Computing Structural Properties of Symmetric Nets

Author's pre-print.
The final publication is available at Springer via
http://dx.doi.org/10.1007/978-3-319-22264-6_9

Computing structural properties of Symmetric Nets

Lorenzo Capra[1], Massimiliano DePierro[2], and Giuliana Franceschinis[3]

[1] Dipartimento di Informatica, Univ. di Milano, Italy
[2] Dipartimento di Informatica, Univ. di Torino, Italy
[3] DiSIT, Univ. del Piemonte Orientale, Alessandria, Italy.

Abstract. Structural properties of Petri Nets (PN) have an important role in the process of model validation and analysis. When dealing with high level Petri nets (HLPN) structural analysis still poses many problems and often tools go through the unfolding of the HLPN model and apply PN structural analysis techniques to the unfolded model: with this approach the symmetries present in the models are completely ignored and cannot be exploited. Structural properties of HLPN can be defined as relations among node instances using symbolic and parametric expressions; the computation of such expressions from the model structure and annotations requires the development of a specific calculus, as the one proposed in the literature for Symmetric Nets (SN). When dealing with Stochastic SN (SSN), comprising stochastic timed transitions and immediate transitions, structural analysis becomes a fundamental step in net-level definition of probabilistic parameters. Moreover some structural relations allow to automatize the derivation of symbolic Ordinary Differential Equations for the solution of SSN models with huge state space. The goal of the present paper is to summarize the language defined to express SNs' structural relations, to complete the formalization of some interesting structural properties as expressions of the calculus, and to provide examples of their use. The algorithms required to support the calculus for symbolic structural relations computation have been recently completed and implemented in a tool called SNexpression.

1 Introduction

Structural properties of Petri Nets have an important role in the process of model validation and analysis, since they can answer interesting questions on the model potential behavior, that can also be exploited to improve the efficiency of state space methods.

When dealing with high level Petri nets it is desirable to exploit the possibilities offered by this class of formalisms, among which the ability to represent systems in a more parametric way and to make regularities in the model structure explicit. Some HLPN formalism have been devised to make some form of symmetry easier to exploit at the level of the analysis: an example of formalism in this class is Symmetric Nets (SN) [6].

While structural properties of PNs usually express relations between nodes in the model, when turning to HLPN one wants to express relations between *node instances*, possibly using symbolic and parametric expressions [8]. The computation of such expressions from the model structure and annotation requires the development of a calculus like that proposed in [7,4]. A similar approach has been applied to HLPN reduction in [9].

In this paper a set of structural properties for the SN formalism are defined: these employ a number of functional operators (transpose, difference, composition, etc.) that allow the arc functions and transition guards of the model to be properly combined. In order to make the calculus closed with respect to its operators, a language for the (symbolic) structural expressions has been defined, with a syntax that extends that of SN arc functions. The calculus has been recently implemented in the SNexpression tool [5] and applied to a number of interesting cases. The implemented calculus embeds a new, efficient algorithm for handling the composition operator (see [3]).

The goal of the present paper is to summarize the language defined to express the model structural properties, and to complete the specification of a number of interesting structural properties, providing several examples of their use and showing their usefulness. All the computations presented in the paper have been carried out using the SNexpression tool which implements the basic calculus used for deriving the structural properties expressions, but also directly supports the computation of structural properties on a SN model (automatically producing the expressions from the model specification, and applying the operators to obtain the result). There are several useful applications for these properties, e.g., the derivation of the *Extended Conflict Sets* (ECS), needed for the net level specification of quantitative parameters for probabilistic conflict resolution in stochastic SNs, in analogy with what is done for Generalized Stochastic Petri nets [1] (the stochastic SN's unfolding).

The paper continues with a section (Sec. 2) providing all the definitions and notations needed in the sequel. In Sec. 3 the structural properties that can be expressed through the language just introduced are defined and some contexts where they can be helpful are illustrated. In Sec. 4 the computation of the properties just introduced on a set of examples is illustrated: the rewriting rules that lead from the initial formulae (of Tab. 1) to the final result belonging to language $\mathcal{L}$ have been implemented in the SNexpression tool, which has been used to obtain all the results shown in this paper. Finally, Sec. 5 summarizes the main contribution of this work and outlines ongoing and future developments.

