneous transition instances, the number (λ_i) of tokens withdrawn/added by any transition instance in the set, the base rate w of transition instances in the set, and the marking-dependent factor (the two last factors are combined in φ). The latter depends on the number of coloured tokens required by the arc functions labelling the input arcs of any transition instance in the set. Its form is dependent on the time semantics.

Some terms $[g_i]T_i[g'_i]$ of $\mathcal{A}$ or $\mathcal{R}$, however, may have to be *preliminarily* split into equivalent sums of tuples representing classes of transition instances characterized by the same intensity. This procedure is described in the next section.

C. Computation of the enabling degree

Let us consider the SODE for place p. We saw that the contribution due to a transition t connected to p is expressed by $\mathcal{R}(t,p)$ or $\mathcal{A}(t,p)$, whose weighted terms $\lambda_i F_i$ represent parametric sets of $n_i = |F_i|$ instances of t, that withdraw/add exactly λ_i tokens from/to p. We may have to split these terms into subterms denoting instances with the same intensity, and we have to derive the marking-dependent factor's formal expression.

Both tasks are straightforwardly carried out by using the set G_t of guards (representing the intensity-based symbolic partition of t's instances) and the associated map $\mu[t]$, computed during the SSN preprocessing (Section IV-A).

If $|G_t| > 1$ (i.e., the intensity-based partition of t's instances is not trivial), G_t guards are used as *filters* to possibly split the parametric sets $F_i = [g_i]T_i[g'_i]$, with $n_i > 1$, into subsets with the same intensity; formally:

$$\lambda_i F_i \mapsto \lambda_i (\textstyle\sum_{g \in G_t} [g \wedge g_i] T_i [g'_i]).$$

This rewriting is coherent, resulting in an equivalent expression, because the domain of any $g \in G_t$ is $cd(t)$, and $\bigvee_{g \in G_t} g \equiv g(t)$ (G_t is a partition of $cd(t)$).

Let F_i' be a subterm of F_i obtained by applying a filter $g \in G_t$ to F_i ($F_i' = F_i \leftrightarrow G_t = \{g(t)\}$). The associated marking-dependent factor to be used in the SODE is directly obtained from the tuple of coefficients $\mu[t](g) = \langle \eta_{p_1}, \ldots, \eta_{p_n} \rangle$. It will take either the form:

$$min_{p_k \in {}^\bullet t, \eta_{p_k} \neq 0}(\mathbf{x}[p_k]/\eta_{p_k}) \quad \text{or:} \quad \prod_{p_k \in {}^\bullet t} \mathbf{x}[p_k]^{\eta_{p_k}},$$

depending on whether the IS or the MA law is used.

Example The $\mathcal{R}$ terms for the partially unfolded input places p_0, p_1 of the simple model introduced at the begin of this section come to be (there are no $\mathcal{A}$ terms for these places):

$$\begin{aligned}
\mathcal{R}(t, p_{0,1}) =& 1\langle S - c_1 \cap S_{C1}, c_1 \rangle [c_1 \in C1] + \\
& 2\langle c_1, S - c_1 \cap S_{C1} \rangle [c_1 \in C1] + \\
& 2\langle c_1, S_{C2} \rangle [c_1 \in C1] + 3\langle c_1, c_1 \rangle [c_1 \in C1] \\
\mathcal{R}(t, p_{0,2}) =& 1\langle S_{C1}, c_1 \rangle [c_1 \in C2] \\
\mathcal{R}(t, p_{1,1}) =& 1\langle S_{C1}, c_1 \rangle [c_1 \in C1] \\
\mathcal{R}(t, p_{1,2}) =& 1\langle S_{C1}, c_1 \rangle [c_1 \in C2]
\end{aligned}$$

