DiSIT, Computer Science Institute Università del Piemonte Orientale "A. Avogadro" Viale Teresa Michel 11, 15121 Alessandria http://www.di.unipmn.it



Deriving Symbolic Ordinary Differential Equations from Stochastic Symmetric Nets without Unfolding

M. Beccuti, L. Capra, M. De Pierro, G. Franceschinis, S. Pernice (beccuti@di.unito.it, inserire indirizzo e-mail, inserire indirizzo e-mail, giuliana.franceschinis@uniupo.it, pernice@di.unito.it)

TECHNICAL REPORT TR-INF-2018-07-03-UNIPMN (July 2018)

Research Technical Reports published by DiSIT, Computer Science Institute, Università del Piemonte Orientale are available via WWW at URL http://www.di.unipmn.it/. Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

- 2018-02 Power (set) Description Logic, L. Giordano, A. Policriti, February 2018.
- 2018-01 A Game-Theoretic Approach to Coalition Formation in Fog Provider Federations (Extended Version), C. Anglano, M. Canonico, P. Castagno, M. Guazzone, M. Sereno, February 2018.
- 2017-02 Configuration and Use of Android Virtual Devices for the Forensic Analysis of Android Applications (see below for citation details), C. Anglano, M. Canonico, M. Guazzone, June 2017.
- 2017-01 A dynamic simulation model for comparing kidney exchange policies, M. Beccuti, G. Franceschinis, S. Villa, March 2017.
- 2016-04 *Tracing sharing in an imperative pure calculus*, P. Giannini, M. Servetto, E. Zucca, December 2016.
- 2016-03 SUPPORTING DATA COMMUNICATION AND PATIENT ASSESSMENT DUR-ING EMERGENCY TRANSPORTATION, M. Canonico, S. Montani, M. Striani, September 2016.
- 2016-02 TECHNICAL NOTE TO Forensic Analysis of the ChatSecure Instant Messaging Application on Android Smartphones (see below for citation details), C. Anglano, M. Canonico, M. Guazzone, September 2016.
- 2016-01 *Reasoning in a rational extension of SROEL*, L. Giordano, D. Theseider Dupré, May 2016.
- 2014-02 A Provenly Correct Compilation of Functional Languages into Scripting Languages, P. Giannini, A. Shaqiri, December 2014.
- 2014-01 An Intelligent Swarm of Markovian Agents, A. Bobbio, D. Bruneo, D. Cerotti, M. Gribaudo, M. Scarpa, June 2014.
- 2013-01 Minimum pattern length for short spaced seeds based on linear rulers (revised), L. Egidi, G. Manzini, July 2013.
- 2012-04 An intensional approach for periodic data in relational databases, A. Bottrighi, A. Sattar, B. Stantic, P. Terenziani, December 2012.
- 2012-03 Minimum pattern length for short spaced seeds based on linear rulers, L. Egidi, G. Manzini, April 2012.
- 2012-02 Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems, C. Anglano, M. Canonico, M. Guazzone, April 2012.
- 2012-01 Trace retrieval and clustering for business process monitoring, G. Leonardi, S. Montani, March 2012.
- 2011-04 Achieving completeness in bounded model checking of action theories in ASP, L. Giordano, A. Martelli, D. Theseider Dupré, December 2011.

Deriving Symbolic Ordinary Differential Equations from Stochastic Symmetric Nets without unfolding

M. Beccuti¹, L. Capra², M. De Pierro¹, G. Franceschinis³, and S. Pernice¹

¹ Dip. di Informatica, Università di Torino, Italy

² Dip. di Informatica, Università di Milano, Italy

 $^{3}\,$ DISIT, Università del Piemonte Orientale, Alessandria, Italy

Abstract. This report concerns the quantitative evaluation of Stochastic Symmetric Nets (SSN) by means of a fluid approximation technique particularly suited to analyse systems with a huge state space. In particular a new efficient approach is proposed to derive the deterministic process approximating the original stochastic process through a system of Ordinary Differential Equations (ODE). The intrinsic symmetry of SSN models is exploited to significantly reduce the size of the ODE system while a symbolic calculus operating on the SSN arc functions is employed to derive such system efficiently, avoiding the complete unfolding of the SSN model into a Stochastic Petri Net (SPN).

Keywords: Stochastic Symmetric Nets, Ordinary Differential Equations, Symmetries, Symbolic analysis, Symbolic structural techniques.

1 Introduction

SSNs [7] are a colored extension of SPNs [10]: both formalisms are widely used for modeling and analysing Discrete Event Dynamic Systems. The underlying stochastic process, a Continuous Time Markov Chain (CTMC), can be automatically generated and studied using numerical and simulative techniques or approximated by an ODE system.

This paper extends the result described in [3] in which: (1) we identified a class of SSNs whose underlying CTMC can be approximated by an ODE system according to Kurtz's theorem [8]; (2) we proposed an algorithm to generate a reduced ODE system exploiting the SSN model symmetries. This algorithm requires an unfolding step before generating the reduced system.

To overcome this limitation we propose a new approach based on a symbolic calculus for SSN arc functions to generate the compact ODE system without prior unfolding. Such calculus was introduced in [4] where a language extending the arc expressions syntax of SSNs and some operators on the language elements were presented and applied to SSN structural properties computation. The calculus was implemented in the SN expression tool [5]. In [6] a more comprehensive formalization of SSN structural properties and the generalization of all operators (with some limitations on composition) to work on multisets extended the method applicability. In this paper the ability to symbolically manipulate the arc functions of SSNs is exploited to build the reduced set of Symbolic ODEs (SODEs) directly. The three main contributions are: (1) the definition of the formulae for the derivation in symbolic form of the terms to be included in each SODE, (2) the definition of an approach to compute the cardinality of specific language expressions representing groups of similar terms in a single ODE, that can thus be *compressed* in a single term in the SODE, and (3) the definition of a procedure to express the *enabling degree* of transition instances appearing in the SODE in symbolic form. The main steps required to automatically derive the complete set of SODE have been implemented.

This is the first approach in which the syntax of SSNs is exploited to directly generate a compact ODE system (we refer to [12] for a general overview on PNs and fluid approximation). Indeed, even in efficient PN tools (e.g., Snoopy [9]) the compact representation of colored models is exploited in model construction and for some basic analysis, but not for the deterministic simulation. In the context of a fluid framework for PEPA, a result similar to that in [3] was presented in [13], but the aggregation is based on exact fluid lumpability.

The paper is organized as follows: in Sec. 2 the background and the notation needed to understand the new approach are introduced. In Sec. 3 the new approach is illustrated on a case study and the main properties needed to automatically generate the SODE are presented. In Sec. 5 we report a set of experimental results illustrating the method efficiency. Conclusions and directions for future work are discussed in Sec. 6.

2 Background

In this section, after presenting the case study used for illustrating the new approach, the SSN formalism is introduced and a description on how to derive the SODE system from an SSN model is presented, recalling the results in [3].

2.1 Our case study in a nutshell

Our case study is inspired by the model presented in [11]: Botnets are networks of compromised machines under the control of an attacker that uses those compromised machines for a variety of malicious/nefarious purposes.

Typically, initial infection involves a malware, called *Trojan horse*, which installs a malicious code into a vulnerable machine. The injected malicious code begins its bootstrap process and attempts to join the Botnet. A machine connected to the Botnet becomes a *bot* and can send spam (a working bot) or infect new machines (a propagation bot). The bot is inactive most of the time to reduce the probability to be detected and becomes active only for very short periods. An infected machine can be recovered if an anti-malware software discovers the virus or if the computer is physically disconnected from the network. The corresponding SSN model is reported in Fig. 1. In the next subsections its main components are introduced together with the elements of the formalism.

2.2 The SSN formalism

The SSN formalism [7] adds *colors* to the SPN formalism, so that information can be associated with the tokens in the net. This feature usually leads to a

more compact system representation which may be exploited during both the construction and the solution of the model [3,7].