2 Basic definitions and notation

In this section the SN formalism [6] is quickly recalled, then the definitions and the notation needed in the next sections are introduced.

2.1 The Symmetric Nets formalism

The SN formalism is introduced through an example. The focus is on the color inscriptions appearing in the model, which are the basis of the calculus intro-

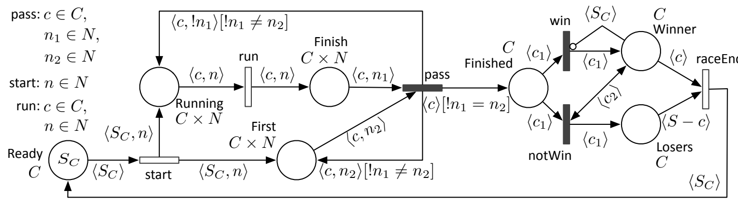

Fig. 1. The SN model of a relay race.

duced later. Let us consider the model in Fig. 1 describing the dynamics of a relay race. The net structure is a bipartite graph whose nodes are places (circles) and transitions[4] that represent the state variables and the events that cause state changes, respectively.

Places are state variables, characterized by a *color domain* defining the variable type, and expressed as a Cartesian product of *basic color classes* (disjoint and finite non empty sets, denoted with capital letters A, B, ..., Z, which may be partitioned into two or more *static subclasses*, and may be ordered). In this model there are two basic color classes: $C = \{IT, FR, D, ES\}$, encoding the competing teams identifiers, and $N = \{0, 1, 2, 3\}$ encoding the athletes identifiers; N is an ordered class, that is a successor function is defined on it, inducing a circular order among its elements ($succ(i) = (i + 1)\%4$). A pair (2-tuple) $\langle c, n \rangle, c \in C, n \in N$ represents athlete n of team c. Each place can contain a multiset of tuples belonging to its color domain: this is called its *marking*.

Also the transitions have a color domain since they describe parametric events; the parameters are color variables denoted with small letters with a subscript. The letter used for a variable implicitly defines its type which is the color class denoted by the corresponding capital letter; subscripts are thus used to distinguish parameters of same type associated with the same transition. Transition `pass` in the net of Fig. 1 has tree parameters $c : C, n_1, n_2 : N$. Transitions can have guards, expressed in terms of predicates on transition variables. The model evolution in time can be simulated by starting from an initial marking (in the example all colors from class C in place `Ready`), and firing one of the enabled *transition instances*. A transition instance is a pair transition-binding (t, b), where a binding is an assignment of colors to the transition parameters. For instance a possible binding for the parameters of `pass` is $c = IT, n_1 = 3, n_2 = 1$; a binding is valid only if it satisfies the transition guard. Hence a transition color domain corresponds to the set of all possible valid bindings for that transition. The arcs connecting transition t to its input, output and inhibitor places ($^\bullet t$, $t^\bullet$, $^\circ t$) are annotated with functions (denoted by $W^-(p, t)$, $W^+(p, t)$ and $W^h(p, t)$, respectively) whose domain is the transition color domain ($cd(t)$), and that map to (multisets on) the place color domain ($Bag(cd(p))$). Input and inhibitor arcs express the conditions for transition enabling, while input and output arcs define the state change produced by the occurrence (firing) of an enabled transition.

---

[4] In the Stochastic SN formalism white transitions are timed, black transitions are immediate and have priority over the timed ones.

Definition 1 (Guards syntax). *Guards in SN models are boolean expressions whose terms are* basic predicates*: the set of basic predicates is:* $[var1 = var2]$ *(true when the same color is assigned to* $var1$ *and* $var2$*),* $[var1 =!var2]$ *(true when the color assigned to* $var1$ *is the successor of that assigned to* $var2$*),* $[d(var1) = S_{subclass\_id}]$ *(true when the color assigned to* $var1$ *belongs to static subclass subclass_id), and* $[d(var1) = d(var2)]$ *(true when the colors assigned to* $var1$ *and* $var2$ *belong to the same static subclass).*