Let us consider first the SODE for the IS law from the point of view of $p_{0,1}$ (the restriction of p_0 for $c \in C_1$). The instances of t that withdraw tokens from this place are given by $\mathcal{R}(t, p_{0,1})$ above. According to its form, such instances are partitioned into four pair-wise disjoint sets. By the way, applying the filters in G_t^{IS} to $\mathcal{R}(t, p_{0,1})$ doesn't have any effect, because each term of cardinality > 1 of $\mathcal{R}(t, p_{0,1})$ (that, we recall, is the transpose of the -preliminarily rewritten- input arc function $I[p_{0,1}, t]$) represents a symbolic set of pairs of different colors of type C. Thus, the SODE for $p_{0,1}$ is (w_t is the transition's base rate):

$$\begin{aligned}
\frac{d\mathbf{x}[p_{0,1}]}{d\nu} = -w_t \cdot (& 1 \cdot |C_1 - 1| \cdot min(\mathbf{x}[p_{0,1}]/2, \mathbf{x}[p_{1,1}]/1) + \\
& \cdot 2 \cdot |C_1 - 1| \cdot min(\mathbf{x}[p_{0,1}]/2, \mathbf{x}[p_{1,1}]/1) + \\
& 2 \cdot |C_2| \cdot min(\mathbf{x}[p_{0,1}]/2, \mathbf{x}[p_{0,2}]/1, \mathbf{x}[p_{1,2}]/1) + \\
& 3 \cdot 1 \cdot min(\mathbf{x}[p_{0,1}]/3, \mathbf{x}[p_{1,1}]/1) \,)
\end{aligned}$$

Let us now consider place $p_{1,1}$ (the restriction of p_1 for $c \in C_1$): the only term of $\mathcal{R}(t, p_{1,1})$ (of cardinality $|C_1|$) needs to be rewritten, according to G_t^{IS}, to reflect the two cases with different enabling degree, namely $c_1 = c_2$ and $c_1 \neq c_2$ (in both cases $c_1, c_2 \in C_1$). Hence the tuple $\langle S_{C_1}, c_1 \rangle [c_1 \in C_1]$ is split into $\langle c_1, c_1 \rangle [c_1 \in C_1]$ and $\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]$, whose sizes are 1 and $|C_1| - 1$, respectively. The SODE for $p_{1,1}$ (IS law) comes to be:

$$\begin{aligned}
\frac{d\mathbf{x}[p_{1,1}]}{d\nu} = -w_t \cdot (& 1 \cdot (|C_1| - 1) \cdot min(\mathbf{x}[p_{0,1}]/2, \mathbf{x}[p_{1,1}]/1) + \\
& 1 \cdot min(\mathbf{x}[p_{0,1}]/3, \mathbf{x}[p_{1,1}]/1) \,)
\end{aligned}$$

As for the MA semantics, let us refer again to place $p_{0,1}$. The corresponding SODE is:

$$\begin{aligned}
\frac{d\mathbf{x}[p_{1,0}]}{d\nu} = -w_t \cdot (& 2 \cdot |C_1| \cdot \mathbf{x}[p_{0,1}]^3 \cdot \mathbf{x}[p_{1,1}]^1 + \\
1 \cdot |C_1| \cdot \mathbf{x}[p_{0,1}]^3 \cdot \mathbf{x}[p_{1,1}]^1 + & 2 \cdot |C_2| \cdot \mathbf{x}[p_{0,1}]^2 \cdot \mathbf{x}[p_{0,2}]^1 \cdot \mathbf{x}[p_{1,2}]^1)
\end{aligned}$$

V. SNEXPRESSION

`SNexpression` is a `Java` tool-set for the symbolic structural analysis of Symmetric Nets and its application. The `SNexpression` design started in the mid of '12. A free version of the software and a short user-manual can be downloaded at the URL: http://www.di.unito.it/∼depierro/`SNexpression`. The tool is organized on two layers: the base is a Library for Symbolic Calculus (LSC); on top of it there is an interface (CLI) implementing a textual calculator.