An SSN is a bipartite directed graph with two types of nodes: *places* and transitions. Places, graphically represented as circles, coincide with the state variables of the system. For instance the places of the Botnet model in Fig. 1 are NoConBot, ConBot, InactiveBot and ActiveBot, corresponding to four possible phases through which a machine under attack can flow. Places contain tokens, whose colors are defined by the color domain cd(), expressed as Cartesian product of color classes C_i . Color classes can be partitioned in static subclasses $\{C_{i,j}, j = 1, \ldots, k\}$. Colors in a class represent entities of the same nature but only colors within the same static subclass are guaranteed to behave similarly. A color class may be *ordered* and in this case a successor function denoted by ! is defined on it, which determines a circular order on its elements. In the model of Fig. 1 there are two color classes: Mac and Loc. The former is partitioned into four static subclasses of cardinality one (the machine infection states): \mathbf{N} (ormal), I(nfected), W(orking Bot), P(ropagation Bot). The latter, representing machine locations, is not partitioned into subclasses. The color domains of all the places is $Mac \times Loc$ (representing pairs (machine infection state, location)).

Transitions, graphically drawn as boxes, represent the system events: in our example, the flow through attack phases and changes in the infection state of a machine. The instances of a transition t are defined by its color domain cd(t)defined as a list of typed variables (with types chosen among the color classes) or as the Cartesian product of its variables types (assuming an implicit order among its variables). The transition variables appear in the functions labeling its arcs. A transition instance $\langle t, c \rangle$ binds each variable to a specific color of proper type. A guard can be used to restrict the allowed instances of t: it is a logical expression defined on cd(t), and its terms, called basic predicates allow one to (1) compare colors assigned to variables of the same type ($x = y, x \neq y$); (2) test whether a color belongs to a given static subclass ($x \in C_{i,j}$); (3) compare the static subclasses of the colors assigned to two variables ($d(x) = d(y), d(x) \neq d(y)$).

For instance *RecInitInf* is a transition in the model of Fig. 1. Its color domain is $Mac \times Mac \times Loc$ (assuming variables' order x, y, l), restricted by the guard $[x \in I \land y \in N]$ to the colors that associate variables x and y to the subset of machines in *infected* and *not infected* state respectively.

The state of an SSN, called *marking*, is defined by the number of colored tokens in each place. The initial marking of the model in Fig. 1, representing one infected machine and 1000 not infected machines in each location, is

 $NoConBot(1000\langle N, Loc \rangle) + ConBot(\langle I, Loc \rangle).$ (1) Places and transitions are connected through arcs decorated with *arc functions* defining both the enabling conditions for the transition instances and the state change caused by their firing. The function on the arc connecting place p and transition t has domain cd(t) and codomain Bag[cd(p)], where Bag[A] is the set of multisets built on set A, and if $b \in Bag[A], a \in A, b[a]$ denotes the multiplicity of a in multiset b. Given a transition instance, the input and output arc functions map the transition color into (multi)sets of colored tokens matching the corresponding place color domain. Input and output arcs are denoted



Fig. 1: SSN model for the Botnet.

 $I, O[p,t] : cd(t) \to Bag[cd(p)]$. A transition instance $\langle t, c \rangle$ is enabled in marking m if $\forall p \in {}^{\bullet}t, \forall c' \in cd(p)I[p,t](c)[c'] \leq m[p][c']$ (${}^{\bullet}t$ and t^{\bullet} represent the set of input and output places of t, respectively). An enabled instance may fire causing a state change from m to m' defined as follows: $\forall p, m'[p] = m[p] - I[p,t](c) + O[p,t](c)$.

The arc functions are formally expressed as sums of tuples, with each tuple element chosen from a set of predefined basic functions whose domain is the transition color domain and whose codomain is $Bag[C_i]$, for a given color class C_i . The tuples may have an associated guard, expressed with the same syntax of transition guards, allowing to include or exclude the tuple from the sum depending on the truth value of the guard for a given transition instance. The basic functions are: projection, denoted by a variable in the transition color domain (e.g., x and l appearing in the arc expression $\langle x, l \rangle$); successor, denoted !x, where x is a variable whose type is an ordered class; a constant function returning all elements in a class (or subclass), denoted S_{C_i} (or $S_{C_{i,j}}$). A linear combination of basic functions is a class function, e.g. $S_{C_i} - x$, where x is of type C_i , is a class function returning all elements of class C_i except element x.

The stochastic behavior of an SSN model is characterized by the assumption that the firing of any enabled transition occurs after a random delay sampled from a negative exponential distribution. A function ω is associated with each transition and defines its firing rate as follows:

$$\omega(t,c) = \begin{cases} r_i & \text{if } cond_i(c), \ i = 1, \dots, n; \\ r_{n+1} & \text{otherwise} \end{cases}$$

where $cond_i$ is a boolean expression comprising standard predicates on the transition color instance. Hence, the firing rate $r_i \in \mathbb{R}^+$ of a transition instance can depend only on the static subclasses of the colors assigned to the transition variables and on the comparison of variables of the same type. We assume that the conditions $cond_i$ are mutually exclusive. For instance, the rate associated with transition InitInf representing the infection propagation event is 10.0 if q = l, otherwise 2; also BeBot, representing the start of Working or Propagation Bot activity, has rate 20.0 if $(y \in W)$ and 2 if $(y \in P)$. The stochastic process driving the dynamics of an SSN model is a CTMC, where the states are identified with SSN markings and the state changes correspond to the marking changes in the model. In this context we assume that all the transitions of the SSN use an infinite server policy, and we define the intensity of $\langle t, c \rangle$ in marking m as:

$$\varphi(m,t,c) = \omega(t,c) \min_{\langle p_j,c'\rangle: I[p_j,t](c)[c'] \neq 0} \left\lfloor \frac{m[p_j][c']}{I[p_j,t](c)[c']} \right\rfloor$$

where the last factor is e(m, t, c), the enabling degree of $\langle t, c \rangle$ in m.

2.3 From SSN models to ODE

In [2] a class of SPN was identified whose stochastic behavior can be approximated through a deterministic process in agreement with the Kurtz's results in [8]: considering an SPN model whose places are all covered by P-semiflows and whose transitions use an *infinite server policy*, the underlying CTMC satisfies the *density dependent property* (i.e. the intensities of the transitions can be expressed as a function of the density of the tokens $\frac{m(p)}{N}$ where N is a constant depending on the P-semiflows and the initial marking) and it is possible to derive a set of ODE providing a good deterministic approximation of the average number of tokens in the places when the number of interacting objects (i.e. tokens) is large. In [3] we showed that similar results can be derived for SSN models and we described how to automatically generate the ODE system from an SSN model through the net unfolding: the average number of tokens in each place of the unfolded net is approximated through the following ODE:

$$\frac{dx_i(\nu)}{d\nu} = \sum_{j=1}^{|T|} \varphi(x(\nu), t_j) (O[p_i, t_j] - I[p_i, t_j])$$
(2)

where $x(\nu)$ is a vector of real numbers representing the average number of tokens in the model places at time ν , T is the set of the net transitions, and $\varphi(x(\nu), t_j)$ is a function defining the intensity of transition t_j in the state $x(\nu)$ as follows:

$$\varphi(x(\nu), t_j) = \omega(t_j) \min_{l: I[p_l, t_j] \neq 0} \frac{x_l(\nu)}{I[p_l, t_j]},\tag{3}$$

where $\omega(t_j)$ is obtained by $\omega(t, c)$ through the unfolding of $\langle t, c \rangle$ into t_j . In [3], we proposed a translation method which reduces the size of the ODE system by automatically exploiting the model symmetries. This is achieved through the notion of "symbolic" ODE (SODE): 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. However, this method still required an initial unfolding of the model to generate the ODE system that is automatically reduced in a second step. The goal of this paper is instead to directly derive the SODE from the SSN.

2.4 Symbolic manipulation of SSN arc functions

In this section the definitions and notations required to explain how to derive the set of SODE are introduced: the method is based on symbolic manipulation of expressions of a language \mathcal{L} (that look like SSN arc functions with a few syntactical extensions) through a set of operators (difference, transpose, composition).