Definition 2 (Arc functions syntax). *A SN function* W *labeling an arc connecting transition* t *and place* p*, is a mapping* $W : cd(t) \rightarrow Bag(cd(p))$

$$W = \sum_i \lambda_i . T_i[p_i], \quad \lambda_i \in I\!N \tag{1}$$

where the sum is a multiset sum and λ_i *are scalars,* $T_i = \langle f_1, \ldots, f_k \rangle$ *are tuples of class functions, and* p_i *is a guard. Class functions syntax (referring to class* C*, without loss of generality) is:*

$$f_i = \sum_{k=1}^{m} \alpha_k . c_k + \sum_{q=1}^{||C||} \beta_q . S_{C_q} + \sum_{k=1}^{m} \gamma_k . !c_k; \;\; \alpha_k, \beta_k, \gamma_k \in \mathbb{Z} \tag{2}$$

Scalars in (2) must be such that no negative coefficients result from the evaluation of f_i *for any legal variables binding.* $||C||$ *is the size of the static partition of* C*.*

An arc function is a weighted sum of (possibly guarded) tuples of class functions. The allowed class functions are: projection (denoted by a variable name), synchronization/diffusion constant function (denoted by S_{class_id}), successor function (defined only for ordered classes and denoted by ! followed by a variable name). An arc function is evaluated on a given binding of the transition: the value of a tuple is the Cartesian product of the value of its elements. The projection function evaluates to the variable binding, the successor function evaluates to the successor of the variable binding, the diffusion/syncronization function S_{class_id} is constant and evaluates to the whole set of elements in $class\_id$. The expression $S_{class\_id} - var$ maps to all the elements in $class\_id$ except for the binding of variable var. Sometimes it is useful to partition a color class into *static subclasses*, denoted by the class identifier with a numeric subscript. In this case it is also possible to use the diffusion/syncronization function restricted to a static subclass. A guarded tuple is evaluated as follows: if the guard is false (for a given binding) it evaluates to the empty multiset, otherwise its value corresponds to its standard evaluation without the guard.

A transition instance (t, b) is enabled in marking m if for each input place p of t the multiset $W^-(p, t)(b)$ is included in $m(p)$, and for each inhibitor place p, the multiplicity of each color in $W^h(p, t)(b)$ is greater than the multiplicity of the corresponding color in $m(p)$. An enabled transition instance may occur, withdrawing from each input place p the multiset $W^-(p, t)(b)$ and adding into each output place p the multiset $W^+(p, t)(b)$.

Let us exemplify the definitions just introduced on the relay race model: it comprises 6 transitions and 7 places, transition `start` represents the start of the race: only variable n is associated with it, representing the number of the first runner in the race: it is common to all teams. The function on the input arc is $\langle S_C \rangle$, so the transition is enabled when all elements of color class C are in place `Ready`. The function labelling the two output arcs is $\langle S_C, n \rangle$, hence the firing of the instance of `start` with binding $n = i$ produces $|C|$ tokens in places `Running` and `First` *colored* with a pair $\langle ID, i \rangle$. Let us focus on the black transitions in the model, in particular, on transition `pass`, since it contains several features: on the output arcs to place `Finish` and `Running` there are two guarded functions which allow for a conditional behaviour, i.e., when the runner is the last of his team (i.e., the predecessor of the runner who started the race, stored in place `First`) then function $W^+($`pass`,`Finish`$) = \langle c, !n_1 \rangle [!n_1 \neq n_2]$ evaluates to *empty*, while $W^+($`pass`,`Finished`$)$ evaluates to a multiset containing only one occurrence of color $\langle c \rangle$. Also the output arc from `pass` to place `First` is guarded, and produces the single colored token $\langle c, n_2 \rangle$ only when n_1 is not the predecessor of n_2. In the model we can also observe an inhibitor arc, going from place `Winner` to transition `win`, ensuring that only the first team arriving at the end of the race is recorded as winner: the function $W^h(win, Winner) = \langle S_C \rangle$ means no tokens must be present in `Winner`, in order for `win` to be enabled. Finally the synchronization modelled by transition `raceEnd` makes use of function $\langle S_C - c \rangle$ that represents all the teams that have not won the race, i.e., the set of all elements in class C except for the one bound to variable c (the winner).