A. The CLI

The CLI is a shell surrounding the library and implementing a parser of expressions that involve the principal operators used in structural analysis of SN models. The CLI reads from the standard input, thus it can be either used as off-the-shelf instrument or integrated in other design/analysis tools.

Commands from the input stream are read and interpreted, one per line. A command basically can be: a definition of a symbol stored in the CLI's memory for future use, e.g., a color domain or a function definition; an expression indicating a formula of the calculus; an invocation of a built-in function or a structural relation defined on Symmetric Nets.

The CLI operates at two different levels: at the low level, the user can directly enter *any* expression of the language representing structural relations, which is solved into a corresponding normal form (as explained next); at a higher level, the user can load the description of a SN net and query the CLI for the net's structural properties. Since version 2.2.0 in fact the CLI provides support to SN definitions that satisfy the .sn syntax explained in appendix A of the `SNexpression`'s short manual. Several functional relations, typical of SN structural analysis, can be directly quested on a loaded net. At the current release, the relations enumerated in Table I can be computed. Structural relations can map onto sets or multisets, the table follows this classification.

Besides that, the CLI was newly equipped wit a module for the automatic derivation of *symbolic* ODE (SODE) [3] from (partially unfolded, section III) Stochastic SN models (SSN). The format of the generated SODE system is fully compliant with the solver of `R`. For the sake of debugging, it is possible to separately run the several subtasks involved in SODE's computation, using built-in commands introduced to the purpose. A one-step command computing the SODE system from an SSN is also available.

B. The Library for Symbolic Calculus

The `SNexpression`'s core is a library implementing a rewriting system. According to this paradigm, a program is a set of rules that are used to rewrite algebraic terms. Rewriting goes on until no more rules apply, in which case the resulting term is in "normal form" and is taken to be the "value" of the initial expression.

TABLE I: `SNexpression` commands working on an SSN

Commands to be applied on an SSN	
load "net_name.sn"	Read from a file the description of an SSN
print_ode	Compute and save the set of Symbolic Ordinary Differential Equations associated with the SSN
Structural relations on sets computable with `SNexpression`	
SC(t, t', p)	Returns the function representing the structural conflict between instances of t and of t' with respect to place p, parameterized on colors of t';
SC(t, p)	It is equivalent to $SC(t, t, p)$;
SCC(t, t', p)	Returns the function representing the t's instances in structural causal connection with t' due to the shared place p, parameterized on colors of t';
SCC(t, p)	It is equivalent to $SCC(t, t, p)$;
SME(t, t', p)	Returns the color function representing the structural mutual exclusion between instances of t and t' due to place p parameterized on colors of t';
Structural relations on multisets computable with `SNexpression`	
ABmset(t, p)	Returns a function, parameterized on $c \in cd(p)$, representing the instances of t adding tokens of color c in p. ABmset(t, p)(c) returns a multiset on $cd(t)$ whose coefficients indicate the number of tokens of color c each instance of t adds to p;
RBmset(t, p)	Similar to ABmset(t, p) but representing the instances of t removing tokens of color $c \in cd(p)$

The expressions manipulated by the LSC are built of SN annotations, and all and only those functional operators that appear in the definition of symbolic structural relations. The normalized expressions match the language $\mathcal{L}$ of SN symbolic structural relations. Normalized function tuples, considered individually, meet a canonical form ensuring that their application results in constant-size (multi)sets and equivalence to the empty constant function (and consequently, function equivalence) can be syntactically checked. It is also possible to set up the normalization so that sums of are composed of pairwise disjoint tuples.

The LSC supports a *parametric* calculus of structural relations, i.e, the size of color-classes are specified by constraints associated with terms. An expression may be split during normalization according to the rules of the calculus, resulting in a list of normalized terms with more specific constraints.