The elements of language $\mathcal L$ have 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 correspond one-to-one to the elements in the Cartesian product D': they are intersections (\cap) of basic class functions from set⁴ $BS = \{v, S - v, S_C, S_{C_k}\}$ denoting functions $D \to Bag[C]$, 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. The functions in BS are a subset of SSN class functions. The intersection is not part of the arc functions syntax, but allows any SSN arc function to be rewritten to a \mathcal{L} expression. For instance $(v_i, v_j \text{ variables of type } C)$: $\langle S_{C_k} - v_i, v_i \rangle [v_i \in C_k] \ (\notin \mathcal{L}) \equiv$ $\langle (S_{C_k} \cap (S - v_i), v_i \rangle [v_i \in C_k] \ (\in \mathcal{L}); \ \langle S - v_i - v_j, v_i, v_j \rangle [v_i \neq v_j] \ (\notin \mathcal{L}) \equiv$ $\langle (S - v_i) \cap (S - v_j), v_i, v_j \rangle [v_i \neq v_j] \ (\in \mathcal{L})$. Symbols $[g_j], [g'_j]$ are defined on D and D', respectively. Symbol [g], where g is a SSN standard predicate defined on D, denotes a function $D \to D$: [g](d) = d if $g(d) = true, [g](d) = \emptyset$ if g(d) = false. Observe that the SSN arc function syntax may include guards but not filters.

Language \mathcal{L} is closed with respect to a set of operators among which the *transpose* and the *difference*; SNexpression (www.di.unito.it/~depierro/SNex) implements the rules for symbolically treating these operators.

Definition 1 (Transpose). Let $f: D \to Bag[D']$ be a function, its transpose $f^t: D' \to Bag[D]$ is defined as: $f^t(x)[y] = f(y)[x], \forall x \in D', y \in D$.

Definition 2 (Difference). Let $f, g : D \to Bag[D']$ be two functions. The difference $f - g : D \to Bag[D']$ is defined as: $f - g(x) = f(x) - g(x), \forall x \in D$. The multiset difference is: $b, b' \in Bag[A], a \in A, (b-b')[a] = max(0, b[a] - b'[a]).$

In the sequel the language, its operators and its properties are the key formal tools to define the SODE characterizing an SSN model without unfolding it. In particular the difference and transpose operators allow us to define and express in symbolic form the functions $\mathcal{R}(t,p)$ and $\mathcal{A}(t,p)$, where t is a transition and p is a place connected to t. Function $\mathcal{R}(t,p)$, called *Removed By*, defines which instances $\langle t,c'\rangle$ of t withdraw tokens of color $c \in cd(p)$ from place p. Function $\mathcal{A}(t,p)$, called Added By, defines which instances $\langle t,c'\rangle$ add tokens of color $c \in cd(p)$ into place p. $\mathcal{R}(t,p)(c)[c']$ and $\mathcal{A}(t,p)(c)[c']$ denote the number of tokens of color c withdrawn by/ added by instance $\langle t,c'\rangle$ from/to p.

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

⁴ To keep the presentation simple, ordered classes are not considered here, but the presented results extend to models including them.

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

For instance, place ActiveBot (whose cd is $Mac \times Loc$) is connected to transition RecActive with $cd : x \in Mac, y \in Mac, l \in Loc$ and with guard $y \in N$. The expression for $\mathcal{R}(RecActive, ActiveBot) = (\langle x, l \rangle [y \in N])^t$, is $\langle \tilde{x}, S_N, \tilde{l} \rangle$ denoting a function from cd(ActiveBot) to Bag[cd(RecActive)]. Here the names \tilde{y} and \tilde{l} indicate respectively the first occurrence of class Mac and of class Loc in cd(ActiveBot), while the color identifying an instance of RecActive is indicated as $\langle x, y, l \rangle$. As expected the instances of RecActive that remove tokens of color $\langle \tilde{x}, \tilde{l} \rangle$ from ActiveBot are those with $x = \tilde{x}, y \in N, l = \tilde{l}$.

3 The symbolic ODE generation method

The approach for deriving the SODE corresponding to a given place p comprises two steps. Let x[p,c] be the number of c-colored tokens in place p at time ν (in order to keep notation simple we will omit time dependency).

Step 1. For each transition t connected to place p: if there is an arc from p to t compute $\mathcal{R}(t, p)$, if there is an arc from t to p compute $\mathcal{A}(t, p)$.

Step 2. The differential equation for place p and color $c \in cd(p)$ is defined as:

$$\frac{dx[p,c]}{d\nu} = \sum_{\langle t,c'\rangle:p\in t^{\bullet},c'\in\mathcal{A}(t,p)(c)} \varphi(x(\nu),t,c')(\mathcal{A}(t,p)(c)[c']) - \sum_{\langle t,c'\rangle:p\in \bullet,c'\in\mathcal{R}(t,p)(c)} \varphi(x(\nu),t,c')(\mathcal{R}(t,p)(c)[c']),$$
(4)

Each sum spans over all instances $\langle t, c' \rangle$ that withdraw (negative sum) or add (positive sum) tokens of color c from/to p. The *intensity* of $\langle t, c' \rangle$ is multiplied by $\mathcal{A}(t,p)(c)[c']$ or $\mathcal{R}(t,p)(c)[c']$ (i.e., by the number of tokens of color c added to or withdrawn from p by $\langle t, c' \rangle$) to get the actual flow of tokens in/out p.

Due to the symmetry of SSN arc functions the above procedure can be done for just an *arbitrary* color $c \in cd(p)$. This statement is only partially true, in fact the symmetry is surely preserved only in subsets of cd(p) containing colors that cannot be distinguished through standard predicates operating on cd(p): for this reason a partial unfolding of the places may be needed (e.g. due to the presence of static subclasses). In order to apply the symbolic approach the intensity $\varphi(x(\nu), t, c')$ must also be symmetric: this may require the partial unfolding of transitions. Special care should be taken in case the cd(t) includes variables with same type: in this case one should treat separately instances in which the same color or different colors are assigned to these variables since this may influence both the rate of $\langle t, c' \rangle$ and the number of tokens of color c flowing in or out of p.

Symbolic representation of ODE Due to symmetries, each summation over the color domain of a given transition t in equation (4) may be computed efficiently by grouping instances with "similar" rate and same number of tokens moved into or out of the place. This may be achieved by expressing equation (4) in a compact, symbolic way. As anticipated, a preliminary partial unfolding of some nodes of the original SSN may be needed: each place p' in the partially unfolded net, derives from a place p in the original model and an SSN predicate g on cd(p)

taking into account the partition of color classes in subclasses, and the possibility that elements of same class in the tuples of cd(p) be equal or different. In the partially unfolded net a filter [g] prefixes the function on any arc connected to p'; notation $p_{[g]}$ shall be used for the place names in the partially unfolded net to put in evidence the original place name and predicate g. If cd(p) contains only one occurrence of C and g is $[c \in C_j]$ we shall use the notation p_{C_j} . For what concerns transitions, each t' in the partially unfolded net must satisfy $\forall c_1, c_2 \in cd(t') : \omega(t', c_1) = \omega(t', c_2) = \omega(t')$, in this case the unfolded transitions t' deriving from transition t in the original model shall be characterized by a guard which is the conjunction of t guard and the condition $cond_i$ associated with value r_i in the definition of $\omega(t)$. This kind of partial net unfolding will be illustrated on the example.

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 expressed as weighted sums of *tuples* $\sum_i \lambda_i F_i, \ \lambda_i \in \mathbb{N}, \ F_i = [g_i]T_i[g'_i], \ \forall c \in cd(p), \forall c' \in cd(t) \ F_i(c)[c'] \leq 1$. Each 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 each parametric set, that may depend on $c \in cd(p)$.

Definition 3 (Constant-size function). A guarded 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 guarded tuple $T[g] \in \mathcal{L}$ is constant size if and only if, for each T's component (a class function) f, f[g] is constant size. The following property defines a syntactical condition for a (guarded) class function f[g] being constant size:

Property 1. f[g] is constant-size if: f either belongs to the basic-set BS of class functions or it takes one of these forms

a)
$$\bigcap_{j \in Q, |Q| < |C|} S - v_j \qquad b) \quad S_{C_k} \bigcap_{j \in J, |J| < |C_k|} S - v_j$$

where in b) for each $v_i: g \Rightarrow v_i \in C_k$.

The cardinalities of terms of type a) and b) are |C| - |Q| and $|C_k| - |J|$, respectively. The cardinalities of functions in BS can be easily inferred. For instance, function $S - v_1 \cap S - v_2[v_1 \neq v_2]$, where v_1, v_2 are two variables of type C, is constant size, with cardinality |C| - 2.