2.2 The language to express structural properties

Definition 3 (Language $\mathcal{L}$). *Let $\Sigma = \{A, B, \ldots, Z\}$ be the set of (finite and disjoint) basic color classes, and let $\mathcal{D}$ be any color domain built as Cartesian product of classes in Σ, ($\mathcal{D} = A^{e_A} \times B^{e_B} \times \ldots \times Z^{e_Z}, e_* \in \mathbb{N}$). Let $T_i : \mathcal{D} \to Bag(\mathcal{D}')$ and $[g'_i]$ and $[g_i]$ SN standard predicates on $\mathcal{D}'$ and $\mathcal{D}$, respectively .*

The set of expressions:

$$\mathcal{L} = \left\{ F : F = \sum_i \lambda_i.[g'_i] T_i [g_i], \quad \lambda_i \in \mathbb{N}^+ \right\}$$

is the language used to express SN structural relations, where $T_i = \langle f_1, \ldots, f_l \rangle$ are function-tuples formed by class functions f_j, defined in turn as intersections of language elementary functions $\{a, !^k a, S_A, S - a, S - !^k a, \emptyset_A\}$ (projection, k^{th} successor, constant function corresponding to all elements of basic class A, projection/successor complement and the empty function; where A represents any basic class and a any variable of type A).

Language $\mathcal{L}$ defined in Def. 3 actually extends the set of functions used in SN: indeed, predicate $[g'_i]$, called *filter*, is not allowed in SN and permits the elements satisfying the boolean condition g'_i to be selected from the result of the application of $T_i[g_i]$. On the other hand, SN arc functions $W^-(p,t)$, $W^+(p,t)$, $W^h(p,t)$ can be written as elements of $\mathcal{L}$. The calculus we provide defines the following functional operators on $\mathcal{L}$:

F^t	Transpose	$F \cap F'$	Intersection	$\overline{F}$	Support
$F - F'$	Difference	$F + F'$	Sum	$\overline{F} \circ \overline{F'}$	Composition

All operators but composition apply to elements of $\mathcal{L}$ that map to multisets, and whose definition is consistent with the operator semantics. The composition is currently defined on a subset of $\mathcal{L}$ consisting of functions mapping to sets.

In the sequel the term *expression* will be used to indicate formulae that contain language functions and functional operators from the table above. The symbolic calculus is able to *solve* all the considered operators, that means $\mathcal{L}$ is closed w.r.t. them. Appropriate rewriting rules have been defined that simplify an arbitrary expression with operators until an expression in $\mathcal{L}$ is obtained. In some cases we are interested in obtaining an expression where terms are pairwise disjoint (i.e., when the expression is evaluated for any color in its domain, the multisets obtained by evaluating each term are disjoint). Each rewriting rule is based on the algebraic properties of functions appearing as operands.

A detailed description of these rules can be found in [4], where the difference, intersection, transpose operators rewriting rules have been first introduced. In [3] the calculus is completed with the details of composition.

3 Structural properties computation

In this section a number of structural properties will be defined and formalized through expressions in the language introduced in Sec. 2. Examples of computation of these structural properties will be illustrated in Sec. 4.

Let us first define a symbolic relation between nodes in a Symmetric Net.

Definition 4. *[Symbolic relation] Given a binary relation $\mathcal{R}$ between the instances of nodes s and s' of a SN model defined as*

$$\mathcal{R} \subseteq (s \times \mathcal{C}(s)) \times (s' \times \mathcal{C}(s'))$$

its symbolic representation denoted $\mathcal{R}(s, s')$ is a mapping from $\mathcal{C}(s')$ to $2^{\mathcal{C}(s)}$ such that

$$\mathcal{R}(s, s')(c') = \{c : (s, c)\mathcal{R}(s', c')\} \textit{for each } c' \in \mathcal{C}(s')$$

We are interested in deriving symbolic relations between instances of SN node pairs (place-transition, transition-place, transition-transition) in the form of an expression of $\mathcal{L}$. Such relations are derived by properly combining the SN functions appearing on the arcs connecting the nodes.

Tab. 1 provides the formulae showing how each structural relation depends on the arc functions. The calculus partially defined in [4] and completed in [3] allows us to apply transformations defined as rewriting rules that are repeatedly applied to the formulae according to the semantics of the operators appearing in them, until one obtains as a result an expression of language $\mathcal{L}$.