Earlier versions of the library didn't fully support the symbolic calculus of SN structural relations, due to some limitations in solving the composition operator. The current version overcomes all previous restrictions. In addition, it makes it possible to manipulate (non-parametric) expressions mapping to multisets, as SN arc functions originally do. All the aforementioned functional operators apply to multiset-expressions as well, and are solved by the library, with the only exception of composition, which, in the available version, is treated only partially (we are completing its general treatment).

a) Architecture and implementation: The library's highly modular structure consists of around one hundred `Java` classes/interfaces. To face complexity both in design and debug/testing, the library has been developed following rigorously, in all its parts, a variant of the Interpreter pattern.

Interpreter's intent is, given a language, define a representation for its grammar along with an engine that uses it to interpret language's sentences. A type is associated with each rule of the grammar, which is either a Composite object (a rule that references to other rules) or a leaf in the type hierarchy. The root is an abstract type declaring an interpretation method (`xxxSimplify` in our setting), which is implemented by each concrete subtype. Reducing a term to a normal form thus relies on the recursive traversal of the underlying Composite.

Interpreter doesn't permit any distinction between domain-specific and generic rules, thus enforcing the writing of boilerplate, hard-to-maintain, code. The LSC has been designed according to a variation of Interpreter [10] which cleanly separates "generic" rules from domain-specific ones. The idea is simply to gather into abstract types at the roots of the type hierarchy polymorphic rewriting rules, demanding the definition of domain-specific rules to the leafs. Term normalization is thus straightforwardly and efficiently implemented as a fixed-point iteration alternating specific and generic rules.

The top of LSC type hierarchy is shown in Figure2. Interface `Expression` represents rewritable terms with a given arity. Any expression is provided with domain-specific (method `specSymplify`) and generic (method `genSymplify`) rules. `Expression.normalize` implements a fixed-point normalization relying on the two methods above. Sub-interfaces `ParametricExpr` and `LogicalExpr` denote terms with an associated constraint and terms of a generic Boolean algebra, respectively. The `ParametricExpr.simplify` method normalizes and possibly splits a parametric expression, up to a fixed-point.

The abstract classes `ClassFunction`, `Guard`, and `FunctionTuple` represent the elements of a SN arc-function. They are the roots of sub-hierarchies with a similar structure, all matching Boolean algebra. Rewriting this kind of terms involves rules such as De Morgan, associativity, idempotence, and so forth, factorized in `LogicalExpr` and sub-interfaces.

Interface `BagExpr` represents the root of the hierarchy of multiset-expressions, that mostly builds on the same types describing the SN annotations.

b) Advantages of the adopted design: The adopted design, even though requires the use of a large number of small-size classes/interfaces, has a number of advantages:

- *extensibility/easy maintainability:* any changes or updates can be done at a very low price, e.g., adding new types (rules) just means to identify the right place in the hierarchy where insert them.
- *modular testing/debugging:* every single element of the grammar can be separately treated (normalized), in a uniform way;