When transposing a given expression with the SN expression tool each term [g']T[g] in the resulting sum is such that T[g] is constant size. We finally state a syntactical condition on a filter [g'] ensuring that $[g']T[g] \in \mathcal{L}$ is constant-size.

Property 2. $[g']T[g] \in \mathcal{L}$ is constant-size if T[g] is constant size and

- 1. g' is a conjunctive form composed only of (in)equations $c_i = (\neq)c_i, i < j$,
- 2. for each (in)equation $c_i = (\neq)c_j$ the corresponding class-*C* functions f_i, f_j in *T* are such that $f_j \equiv f_i$,

Condition (2) says that tuple components referred to by any (in)equation in the filter must be equal.

Proof of Property 2 Let us constructively prove Property 2 by sketching the general algorithm for computing tuple cardinality.

We can express $[g]T[g']: D \to Bag[D']$ as $(\bigotimes_{C \in D'} [g_C]T_C)[g']$, where $[g_C]T_C$: $D \to Bag[C^e]$, e being the number of repetitions of C in D'. In other words we consider separately the subtuples of T involving each class C and the terms in g involving those components. Note that it may be $g_C = true$ for some colour class C. The function [g]T[g'] is constant-size iff every $[g_C]T_C[g']$ is constant-size and in this case $|[g]T[g']| = \prod_C |[g_C]T_C[g']|$.

Let us focus on $[g_C]T_C[g']$. Let $J(g_C) = \{j\}$, s.t. c_j occurs in g_C , in other words $J(g_C)$ identifies the set of variables c_j of type C appearing in g_C . We can partition g_C (a conjunctive form) into $\{g_1, \ldots, g_n\}$, such that for each g_i, g_j , $i \neq j, J(g_i) \cap J(g_j) = \emptyset$ (in this way we separate independent subsets g_i of terms in g_C)). The terms in g_i can be partitioned in equalities and inequalities: let us introduce the notation $g_i = g_{i,eq} \wedge g_{i,neq}$ to separate the two parts of g_i . Note that $g_{i,eq}$ or $g_{i,neq}$ may be simply true. Without loss of generality, we assume that equalities in $g_{i,eq}$ take all the form $c_j = c_x$ (for an arbitrarily fixed c_j), and $g_{i,eq} \neq true \wedge g_{i,neq} \neq true \Rightarrow J(g_{i,eq}) \cap J(g_{i,neq}) = \{j\}$, in other words if g_i contains both equalities and inequalities there is just one variable c_j occuring both in $g_{i,eq}$ and in $g_{i,neq}$.

Under the initial hypothesis, all elements in subtuple T_C corresponding to the index set $J(g_i)$ are equal. Let λ (> 1) be their cardinality, and let us denote card_i the cardinality of the T_C 's subtuple corresponding to $J(g_i)$ after being filtered through g_i . If $g_{i,neq} \equiv true$ (g_i just contains equalities) then card_i is simply λ . Otherwise $g_{i,neq}$ can be seen as a system of inequalities among n = $|J(g_{i,neq})|$ integer variables on the domain $\{1, \ldots, \lambda\}$. Let G be the connected graph of order n representing such a system: the number of system's solutions (= card(i), for the particular form of g_i) is the chromatic polynomial value $P(G, \lambda)$, corresponding to the number of distinct λ -colourings of G.

Finally, the cardinality of $[g_C]T_C[g']$ is obtained by multiplying $\prod_i card(i)$ by the cardinality of T_C components not corresponding to any index in $J(g_i)$.

Example: the tuple $[c_1 \neq c_2 \land c_2 \neq c_3] \langle S_{C1} - c, S_{C1} - c, S_{C1} - c, S, c \rangle [c \in C1]$ has domain C and co-domain $C \times C \times C \times C$ (i.e. C^4); each c_i appearing in the filter represents the i - th element in tuple T, $C = C1 \cup C2$ hence |C| = |C1| + |C2|and |C1| = 4, |C2| = 2. Observe that the first three elements in the tuple are equal, and this is coherent with the hypothesis that elements compared in some term of the filter g' must be equal. The tuple can be divided in two independent sub-tuples: the first $[c_1 \neq c_2 \land c_2 \neq c_3] \langle S_{C1} - c, S_{C1} - c, S_{C1} - c \rangle [c \in C1]$ and the second $\langle S, c \rangle [c \in C1]$. The guard makes the elements $S_{C1} - c$ constant size: without this guard the size would be |C1| if $c \notin C1$ and |C1| - 1 if $c \in C1$.

The filter of the second subtuple is *true*. The filter of the first subtuple doesn't involve any equality while it comprises two inequalities. The cardinality of the tuple elements are: $|S_{C1} - c| = |C1| - 1$, |S| = |C|, |c| = 1.

The first subtuple has as many elements as the number of possible colorings of a graph G with three nodes, each one associated with one of the three variables c_1, c_2 , and c_3 , and an edge between pairs of variable-nodes that appear in an inequality of the filter. Since $|S_{C1} - c| = |C1| - 1 = 3$ in this case P(G, 3) = 12. The filter of the second subtuple is *true*, hence its cardinality is simply |S||c| =|C| = 6. Finally the cardinality of the complete tuple is 12 * 6 = 72.

Property 3. Any expression $e \in \mathcal{L}$ can be rewritten as a weighted sum of constantsize terms $[g'_i]T_i[g_i]$.

The SNexpression tool can be instrumented to produce expressions in the form introduced in Property 3. The expression obtained from the tool does not have a canonical form: depending on the order of application of the rewriting rules the expression terms may be grouped in different ways; it is however guaranteed that the terms $F_i = [g_i]T_i[g'_i]$ appearing in \mathcal{R} or \mathcal{A} are pairwise disjoint and *constant-size*. Thus, according with the transpose semantics, a term $[g_i]T_i[g'_i]$ of \mathcal{R} or \mathcal{A} represents a set of $n_i = |[g_i]T_i[g'_i]|$ t's colour instances each one 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 enablingdegree and hence consequently the same intensity (denoted by $\varphi(x(\nu), t)$), we could directly express the SODE relating place p:

$$\frac{dx[p,c]}{d\nu} = \sum_{t:p\in t^{\bullet}, F_i \text{ in } \mathcal{A}(t,p)} \lambda_i n_i \varphi(x(\nu),t) - \sum_{t:p\in \bullet, F_j \text{ in } \mathcal{R}(t,p)} \lambda_j n_j \varphi(x(\nu),t)$$
(5)

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 ω of any transition instance in the set, and its enabling degree (the two factors are combined in φ). The last factor depends on the number of colored tokens required by the arc functions labelling the input arcs of any transition instance in the set. Some terms $[g_i]T_i[g'_i]$ of \mathcal{A} or \mathcal{R} may have to be split into equivalent sums of tuples representing classes of transition instances with the same enabling degree. The procedure for computing the enabling degree is described later.



Fig. 2: Partially unfolded (sub)net (note filters $[c \in N/I/P/W]$) including all transitions connected to place $NoConBot_N$.

Table 1: List of rates in the $NoConBot_N$ equation and functions \mathcal{A} and \mathcal{R} needed to derive it, where $g = (c \in N)$.

The Botnet example Let us illustrate the idea on the Botnet example to point out the main problems that have to be solved to automatize the whole process. Only the equation for place $NoConBot_N$, obtained by partially unfolding place NoConBot, is developed completely since similar arguments apply to the other places. The method generates one distinct equation for each place in the partially unfolded net: since all places in the BotNet model have $cd(p) = Mac \times Loc$ and only Mac is partitioned in four static subclasses, each place p will be unfolded into four new places p_I , p_N , p_W and p_P , and filters $[c \in X]$, where X stands for a static subclass of Mac, will prefix the functions on the arcs connected to place p_X as shown in Fig.2. In some cases it is possible to simplify the partially unfolded net taking into account the transition guards: in Fig.2 for instance some filters are not present because the transition guards make them redundant (e.g. see the arc from RecActive to $NoConnBot_N$), moreover if the combination filtertransition guard results in a surely empty function, then the arc can be deleted (this is the case for the arc from $ActiveBot_W$ and transitions $InitInf_{[a]}$). On the Botnet example only a subset of colors can be found in the model places as explained hereafter: this allows to further simplify the partially unfolded model. The first element (Mac) of tokens in NoConBot can only be in N or in I, those in Active/InactiveBot can only be in W and P (see grey-colored empty instances in Fig.2), finally those in place ConBot can only be in I.