Some ausiliary relations: SbT, SfP, AbT, AtP. The first relations that are introduced here will be used to characterize more complex ones, but can also be used for other interesting applications [2]. They involve a pair of nodes, place and transition, directly connected through one or more arcs:

$$
\begin{aligned}
SbT(p,t) &= \overline{W^-(t,p) - W^+(t,p)}\\
SfP(t,p) &= \overline{W^-(t,p) - W^+(t,p)}^{\,t} = SbT(p,t)^t\\
AbT(p,t) &= \overline{W^+(t,p) - W^-(t,p)}\\
AtP(t,p) &= \overline{W^+(t,p) - W^-(t,p)}^{\,t} = AbT(p,t)^t\\
SC(t,t') &= \textstyle\bigcup_{p\in {}^\bullet t\cap {}^\bullet t'} SfP(t,p)\circ \overline{W^-(t',p)} \ \cup\ \bigcup_{p\in t^\bullet\cap {}^\circ t'} AtP(t,p)\circ \overline{W^h(t',p)}\\
SC(t,t) &= \textstyle\bigcup_{p\in {}^\bullet t} SfP(t,p)\circ \overline{W^-(t,p)} - Id \ \cup\ \bigcup_{p\in t^\bullet\cap {}^\circ t} AtP(t,p)\circ \overline{W^h(t,p)} - Id\\
SCC(t,t') &= \textstyle\bigcup_{p\in t^\bullet\cap {}^\bullet t'} AtP(t,p)\circ \overline{W^-(t',p)} \ \cup\ \bigcup_{p\in {}^\bullet t\cap {}^\circ t'} SfP(t,p)\circ \overline{W^h(t',p)}\\
SME(t,t') &= \textstyle\bigcup_{p\in {}^\bullet t\cap {}^\circ t'} \overline{W^-(t,p)}^{\,t}\circ \overline{W^h(t',p)} \ \cup\ \bigcup_{p\in {}^\circ t\cap {}^\bullet t'} \overline{W^h(t,p)}^{\,t}\circ \overline{W^-(t',p)}\\
&simple
\end{aligned}
$$

Table 1. Structural relations are obtained by properly combining the arc functions through intersection, transpose, sum, difference, support, and composition operators.

$SbT(p,t) : cd(t) \to 2^{cd(p)}$, *Subtracted by Transition*: provides the set of colored tokens that a given instance of t withdraws from p; it is simply defined as (the support of) the *multiset* difference of the function appearing on the input arc and the function appearing on the output arc connecting t and p;

$SfP(t,p) = SbT^t(p,t) : cd(p) \to 2^{cd(t)}$, *Subtracts from Place* (transpose of SbT): given a color of p it provides the set of instances of t that withdraw it;

$AbT(p,t) : cd(t) \to 2^{cd(p)}$, *Added by Transition*: provides the set of colored tokens an instance of t adds into p when it is fired; it is simply defined as (the support of) the *multiset* difference of the function appearing on the output arc and the function appearing on the input arc connecting t and p;

$AtP(t,p) = AbT^t(p,t) : cd(p) \to 2^{cd(t)}$, *Adds to Place* (transpose of AbT): given a color of p it provides the color instances of t that add it into p

Structural Conflict: Two transition instances (t,c) and (t',c') are in conflict in a given marking M if the firing of the former produces a change in state that modifies the enabling condition of the latter, possibly disabling it. The structural conflict relation defines some conditions in the model structure and its annotations, that may lead to an actual conflict in some marking. The symbolic relation $SC(t,t')$ has color domain $cd(t')$ and co-domain $2^{cd(t)}$, so that when applied to a color c' in $cd(t')$ provides the subset of $cd(t)$ identifying the instances (t,c) of t that may disable (t',c'). An instance (t,c) may disable (t',c') either because it withdraws a token from an input place which is shared by the two transitions, or because it adds a token into an output place which is connected to (t') through an inhibitor arc. Let us consider the two cases separately: let $p \in {}^\bullet t' \cap {}^\bullet t$, function $\overline{W^-(t',p)}$ gives the set of colored tokens in p required for the enabling of the instances of t'. Since $SfP(t,p)$ gives the instances of t that withdraw a given colored token from p, then the composition $SfP(t,p)\circ \overline{W^-(t',p)}$ provides the instances of t that may disable a given instance of t' because they require non-disjoint sets of colored tokens in the shared input place p. Similarly for the case of $p \in t^\bullet \cap {}^\circ t'$ function $\overline{W^h(t',p)}$ gives the set of colored tokens in p that may disable t', while $AtP(t,p)$ gives the instances of t that add a given colored token in p, so that $AtP(t,p)\circ\overline{W^h(t',p)}$ provides the instances of t that may disable a given instance of t' because they add in p colored tokens that