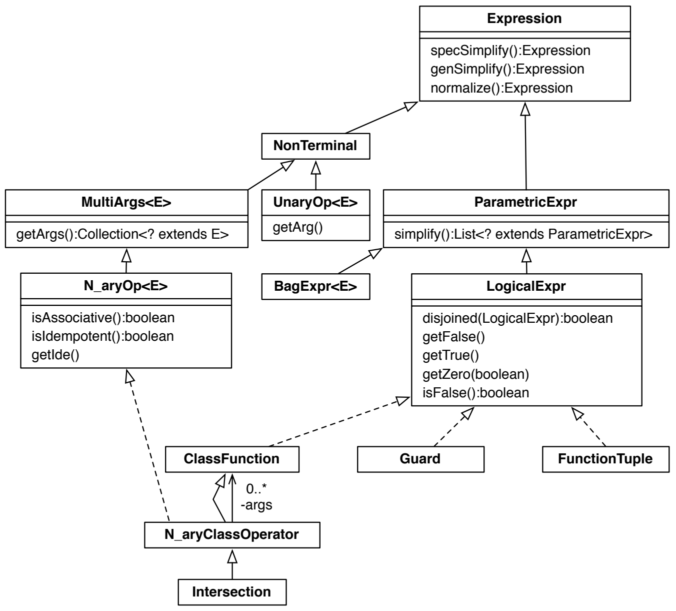


Fig. 2: The class-diagram of expressions

- *efficiency:* term normalization is a complex task (e.g., bringing a term to a disjoint normal form has a complexity exponential in the number of involved literals) hence efficiency issues have been taken into a great consideration: the major overhead, due to equivalence checks, has been greatly reduced by the adopted technique (based on syntactical comparisons and limiting the use of recursion) The normalization times for the examples reported on the web site (some of which very complex) range from ms to a very few secs.

C. Application of the tool to example SSNs

In this section the results on the SIRS example are shown and those on a Botnet model are summarized: the number of equations and overall size of the SODE system is compared with the corresponding sizes of the ODE system obtained from the complete unfolding of the SSN model. In this example the mass action law is the suitable one for the computation of transition intensities. The second example concerns a botnet model introduced in [3] and inspired by [18]: in this case the infinite server semantics is more appropriate. In both cases the method based on the complete unfolding prevents the computation of indices when making the size of one class grow, while the SODE system size does not change (only the coefficients in the SODE are updated).

Figure 3 plots the average marking of four places of the SIRS model: the results have been computed both with the set of ODE obtained from the completely unfolded model (up to a certain size), and with the set of SODE obtained with the SNexpression tool: the plotted lines show a perfect matching, indeed the relative difference is very small (always below 10^{-7}, the precision set for the numerical solution of the system of differential equations). Table II shows the size of the system of ODE and of SODE for different sizes of static

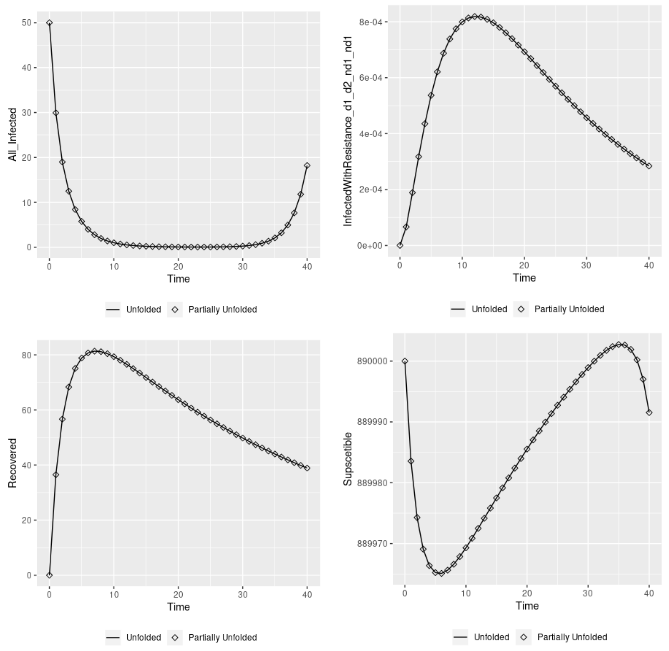


Fig. 3: Some measures obtained from the completely unfolded model and the compact one: in each plot the two curves match.

subclass DC showing the limit of the solution based on the complete unfolding.

Similar results have been obtained for a SSN Botnet Model in [3]: in this case two color classes have been used, one is composed of four cardinality one static subclasses, the other one represents the identity of various LANs that can undergo a Botnet attack; only the latter allows to exploit symmetries. Also in this example the number of SODE does not grow while increasing the size of the second color class while the completely unfolded model grows very quickly to an unmanageable size: the SODE system has 7 equations with 28 terms overall while the number of equations in the ODE system is 20 times larger when the symmetric class contains 10 colors, 57 times when the class has 20 colors, 250 times when the class has 50 colors.