Table 1 shows the expressions \mathcal{A} and \mathcal{R} for each transition connected to place $NoConBot_N$ from which we can generate a differential equation with several terms, depending on the number of terms in the expressions \mathcal{A} and \mathcal{R} . Observe that some term may need to be transformed into an equivalent sum of terms to separate the transition instances with different enabling degree or rate.

In our model we assume that all transitions have uniform base rate except *InitInf* and *BeBot*: the former has a different rate depending whether the two locations l' (of the machine which is going to be infected) and l (of the bot which is going to propagate the infection) are equal or different. The latter has a different rate for working bot and propagation bot generation. Hence the partial unfolding shall generate two instances of InitInf, $InitInf_{[l=l']}$ and $InitInf_{[l\neq l']}$ (see Fig.2), and two instances of BeBot: $BeBot_W$ and $BeBot_P$.

In the following equation we denote with $x[p_X, c, l]$ the mean number of tokens in the place instance $\langle p_X, c, l \rangle$ (where X is one of the static subclasses in *Mac*). Thus, the differential equation corresponding to $\langle NoConBot_N, c, l \rangle$ is:

$$\frac{dx[NoConBot_N, c, l]}{du} = |P|\omega_1 x[ActiveBot_P, c', l] + |W|\omega_2 x[ActiveBot_W, c', l] +$$

+ $|P|\omega_3x[InactiveBot_P, c', l]$ + $|W|\omega_4x[InactiveBot_W, c', l]$ + $\omega_7x[NoConBot_I, c', l]$ + - $|P||I|\omega_5\min(x[NoConBot_I, c', l], x[ActiveBot_P, c'', l])$ +

 $-|P||I|(|Loc|-1)\omega_6\min(x[NoConBot_I, c', l], x[ActiveBot_P, c'', l]),$

where rates ω_i are defined in Table 1. Coefficients |P| and |W| in the first four terms are the cardinality of the tuples $\langle S_P, c_1, l_1 \rangle$ and $\langle S_W, c_1, l_1 \rangle$ respectively: these are obtained by splitting the term $\langle S_{Mac}, c_1, l_1 \rangle [c1 \in N]$, common to $\mathcal{A}(RecActive, NoConBot_N)$ and $\mathcal{A}(RecInactive, NoConBot_N)$, into $\langle S_P, c_1, l_1 \rangle [g] +$ $\langle S_W, c_1, l_1 \rangle [g]$ (the terms $\langle S_I, c_1, l_1 \rangle [g] + \langle S_N, c_1, l_1 \rangle [g]$) do not appear because they correspond to instances of *RecActive* and *RecInactive* that will never be enabled). Coefficients |P||I| and |P||I|(|Loc|-1) in the last two terms of the SODE are the cardinalities of tuples $\langle c_1, S_I, S_P, l_1, l_1 \rangle$ and $\langle c_1, S_I, S_P, l_1, S_{Loc} - l_1 \rangle$, respectively. We omit the factor 1 preceding the ω_i , derived from the coefficient of the corresponding term in \mathcal{A} or \mathcal{R} .

Computation of the enabling degree Let us consider the SODE for place p. 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 $\lambda_i[g_i]T_i[g'_i]$ represent parametric sets of $n_i = |[g_i]T_i[g'_i]|$ instances of t, that withdraw/add λ_i tokens from/to p. We need a method to derive the enabling degree of such instances, by possibly splitting terms with $n_i > 1$ into subterms denoting instances with same enabling degree.

Let $F_i = [g_i]T_i[g'_i]$ be one such term. Due to symmetries, for each place $p' \in {}^{\bullet}t$ we just have to evaluate the arc function I[p', t] on an arbitrary element of the parametric set F_i . This operation corresponds to a *composition* of two elements of $\mathcal{L}: I[p', t] \circ Tr_i$, where Tr_i is a cardinality-1 symbolic tuple *representative* of F_i . This particular composition is supported by the SNexpression tool and results in an element of \mathcal{L} .

Definition 4 (Composition). Given l_1 and l_2 in \mathcal{L} where l_2 has constant size equal to 1, the composition $l_1 \circ l_2$ is defined as $l_1 \circ l_2(c) = l_1(l_2(c))$; where l_1 is evaluated on the single element in the (multi)set returned by $l_2(c)$.

The representative tuple Tr_i has the same co-domain as F_i and domain D_e , which may be equal to D of F_i or be extended. If $n_i = 1$ the representative tuple Tr_i coincides with $[g_i]T_i[g'_i]$. Otherwise it is defined as $[g_i]T'_i[g''_i]$, according to Table 2, which maps T_i components to the corresponding T'_i ones; a conjunction of additional predicates may be introduced. The type of class-functions on the first row is the only admitted in constant-size tuples. Symbols c_h are new variables (index h must exceed the number of repetitions of C in D) that occur only once in Tr_i . These symbols cause an extension of the original domain.

 Table 2: Syntactical rules to derive a representative tuple

T_i component	$ f, f[g_i'] = 1$	S	S_{Ck}	$\bigcap_{w \in A} S - c_w$	$S_{Ck} \bigcap_{w \in A} S - c_w$
T'_i component	f	c_h	c_h	c_h	C_h
$g_i'' = g_i' \wedge \dots$	-	-	$c_h \in Ck$	$\bigwedge_{w \in A} c_h \neq c_w$	$c_h \in Ck \bigwedge_{w \in A} c_h \neq c_w$

As an example, consider $\langle S - c_1 \cap S - c_2, c_1 \rangle [c_1 \neq c_2]$, with domain C^2 . Its representative, with domain C^3 , is $\langle c_3, c_1 \rangle [c_1 \neq c_2 \wedge c_1 \neq c_3 \wedge c_2 \neq c_3]$.

The following property formalizes the notion of *representative* tuple:

Property 4. Let $c' \in D_e$, and let $c'_D \in D$ denote the projection on D of c'

- $\forall c' \in D_e : Tr_i(c') \in [g_i]T_i[g'_i](c'_D);$
- $\forall c \in D : [g_i]T_i[g'_i](c) = \bigcup_{c' \in D_e, c'_D = c} Tr_i(c').$

The composition $I[p', t] \circ Tr_i$ results in $\sum_j \lambda_j F_j$, $F_j = [g_j]T_j[g'_j]$. If this summation contains a single term then λ_1 is the coefficient to be used as divisor of x[p'], in the formal expression of the enabling degree of t. Otherwise it can be rewritten⁵ so that its terms are pairwise disjoint $(F_{j_1} \cap F_{j_2} \equiv \emptyset)$, and guards $[g'_i]$ are either equal or mutually exclusive.

We can thus partition the summation into subsums $\sum_{j_1} + \sum_{j_2} \dots$ of terms characterized by having the same guard, i.e., $(\sum_{j_h} \lambda_{j_h}[g_{j_h}]T_{j_h})[g_h]$. The guard of each subsum (that, we recall, is a function $cd(t) \to cd(t)$) identifies a subset of t's instances that require the same number λ_{j_h} tokens of a color c_{j_h} from place p', so that the enabling degree (w.r.t. input place p') can be expressed as: $x[p']/\lambda_h^*$, $\lambda_h^* = max(\{\lambda_{j_h}\})$. These guards are applied as filters to split the parametric set F_i of t's instances, into subsets with constant enabling degree (w.r.t. p'): formally $\lambda_i F_i \mapsto \lambda_i (\sum_h [g'_h \land g_i] T_i[g'_i])$. By repeatedly applying the procedure on the obtained subterms on the re-

By repeatedly applying the procedure on the obtained subterms on the remaining places of $\bullet t$, we finally get the SODE expression for F_i , that will take the form: $\lambda_i \sum_h n_{i_h} \omega_t e_{i_h}(x)$, where $e_{i_h}(x) = \min_{p' \in \bullet t} (x[p']/\lambda_{i_h}^*)$, $\sum_h n_{i_h} = n_i$.