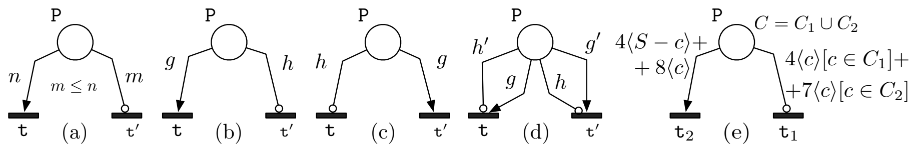

Fig. 2. Structural mutual exclusion patterns

may disable t'. Finally $SC(t, t')$ is obtained by summing up over all common input places and common output-inhibitor places. The complete definition is shown in Tab. 1. Observe that it may be the case that different instances *of the same transition* are in conflict. The same expression can be used in this case, but at the end one must subtract from the set of conflicting instances the instance to which SC is applied: this explains why Id is subtracted. Observe that the SC relation is not symmetric.

Structural Causal Connection: Two transition instances (t, c) and (t', c') are in causal connection in a given marking M if the firing of the former produces a change in state that modifies the enabling condition of the latter, possibly causing its enabling. The structural causal connection relation defines some conditions in the model structure and its annotations, that may lead to an actual causal connection in some marking. The symbolic relation $SCC(t, t')$ has color domain $cd(t')$ and co-domain $2^{cd(t)}$, so that when applied to a color c' in $cd(t')$ provides the subset of $cd(t)$ identifying the instances (t, c) of t that may cause the enabling of (t', c'). In this case we should concentrate on output places of t that are input places for t' and on input places of t that are inhibitor places for t', and in the former case the expression $AtP(t, p) \circ \overline{W^-(t', p)}$ is used, while in the latter case the expression $SfP(t, p) \circ \overline{W^h(t', p)}$ is used. The complete definition is shown in Tab. 1.

Structural Mutual Exclusion: Two transition instances (t, c) and (t', c') are in (structural) mutual exclusion if the enabling of (t', c') in any M implies the fact that (t, c) is not enabled in M, and viceversa. This situation arises when in the net structure a place p exists that is input place for t and inhibitor place for t', and the number of tokens (of any color) required in the input place p for the enabling of t is greater than or equal to the upper bound on the number of tokens (of the same color) in p imposed by the inhibitor arc connecting p and t'.

In a (uncolored) Petri Net (possibly obtained by unfolding an SN) the necessary structural condition for two transitions to be in SME relation is the one depicted in Fig. 2.(a) where t and t' are in SME relation because, with respect to place P, the condition for the enabling of t in marking M is $M(P) \ge n$ while the condition for the enabling of t' in M is $M(P) < m$; since $((M(P) \ge n) \wedge (m \le n)) \Rightarrow not(M(P) < m)$ and $((M(P) < m) \wedge (m \le n)) \Rightarrow not(M(P) \ge n)$ the two transitions are indeed in SME relation, and the relation is symmetric.

When turning to SN models, we are looking for SME conditions on transition instances. The patterns that may lead to mutual exclusion of t and t' instances are depicted in Figs. 2.(b), 2.(c), and 2.(d).

Let us define a symbolic relation $SME(t, t') : cd(t') \to 2^{cd(t)}$ defined as follows: $SME(t, t')(c') = \{c \in cd(t) : (t, c)SME(t', c')\}$ i.e. a function giving the set of instances of t that are surely disabled in any marking where instance (t', c') is enabled. If all functions on input and inhibitor arcs were all functions mapping onto sets (i.e. on multisets with all multiplicities ≤ 1) than the SME relation could be computed by means of the expression shown in Tab. 1 (in the table it is called $SMEsimple$, because of the restriction on the functions W^- and W^h involved in the expression). The expression accounts for any possible structural pattern, including the general situation like the one depicted in Fig. 2.(d). The expression for $SMEsimple(t, t')$ is the union of two parts, the former considers the case in which t' is connected to P through an inhibitor arc, while t is connected to P via an input arc; the latter considers the case in which t' is connected to P through an input arc and t instead through an inhibitor arc. Of course the two situations may overlap (as in Fig. 2.(d)), moreover this may apply also when t and t' are the same transition (some instances of t may well be in mutual exclusion with other instances of the same transition, as shown in next section).