The files describing both nets (described according to the SNexpression syntax) and the SODE system automatically obtained for each model are available for download in the SNexpression web page.

VI. Conclusions and future work

In this paper a new module of SNexpression (a quite recently developed tool targeted at the structural analysis of SNs) has been presented, for the automatic derivation of a set of symbolic ODE (SODE) from a SN model with stochastic timing (SSN): the technique significantly extends the range of models that can be analysed to cases with a huge number of ODEs (and huge state-spaces), by efficiently solving compact equivalent systems of ODEs reflecting the model's symmetry. The set of SODE is automatically derived from a *partially*

Complete Unfolding						
n	5	6	7	8	9	10
♯ ODE	1311	2418	4115	6582	10023	14666
♯ Trans.	831	2076	4409	8334	14451	23456
R size (MB)	0.350	0.718	1.6	2.8	4.6	7.4
Sol. time (s)	30	67.45	131.98	785.94	1770.35	n.c.
Mem. peak (GB)	0.96	1.18	1.6	2.9	6.6	13.4
Partial Unfolding (SODE)						
n	> 4					
♯ SODE	12					
♯ terms	74					
R size	6KB					
Sol. time (s)	0.5					

TABLE II: Comparison of the size of the set of ODE and the set of SODE varying $n = |\mathrm{DC}|$.

unfolded SSN model, through an algebraic manipulation of its colour annotations. Although the idea behind the approach has been presented in [3], in this paper it has been improved (a much more simple and general algorithm has been developed) and implemented in SNexpression. The paper also provides a high-level description of the main components of the tool and shows a few examples of how its facilities can help in the specific case of Symbolic ODE computation. To the best of our knowledge no other tool implements this kind of automatic ODE *aggregation*: even in efficient PN tools (e.g., Snoopy [17]) a compact representation of colored Petri nets is exploited in model construction and for some basic analysis, but not for deterministic simulation. In the context of a fluid framework for PEPA, a result similar to that in [5] was presented in [19], but the aggregation is based on exact fluid lumpability. SNexpression also supports the computation of several (symbolic) structural properties, in particular the recent implementation of (one kind of) the composition operator is the first step towards the automatic verification of a condition (coloured P-invariant coverage) that is required [2] to ensure that the average marking evaluated by solving the system of ODE derived from the net well approximates the real average. An integration of SNexpression (in particular, of the module computing and solving the system of SODE) with the GreatSPN SSN editor and analyser, is also planned.

Appendix

A. Symmetric Nets

A SN is a tuple $(P, T, \Sigma, var, cd, \Phi, I, O, m_0)$. P and T are the sets of *places* and *transitions*, respectively. $\Sigma = \{C_i : i = 1 \dots n\}$ is the set of *basic color classes* on which transition and place color domains are built[12]. Each basic color class can be statically partitioned in subclasses, $C_i = \bigcup_k C_{i,k}$. Place p color domain $cd(p)$ is defined as Cartesian product of a set of color classes (possibly with repetition of the same class). var assigns to each transition $t \in T$ a set of variables, each taking values in a given element C_i of Σ (the variable's type); fixed an order on the set of variables, the color domain of t, $cd(t)$, is defined as the Cartesian product of the its variables' types. Φ is a function assigning a guard to each transition t; $\Phi(t) : cd(t) \rightarrow \{true, false\}$ is a boolean function defined on $var(t)$, and is denoted through a boolean expression whose terms are basic predicates on $var(t)$; the admissible basic predicates are $v = v'$, $v \in C_{i,q}$, $d(v) = d(v')$ where $v, v' \in var(t)$ have same type C_i and $d(v)$ denotes the static subclass $C_{i,j}$ of the color assigned to v by a given transition instance $c \in cd(t)$.

m_0 denotes the initial marking, that associates to each $p \in P$ a multiset on $cd(p)$.