Example Let us illustrate the procedure on the Botnet model. When building the SODE of place $\langle NoConBot_N, c \in N, l \in Loc \rangle$ all the connected transitions should be considered: they are shown in Fig.2. Let us consider only one of them: $InitInf_{[l \neq l']}$: in Tab.1 we find the expression for $\mathcal{R}((InitInf_{[l \neq l']}, c, c' \in I, c'' \in P, l, l'), (NoConBot_N, c \in N, l \in Loc))$, namely $\langle c, S_I, S_P, l, S - l \rangle [c \in N]$. The last term in the SODE of $NoConBot_N$ originates from this expression which represents |I||P|(|Loc|-1) transition instances. A representative tuple for it is: $\langle c, c', c'', l, l' \rangle [c \in N, c' \in I, c'' \in P, l \neq l']$; to compute its enabling degree we need to know how many tokens are required in each input place ($ActiveBot_P$ and $NoConBot_N$) to ensure its enabling. We already have the number of tokens (of color $\langle c \in N, l \in Loc \rangle$) required in $NoConBot_N$ since it is the coefficient of the

⁵ In the SN xpression implementation there is an option to enforce such rewriting.



Fig. 3: Second SSN model

considered term in \mathcal{R} that is 1; the multiset of tokens required in $ActiveBot_P$ by the representative instance of $InitInf_{l\neq l'}$ can be computed by performing the composition $\langle c'', l' \rangle [c'' \in P] \circ \langle c, c', c'', l, l' \rangle [c \in N, c' \in I, c'' \in P, l \neq l']$ resulting in $\langle c'', l' \rangle [c'' \in P]$, so it is only one token, hence the enabling degree is $e(x, InitInf_{[l\neq l']}, c, c' \in I, c'' \in P, l, l') = min(x(NoConBot_I, c, l), x(ActiveBot_P, c'', l'))$. Similar arguments apply to $InitInf_{[l=l']}$. In the other terms of the SODE the min function does not appear because the corresponding transitions have only one input place: for each of them the procedure illustrated above indicates that only one colored token is required by the input arc function composed with the representative tuple (so the divisor in the enabling degree formula is simply 1).

4 Second SSN example

This section provides another model: it is small but it shows some interesting situations not illustrated in the Botnet example. The SSN model of Fig. 3 has only one color class C with two static subclasses C_1, C_2 . Note that in the color domain of place P_2 class C appears twice. Due to the transition guard, the first element of the pairs in P_2 must necessarily belong to C_1 . In order to compute the SODE we have to consider a partial unfolding of the model based on the static partition of the places color domain. We denote P_{0-i} and P_{1-i} , i = 1, 2 the two instances of P_0 and P_1 . The possible instances of P_2 instead are: P_{2-11}, P_{2-12} with double index due to the repetition of C in cd(P2), and first index equal to one due to the transition guard. Place P_{2-11} has repetition of class C in its color domain and when the two elements belong to the same static subclass the case in which the two elements are equal or different must be separated: $P_{2-11neq}$ and $P_{2-11neq}$. The partially unfolded model is shown in Fig.4 while in Fig. 5 it is shown a simplified version of the same partially unfolded model. Let's compute the \mathcal{R} and \mathcal{A} expressions for each place instance.

 $\mathcal{R}(T, P_{0-1}) = 1 \langle S - c_1 \cap S_{C1}, c_1 \rangle [c_1 \in C1] + 2 \langle c_1, S - c_1 \cap S_{C1} \rangle [c_1 \in C1] + 2 \langle c_1, S_{C2} \rangle [c_1 \in C1] + 3 \langle c_1, c_1 \rangle [c_1 \in C1]$ $\mathcal{R}(T, P_{0-2}) = 1 \langle S_{C1}, c_1 \rangle [c_1 \in C2]$



Fig. 4: Second SSN model after partial unfolding



Fig. 5: Second SSN model after partial unfolding and simplification

 $\begin{aligned} \mathcal{R}(T, P_{1-1}) &= 1 \langle S_{C1}, c_1 \rangle [c_1 \in C1] \\ \mathcal{R}(T, P_{1-2}) &= 1 \langle S_{C1}, c_1 \rangle [c_1 \in C2] \\ \mathcal{A}(T, P_{2-11eq}) &= 1 \langle c_1, c_2 \rangle [c_1 \in C1, c_2 \in C1, c_1 = c_2] \\ \mathcal{A}(T, P_{2-11neq}) &= 1 \langle c_1, c_2 \rangle [c_1 \in C1, c_2 \in C1, c_1 \neq c_2] \\ \mathcal{A}(T, P_{2-12}) &= 1 \langle c_1, c_2 \rangle [c_1 \in C1, c_2 \in C2] \end{aligned}$

Now let us consider the ODE from the point of view of P_{0-1} (i.e. 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 expression such instances are partitioned into four disjoint sets. Let us analyse a representative instance Trbelonging to each of these sets in order to determine their enabling degree.

Any instance $Tr = \langle T, c_2, c_1 \rangle$ of T, with $c_1, c_2 \in C_1, c_2 \neq c_1$, representing one of the instances in $\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]$ withdraws 1 token $c_2 \in C_1$ from P_{0-1} and 2 tokens $c_1 \in C_1$ from P_{0-1} plus 1 token $c_2 \in C_1$ from P_{1-1} . This is obtained by composing $I(P_{0-1}, T)$ and $I(P_{1-1}, T)$ with the representative tuple Tr. The enabling degree of $\langle T, c_2, c_1 \rangle$ in set $\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]$ is:

$$min(x(P_{0-1})/1, x(P_{0-1})/2, x(P_{1-1})/1) = min(x(P_{0-1})/2, x(P_{1-1})/1)$$

There are $|\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]| = |C_1| - 1$ instances of T of this kind.

Table 3: Computation of enabling degree of T for the ODE of P_{0-1}

Tr = representative tuple of $\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]$				
$Tr = \langle c_2, c_1 \rangle [c_1 \in C_1, c_2 \in C_1, c_2 \neq c_1]$				
$\overline{I[P_{0-1},T] \circ Tr} = (1\langle c_1 \rangle + 2\langle c_2 \rangle)[c_1 \neq c_2, c_1 \in C_1, c_2 \in C_1]$				
$I[P_{1-1}, T] \circ Tr = 1 \langle c_1 \rangle [c_1 \neq c_2, c_1 \in C_1, c_2 \in C_1]$				
$I[P_{0-2},T] \circ Tr = \emptyset$				
$I[P_{1-2},T] \circ Tr = \emptyset$				
Tr = representative tuple of $\langle c_1, S - c_1 \cap S_{C_1} \rangle [c_1 \in C_1]$				
$Tr = \langle c_1, c_2 \rangle [c_1 \in C_1, c_2 \in C_1, c_1 \neq c_2]$				
$I[P_{0-1}, T] \circ Tr = (2\langle c_1 \rangle + 1\langle c_2 \rangle)[c_1 \neq c_2, c_1 \in C_1, c_2 \in C_1]$				
$I[P_{1-1}, T] \circ Tr = 1 \langle c_2 \rangle [c_1 \neq c_2, c_1 \in C_1, c_2 \in C_1]$				
$I[P_{0-2},T] \circ Tr = \emptyset$				
$I[P_{1-2},T] \circ Tr = \emptyset$				
Tr = representative tuple of $\langle c_1, S_{C_2} \rangle [c_1 \in C_1]$				
$Tr = \langle c_1, c_2 \rangle [c_1 \in C_1, c_2 \in C_2]$				
$I[P_{0-1}, T] \circ Tr = 2\langle c_1 \rangle [c_1 \in C_1, c_2 \in C_2]$				
$I[P_{1-1},T] \circ Tr = \emptyset$				
$I[P_{0-2}, T] \circ Tr = 1 \langle c_2 \rangle [c_1 \in C_1, c_2 \in C_2]$				
$I[P_{1-2}, T] \circ Tr = 1 \langle c_2 \rangle [c_1 \in C_1, c_2 \in C_2]$				
Tr = representative tuple of $\langle c_1, c_1 \rangle [c_1 \in C_1]$				
$\mathrm{Tr} = \langle c_1, c_1 \rangle [c_1 \in C_1]$				
$I[P_{0-1},T] \circ Tr = 3\langle c_1 \rangle [c_1 \in C_1]$				
$ I[P_{1-1},T] \circ Tr = 1 \langle c_1 \rangle [c_1 \in C_1]$				
$\left I[P_{0-2},T]\circ Tr=\emptyset\right.$				
$I[P_{1-2},T] \circ Tr = \emptyset$				