The expression $\overline{W^h(t,p)}^t \circ \overline{W^-(t',p)}$ (with $p \in^\circ t \cap^\bullet t'$) applied to a given color $c \in cd(t')$ first derives the set of colored tokens withdrawn from p by t', then by applying the transpose of $W^h(t,p)$ to this set, one obtains all t instances that would be inhibited by any color in such set. The other expression $\overline{W^-(t,p)}^t \circ \overline{W^h(t',p)}$ instead (with $p \in^\bullet t \cap^\circ t'$) applied to a given color $c \in cd(t')$ gives the set of colors that *should not appear in* p for t' to be enabled; by applying the transpose of $W^-(t,p)$ to this set one obtains all instances of t that require those colors in p to be enabled.

Let us now consider a more general case, where the input and inhibitor arcs are labelled with functions that map onto proper multisets. We first need to introduce an operator useful to define the SME relation in the general setting. Let $g : D_g \to Bag(D)$ and $h : D_h \to Bag(D)$ be two arc functions with same codomain, the comparison function $g \trianglerighteq h$ is defined as:

$$g \trianglerighteq h(c) = \{c' \in D_g : \exists d \in D, g(c')(d) \geq h(c)(d) > 0\} \forall c \in D_h$$

For any color $c \in D_h$ this function gives the set of all colors $c' \in D_g$ such that $g(c') \in Bag(D)$ contains at least one element d, also contained in $h(c)$, whose multiplicity is greater in $g(c')$ than in $h(c)$.

If we consider now a situation where g is the arc function $W^-(t,p) : cd(t) \to cd(p)$ associated with the input arc from p to t and h is the arc function $W^h(t',p) : cd(t') \to cd(p)$ associated with the inhibitor arc from p to t', then $SME_H(t,t',p) = g \trianglerighteq h$. In words: given an instance (t',c) of t' it computes the set of instances of t which are surely disabled when (t',c) is enabled, because of place p, which is inhibitor for t' and input for t. If we are in the situation of Fig. 2.(d), where there is another pair of input-inhibitor arcs departing from p and directed towards t' and t, respectively, we can take the transpose of function $SME_H(t',t,p)$ to obtain another type of SME relation $SME_I(t,t',p) = SME_H(t',t,p)^t$ which, given an instance (t',c) of t' , returns the set of instances of t which are surely disabled if (t',c) is enabled, because of place p which is input for t' and inhibitor for t. Finally

$$SME(t,t') = \bigcup_p SME_H(t,t',p) + SME_I(t,t',p) = \bigcup_p SME_H(t,t',p) + (SME_H(t',t,p))^t$$

Let's define an algorithm implementing the computation of $SME_H(t,t',p)$: it is based on the representation of functions $W^h(t',p), W^-(t,p), W^-(t',p), W^h(t,p)$ in the form of weighted sums of pairwise disjoint terms such that each term is in the form $[b'_i]\langle f_1,\dots,f_l\rangle[b_i]$, where functions f_i are intersections of language elementary functions (see Def. 3), and b'_i, b_i are standard predicates. In the sequel, let g be the function labelling the input arc (of t or t') and h the function labelling the inhibitor arc (of t or t'). They are in the form:

$$g^t = \sum_{i=1}^{K} m_i.G_i^t \quad h = \sum_{i=1}^{K'} n_i.H_i$$

Since the terms G_i are disjoint (and hence so are the terms G_i^t) and the terms H_j are disjoint we can compare directly the weights of pairs G_i^t, H_j without instantiating the functions on a specific colour. We have mutual exclusion when $m_i \geq n_j$, hence $SME_H(t,t') = \bigcup_{i,j:m_i \geq n_j} G_i^t \circ H_j$.