I and O associate each pair $(t, p) \in T \times P$ to a map $I[p, t]$, $O[p, t]$ (called arc function), annotating a corresponding oriented arc connecting t and p (if the arc doesn't exist the corresponding function is the empty constant). An arc function has arity $cd(t) \rightarrow Bag[cd(p)]$, and it is defined through an expression:

$$\sum_i \lambda_i . T_i[g_i], \quad \lambda_i \in \mathbb{N}$$

T_i are function-tuples: a function-tuple T, denoted by $\langle f_1, \dots, f_k \rangle$, is the Cartesian product of *class functions* f_i. Each class-function f is a linear function defined on a subset of variables of $var(t)$ of the same type. Let $var_{C_i}(t) = \{v_1, \underset{\smile}{\dots} v_m\}$ be the subset of variables in $var(t)$ of type C_i, and $\widetilde{C_i}$ the set of static subclasses of C_i, then $f : C_i^m \rightarrow Bag[C_i]$ is so defined:

$$f = \sum_{k=1}^{m} \alpha_k . v_k + \sum_{q=1}^{|\widetilde{C_i}|} \beta_q . S_{i,q}$$

where $\alpha_k, \beta_k \in \mathbb{Z}$. $S_{i,q}$ is a constant functions called *diffusion/synchronization*: it maps its argument to the set $C_{i,q}$. The following equivalence holds: $\sum_q S_{i,q} = S_i$. Scalars must be such that no negative coefficient result from the evaluation of f for any color satisfying the guard possibly associated with the function-tuple or transition.

Fixed an order on $var(t)$ defining $cd(t)$, consistent with the local ordering on $var_{C_i}(t)$, and assuming that the color domain of class-functions naturally extends to the tuple's color domain, the semantics of a function-tuple $T := \langle f_1, \dots, f_k \rangle$ is $T(c) = \langle f_1(c), \dots, f_k(c) \rangle$, where the Cartesian product of multisets is defined as: let $m_1 \in Bag[A], m_2 \in Bag[B]$, then $\langle m_1, m_2 \rangle$ is an element of $Bag[A \times B]$ such that $\forall a \in A, b \in B$, $\langle m_1, m_2 \rangle(\langle a, b \rangle) = m_1(a) \cdot m_1(b)$.

A guarded tuple $T[g]$, where g is a predicate defined similarly to transition guards, maps each element $c \in cd(t)$ in $T(c)$ if $g(c) = true$, in $\emptyset$ otherwise.

B. Operators of the symbolic structural relations calculus

Definition 2 (Transpose): Let $f : D \rightarrow Bag[D']$ be a function, its transpose $f^t : D' \rightarrow Bag[D]$ is defined as: $f^t(x)[y] = f(y)[x], \forall x \in D', y \in D$.

Definition 3 (Difference): Let $f, g : D \rightarrow Bag[D']$ be two functions. The difference $f - g : D \rightarrow Bag[D']$ is defined as: $f - g(x) = f(x) - g(x), \forall x \in D$.

The multiset difference is: $b, b' \in Bag[A], a \in A$, $(b - b')[a] = max(0, b[a] - b'[a])$.

[12]The SN formalism has also circularly ordered classes but we omit this feature here since they are not considered in this paper.

References

[1] S. Baarir, M. Beccuti, D. Cerotti, M. D. Pierro, S. Donatelli, and G. Franceschinis. The GreatSPN tool: recent enhancements. *SIGMETRICS Performance Evaluation Review*, 36(4):4–9, 2009.

[2] M. Beccuti, E. Bibbona, A. Horvath, R. Sirovich, A. Angius, and G. Balbo. Analysis of Petri net models through stochastic differential equations. *Lecture Notes in Computer Science.*, 8489 LNCS:273–293, 2014.

[3] M. Beccuti, L. Capra, M. D. Pierro, G. Franceschinis, and S. Pernice. Deriving Symbolic Ordinary Differential Equations from Stochastic Symmetric Nets Without Unfolding. In *Computer Performance Engineering - 15th European Workshop, EPEW 2018, Paris, France, October 29-30, 2018, Proceedings*, volume 11178 of *LNCS*, pages 30–45. Springer, 2018.