Table 4: Computation of enabling degree of T for the ODE of P_{0-2}

$Tr = \langle c_2, c_1 \rangle [c_1 \in C2, c_2 \in C_1]$
$I[P_{0-1}, T] \circ Tr = 2 \langle c_2 \rangle [c_1 \in C_2, c_2 \in C_1]$
$I[P_{0-2}, T] \circ Tr = 1 \langle c_1 \rangle [c_1 \in C_2, c_2 \in C_1]$
$I[P_{1-1},T] \circ Tr) = \emptyset$
$I[P_{1-2}, T] \circ Tr = 1 \langle c_1 \rangle [c_1 \in C_2, c_2 \in C_1]$

Table 5: Computation of enabling degree of T for the ODE of P_{1-1}

Any instance $Tr = \langle T, c_1, c_2 \rangle$ with $c_1, c_2 \in C_1, c_2 \neq c_1$, representing one of the instances in $\langle c_1, S - c_1 \cap S_{C_1} \rangle [c_1 \in C_1]$ withdraws 2 tokens $c_1 \in C_1$ from P_{0-1} , 1 token $c_2 \in C_1$, with $c_2 \neq c_1$ from $P_{0,1}$, 1 token $c_1 \in C_1$, with $c_2 \neq c_1$ from P_{1-1} . Its enabling degree is:

$$min(x(P_{0,1})/1, x(P_{0,1})/2, x(P_{1,1})/1) = min(x(P_{0,1})/2, x(P_{1,1})/1)$$

There are $|\langle c_1, S - c_1 \cap S_{C_1} \rangle |c_1 \in C_1|| = |C_1| - 1$ instances of T of this kind.

Any instance $\langle T, c_1, c_2 \rangle$ with $c_1 \in C_1, c_2 \in C_2$ belonging to $\langle c_1, S_{C_2} \rangle [c_1 inC_1]$ withdraws 2 tokens $c_1 \in C_1$ from P_{0-1} , 1 token $c_2 \in C_2$ from P_{0-2} , 1 token $c_2 \in C_2$ from P_{1-2} . Its enabling degree is:

$$min(x(P_{0,1})/2, x(P_{0,2})/1, x(P_{1,2})/1)$$

There are $|\langle c_1, S_{C_2} \rangle [c_1 \in C_1]| = |C_2|$ instances of T of this kind

Instance $\langle T, c_1, c_1 \rangle$ belonging to $\langle c_1, c_1 \rangle [c_1 \in C_1]$ withdraws 3 tokens $c_1 \in C_1$ from P_{0-1} and 1 token of the same color from P_{1-1} . Its enabling degree is:

$$min(x(P_{0,1})/3, x(P_{1,2})/1)$$

There is only one transition instance of this kind.

The composition operations leading to the above results are summarized in Table 3. Hence, the ODE for $P_{0,1}$ is:

$$\begin{aligned} \frac{dx(P_{0,1}:c\in C_1)}{dt} &= -\omega_1 1*|C_1-1|*\min(x(P_{0,1})/2,x(P_{1,1})/1)\\ &-\omega_2 2*|C_1-1|*\min(x(P_{0,1})/2,x(P_{1,1})/1)\\ &-\omega_3 2*|C_2|*\min(x(P_{0,1})/2,x(P_{0,2})/1,x(P_{1,2})/1)\\ &-\omega_4 3*1*\min(x(P_{0,1})/3,x(P_{1,1})/1)\end{aligned}$$

where:

 $\omega_1 = \omega(T, c', c: c, c' \in C_1 \land c \neq c'), \ \omega_2 = \omega(T, c, c': c, c' \in C_1 \land c \neq c'), \\ \omega_3 = \omega(T, c, c': c \in C_1, c' \in C_2), \ \omega_4 = \omega(T, c, c: c \in C_1) \\ \text{Let us consider place } P_{0,2} \text{: its ODE contains only one term deriving from the single term of } \mathcal{R}() \text{ (of size } |C_1|); \text{ all instances represented by } \mathcal{R}() \text{ have same}$

single term of $\mathcal{K}()$ (of size $|C_1|$); all instances repenabling degree:

$$\frac{dx(P_{0,2}:c\in C_2)}{dt} = -|C_1|\omega(T,c'\in C_1,c\in C_2)\min(x(P_{0,1})/2,x(P_{0,2})/1,x(P_{1,2})/1)$$

Indeed $RBm(T, P_{0-2})$ contains only one term $1\langle S_{C1}, c_1 \rangle [c_1 \in C2]$ (of size $|C_1|$), its representative tuple is $Tr = \langle c_2, c_1 \rangle [c_1 \in C2, c_2 \in C_1]$. Composing the arc functions of T's input places with Tr we get the expressions in Table 4. From those expressions it appears that is not necessary to partition the set of instances represented by $\langle S_{C1}, c_1 \rangle [c_1 \in C2]$ since all its instances have the same enabling degree $min(x(P_{0,1})/2, x(P_{0,2})/1, x(P_{1,2})/1)$.

Let us now consider place $P_{1,1}$, in this case the result of the computation of $\mathcal{R}()$ needs to be rewritten to reflect two cases with different enabling degree, namely

 $c_1 = c_2$ and $c_1 \neq c_2$ in both cases $c_1, c_2 \in C_1$. Let us consider an instance $\langle T, c_1, c_2 \rangle$ with $c_1 \in C_1, c_2 \in C_1$) representative of set $\langle S_{C_1}, c_1 \rangle [c_1 \in C_1]$. Table 5 summarizes the result of the composition of the input arc function and the representative tuple for each input place of T.

The first line refers to place $P_{0,1}$ and the result comprises three terms: the first two have got the same guard and correspond to instances $\langle T, c_1, c_2 \rangle$ with $c_1 \neq c_2$), their enabling degree is $x(P_{0,1})/2$ (which is the $min(x(P_{0,1})/1, x(P_{0,1})/2)$ where the values dividing $x(P_{0,1})$ are the coefficients of the two terms with common guard $[c_1 \neq c_2, c_1 \in C_1, c_2 \in C_1]$). The third term correspond to instances $\langle T, c_1, c_2 \rangle$ with $c_1 = c_2$, their enabling degree is $x(P_{0,1})/3$, again the denominator 3 is the coefficient of the tuple with guard $[c_1 = c_2, c_1 \in C_1]$.

Hence the tuple $\langle S_{C_1}, c_1 \rangle [c_1 \in C_1]$ must be split in $\langle c_1, c_1 \rangle [c_1 \in C_1]$ 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.

Considering the other input place $P_{0,2}$ the composition of $I(P_{0,2})$ with the representative tuple results in \emptyset , similarly for $P_{1,2}$.

The third line in Table 5 refers to place $P_{1,1}$ as expected it does not require any further refinement of the set of T instances and the contribution of $P_{1,1}$ to the computation of the enabling degree is independent on the fact that the two elements of the T instance are equal or different.

Hence $\langle T, c_1, c_1 \rangle$ with $c_1 \in C_1$ has enabling degree $min(x(P_{0,1})/3, x(P_{1,1})/1)$ while $\langle T, c_1, c_2 \rangle$ belonging to set $\langle S - c_1 \cap S_{C_1}, c_1 \rangle [c_1 \in C_1]$ has enabling degree: $min(x(P_{0,1})/2, x(P_{1,1})/1).$

In conclusion the ODE for
$$P_{1,1}$$
 is:

$$\frac{dx(P_{1,1}:c\in C_1)}{dt} = -(|C_1| - 1)\omega(T, c', c: c' \neq c,)\min(x(P_{0,1})/2, x(P_{1,1})/1) - \omega(T, c, c: c\in C_1)\min(x(P_{0,1})/3, x(P_{1,1})/1)$$

Let us now consider $P_{1,2}$: here we do not need to separate different cases, and the equation contains only one term as $\mathcal{R}(T, P_{1,2})$; observe that the size of the tuple $\langle S_{C1}, c_1 \rangle [c_1 \in C2]$ is $|C_1|$ and the representative tuple $Tr = \langle c_2, c_1 \rangle [c_1 \in C2, c_2 \in C_1]$ is the same already considered for place P_{0-2} and hence it is possible to refer to Table 4 for the computation of the enabling degree of this set of Tinstances. In conclusion the ODE for place P_{1-2} is:

$$\frac{dx(P_{1,2}:c\in C_2)}{dt} = -|C_1|\omega(T,c',c:c\in C_2,c'\in C_1)\min(x(P_{0,1})/2,x(P_{0,2})/1,x(P_{1,2})/1)$$

Concerning places $P_{2,12}$, $P_{2,11eq}$ and $P_{2,11neq}$, function \mathcal{A} returns a cardinality one tuple (so there is no need to find a representative tuple for the enabling degree evaluation). Concerning the computation of the enabling degree, it is sufficient to follow the same procedure already considered for the other places (see the three tables above); we report the corresponding ODEs here:

$$\frac{dx(P_{2,12}:c' \in C_1, c \in C_2)}{dt} = +\omega(T, c', c: c' \in C_1, c \in C_2)min(x(P_{0,1})/2, x(P_{0,2})/1, x(P_{1,2})/1)$$

$$\frac{dx(P_{2,11eq}:c' \in C_1, c \in C_1, c = c')}{dt} = +\omega(T, c', c: c' \in C_1, c \in C_1, c = c')min(x(P_{0,1})/3, x(P_{1,1})/1)$$

$$\frac{dx(P_{2,11neq}:c' \in C_1, c \in C_1, c \neq c')}{dt} = +\omega(T, c', c: c' \in C_1, c \in C_1, c \neq c')min(x(P_{0,1})/2, x(P_{1,1})/1)$$

Experimental results 5

In this section we report some experimental results showing the effectiveness of the proposed method on the Botnet example. All the experiments are performed using a prototype implementation which combines: 1) GreatSPN [1] to draw the model and to generate R scripts encoding the ODE systems derived by the unfolded net; 2) SNexpression [5], a java tool, to compute and support the user into the creation of the SODE system applying the approach presented in this paper; 3) the R framework to solve the ODE systems (i.e. *deSolve package*).

The experiments consisted in (1) generating and solving the SODE system using the new approach and (2) unfolding the SSN model to evaluate the cost of this operation, which is the dominating cost of the method [3] as |Loc| increases, and compare the results obtained from the ODE system of the unfolded model against those obtained from the SODE system.

The SODE system and the reduced ODE system generated with the method in [3] have the same number of equations, however the equations may not be identical because the new method may group homogeneous transition instances. In the Botnet model the equation associated with $NoConBot_N$ has one term representing |Loc| - 1 terms in the corresponding equation from the unfolding. As a consequence the number of terms in the ODE system grows linearly with |Loc| while in the SODE it is constant.

We also compared the results obtained by solving the SODE system with those obtained from the ODE system of the unfolded model when the initial marking is that in eq.1: using the R function lsoda() for numerically solving the systems, the difference between the computed solutions is smaller than $1.0e^{-11}$.

Table 6: ODE vs. SODE system size							Table 7: Unfolding time			
Loc	Num. of terms		Mean solution time (sec.)		Le	c	Unfolding (sec.)			
	ODE	ODE/SODE	ODE	SODE		0	2.484			
1	42 (11 eq.)	2.511(1.57)	0.3790	0.0846	5	0	13.219			
10	600 (110 eq.)	21.43(15.7)	38.8110	0.2381	10	00	43.949			
20	1600 (220 eq.)	57.14(31.43)	572.1798	0.2920	15	60	320.979			
50	7000 (550 eq.)	250(78.58)	> 4h	0.2479	20	00	out of memory			

Table 6: ODE vs. SODE system size

The SODE system comprises 7 equations (the number of places in the partially unfolded net is 16 but 9 of them are always empty) and the total number of terms is 28. The procedure to derive the seven SODE from the net structure takes slightly less than one second (including the initial partial unfolding) and does not depend on |Loc|. In Table 6 the number of terms and equations of the ODE system obtained from the unfolded net and the reduction ratio achieved when the SODE system is adopted are shown. The fourth and fifth column contain the execution time required to solve the ODE and the SODE system on a 2.50GHz Intel i3-3120M processor with 4GB of RAM. In Table 7 the time required by the unfolding step is shown as a function of |Loc|; it was not possible to generate the unfolded model for |Loc| = 200 for insufficient memory. From these results we can conclude that the SODE approach is effective and overcomes the limitations of the methods that require the complete unfolding.

6 Conclusions and future work

In this paper we have proposed a new approach for generating a reduced set of ODE approximating the dynamic behavior of a SSN model: this is based on the observation that, due to the model symmetries, groups of *equivalent* equations, generated from the unfolded model could be substituted by a unique representative [3] so that the reduced system could be solved more efficiently.

The novelty of the present paper consists in the ability to automatically derive a Symbolic ODE for each group of *equivalent* ODE without ever computing the (complete) unfolding of the SSN. The new method is based on a recently developed extension of a symbolic calculus for the computation of SSN structural properties and its implementation in the SNexpression tool. In the paper the steps required to generate the system of SODE are defined in details. Some preliminary experimental results are reported to compare the new method with a previous method based on the model unfolding. The results have been obtained through a prototype implementation which combines different tools as GreatSPN, SNexpression and the R framework. The performance improvement observed on a relatively simple example may lead to substantial saving in more complex cases with good symmetric structure (large color classes with a few static subclasses). The complete implementation of the whole automatic procedure for generating the system of SODE of a SSN model is in progress.

The proposed method relies on the specific way of modeling symmetric systems provided by the SSN formalism. It is not straightforward to extend it to other formalisms that allow to express symmetries at the level of the model syntax, in some cases this may be achieved by showing a correspondence between formalism constructs: this is an interesting topic for future work.

Another foreseen evolution is to extend our approach for cases in which the deterministic approximation is not suited. In particular we will investigate how to combine SSN formalism with the diffusion approximation proposed by Kurtz [8] in which the deterministic process is replaced by the Ito's process.

Acknowledgments This work is original and the contribution of G. Franceschinis was supported by the Università del Piemonte Orientale. The work of M. Beccuti was supported by Fondazione CRT for the project Experimentation and study of models for the evaluation of the performance and the energy efficiency of the Competence Center for Scientific Computing at the Università di Torino.

References

- S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, and G. Franceschinis. The GreatSPN tool: recent enhancements. ACM SIGMETRICS Perf. Evaluation Review, Special Issue on Tools for Performance Evaluation, 36(4):4–9, 2009.
- M. Beccuti, E. Bibbona, A. Horvath, R. Sirovich, A. Angius, and G. Balbo. Analysis of Petri net models through stochastic differential equations. In Proc of the Int. Conf. on Application and Theory of Petri Nets and other Models of Concurrency, ICATPN'14, volume 8489 LNCS, pages 273–293. Springer, 2014.
- 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.
- L. Capra, M. De Pierro, and G. Franceschinis. A high level language for structural relations in well-formed nets. In G. Ciardo and P. Darondeau, editors, *Int. Conf.* on Applications and Theory of Petri Nets 2005, pages 168–187. Springer, 2005.
- L. Capra, M. De Pierro, and G. Franceschinis. A tool for symbolic manipulation of arc functions in symmetric net models. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '13, pages 320–323, Torino, Italy, 2013. ICST, Brussels, Belgium.
- L. Capra, M. De Pierro, and G. Franceschinis. Computing structural properties of symmetric nets. In Proc. of the 15th International Conference on Quantitative Evaluation of Systems, QEST 15, Madrid, ES, 2015. IEEE CS.
- 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.
- T. G. Kurtz. Strong approximation theorems for density dependent Markov chains. Stochastic Processes and their Applications, 6(3):223–240, 1978.
- F. Liu, M. Heiner, and D. Gilbert. Coloured Petri nets for multilevel, multiscale and multidimensional modelling of biological systems. *Briefings in bioinformatics*, 11 2017.
- M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions* on Computers, 31(9):913–917, 1982.
- E. V. Ruitenbeek and W. H. Sanders. Modeling peer-to-peer botnets. In Proc. of the 5th International Conference on Quantitative Evaluation of Systems), QEST 08, pages 307–316, Washington, DC, USA, 2008. IEEE CS.
- M. Silva. Individuals, populations and fluid approximations: A Petri net based perspective. Nonlinear Analysis: Hybrid Systems, 22:72 – 97, 2016.
- 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.