[4] M. Beccuti, L. Capra, M. D. Pierro, G. Franceschinis, and S. Pernice. Deriving Symbolic Ordinary Differential Equations from Stochastic Symmetric Nets Without Unfolding. Technical Report TR-INF-2018-07-03-UNIPMN, Univ. del Piemonte Orientale, 2018.

[5] M. Beccuti, C. Fornari, G. Franceschinis, S. Halawani, O. Ba-Rukab, A. Ahmad, and G. Balbo. From Symmetric Nets to differential equations exploiting model symmetries. *Computer Journal*, 58(1):23–39, 2015.

[6] L. Capra, M. D. Pierro, and G. Franceschinis. A high level language for structural relations in well-formed nets. In *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005, Proceedings*, pages 168–187, 2005.

[7] L. Capra, M. D. Pierro, and G. Franceschinis. A tool for symbolic manipulation of arc functions in symmetric net models. In *7th International Conference on Performance Evaluation Methodologies and Tools, ValueTools '13, Torino, Italy, December 10-12, 2013*, pages 320–323, 2013.

[8] L. Capra, M. D. Pierro, and G. Franceschinis. Computing Structural Properties of Symmetric Nets. In *Quantitative Evaluation of Systems, 12th International Conference, QEST 2015, Madrid, Spain, September 1-3, 2015, Proceedings*, volume 9259 of *LNCS*, pages 125–140. Springer, 2015.

[9] L. Capra, M. D. Pierro, and G. Franceschinis. Deriving Symbolic and Parametric Structural Relations in Symmetric Nets: Focus on Composition Operator. Technical Report TR-INF-2019-03-01-UNIPMN, DiSIT, Computer Science Institute, UPO, 2019.

[10] L. Capra and V. Stile. An extension of the interpreter pattern to define domain-parametric rewriting systems. In *2013 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 185–192, Sep. 2013.

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

[12] M. De Pierro. *Ph.D. thesis: Structural analysis of conflicts and causality in GSPN and SWN*. Università di Torino - Italia., 2004 - http://di.unito.it/∼depierro/public/.

[13] C. Dutheillet and S. Haddad. Conflict Sets in Colored Petri Nets. In *proc. of Petri Nets and Performance Models*, pages 76–85, 1993.

[14] S. Evangelista. Syntactical Rules for Colored Petri Nets Manipulation. Technical Report CEDRIC-04-641, CEDRIC-CNAM Paris, January 2004.

[15] S. Evangelista, S. Haddad, and J. Pradat-Peyre. Syntactical Colored Petri Nets Reductions. In D. Peled and Y.-K. Tsay, editors, *Automated Technology for Verification and Analysis 2005*, volume 3707 of *LNCS*, pages 202–216. Springer, Heidelberg, 2005.

[16] S. Evangelista and J.-F. Pradat-Peyre. On the Computation of Stubborn Sets of Colored Petri Nets. In S. Donatelli and P. Thiagarajan, editors, *ICATPN 2006*, volume 4024 of *LNCS*, pages 146–165. Springer Berlin Heidelberg, 2006.

[17] F. Liu, M. Heiner, and D. Gilbert. Coloured Petri nets for multilevel, multiscale and multidimensional modelling of biological systems. *Briefings in bioinformatics*, 11 2017.

[18] E. V. Ruitenbeek and W. H. Sanders. Modeling peer-to-peer botnets. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST 2008)*, QEST '08, pages 307–316, Washington, DC, USA, 2008. IEEE Computer Society.

[19] M. Tschaikowski and M. Tribastone. Exact fluid lumpability for Markovian process algebra. In M. Koutny and I. Ulidowski, editors, *CONCUR 2012 – Concurrency Theory*, pages 380–394. Springer Berlin Heidelberg, 2012.