procedure $\mathrm{SME}_H(t,t',p)$ **:**

Let: $g = W^-(t,p)$ and $h = W^h(t',p)$

$g^t = \sum_{i=1}^{K} m_i.G_i^t,\ G_i^t \cap G_j^t = \emptyset,\ \forall i \neq j; \quad h = \sum_{i=1}^{K'} n_i.H_i,\ H_i \cap H_j = \emptyset,\ \forall i \neq j$

$R = \emptyset$

for each $i = 1,\dots,K$ **do**

  for each $j = 1,\dots,K'$ **do**

    if $m_i \geq n_j$

      $R = R \cup G_i^t \circ H_j$

return R

Let us apply the algorithm to the example in Fig. 2.(e) which corresponds to the pattern in Fig. 2.(b) with `t2` corresponding to `t` and `t1` corresponding to `t'`. The color class C has two static subclasses, C_1 and C_2. The functions on the arcs, both with domain and codomain C, are $g = 4\langle S-c\rangle + 8\langle c\rangle$ and $h = 4\langle c\rangle[c \in C_1] + 7\langle c\rangle[c \in C_2]$ observe that in both functions the two terms of the sum are disjoint. In this case the (multiset) transpose of g is equal to g itself $g^t = 4\langle S-c\rangle + 8\langle c\rangle$. In order to compute $SME_H(\mathtt{t2},\mathtt{t1},\mathtt{P})$ we compare the coefficients of the two terms in g^t (g_1^t and g_2^t) and those of h (h_1 and h_2): g_1^t has coefficient 4, which is equal to that of h_1, while g_2^t has coefficient 8, greater than those of both h_1 and h_2. Hence $SME_H(\mathtt{t2},\mathtt{t1},\mathtt{P}) = g_1^t \circ h_1 + g_2^t \circ h_1 + g_2^t \circ h_2 = \langle S-c\rangle[c \in C_1] + \langle c\rangle[c \in C_1] + \langle c\rangle[c \in C_2]$ after some simplifications we obtain: $SME_H(\mathtt{t2},\mathtt{t1},\mathtt{P}) = \langle S-c\rangle[c \in C_1] + \langle c\rangle$. Indeed, if `t1` is enabled for a given binding $c = a$ in C_1 there are less than 4 tokens of that color in `P`, hence all instances of `t2` with binding $c \neq a$ are not enabled because they need at least 4 tokens of color a in `P`; if `t1` is enabled for a given binding $c = b$ in C_2 there must be less than 7 tokens of that color in `P`, hence all instances of `t2` with same binding are not enabled since they require 8 tokens of that color. In this simple example $SME(\mathtt{t2},\mathtt{t1}) = SME_H(\mathtt{t2},\mathtt{t1},\mathtt{P})$. Observe that $SME(\mathtt{t1},\mathtt{t2}) = SME(\mathtt{t2},\mathtt{t1})^t = \langle S - c \cap S_{C_1}\rangle[c \in C_1] + \langle S_{C_1}\rangle[c \in C_2] + \langle c\rangle$: the interpretation of this result is left to the reader.

Extended Conflict Sets (ECS): In Stochastic SN (SSN) models, an extension of SNs comprising timed transitions (with exponentially distributed delays) and immediate transitions (firing in 0 time), structural conflict relations are used to identify *at the net level* subsets of immediate transitions whose firing order may influence the relative probability of alternative immediate transition firing sequences.[5] Immediate transitions that are in different ECSs are instead independent and can be fired in any order. ECS computation requires to introduce the Symmetric and Transitive closure of the SC relation: this new relation is denoted SSC^*. The first step to compute the desired relation consists in making the SC relation Symmetric: $SSC(t,t') = SC(t,t') \cup SC^t(t',t)$. The transitive closure is computed iteratively as follows: let us consider matrix M^0 whose rows and columns are indexed on the (immediate) transitions $t_i \in I$, and such that $M^0(t_i,t_j) = SSC(t_i,t_j)$. A family $\{M^1, M^2, ..., M^n\}$ of matrices can be derived by applying the following transformation: $M_{i+1}(t_l,t_j) = M^i \cup \bigcup_{t_k \in I} M^i(t_l,t_k) \circ M^i(t_k,t_l)$. Intuitively each iteration adds into element (t_l,t_j) of the matrix new (farther) indirect connections between t_l and t_j established by transitivity through an intermediate transition t_k. The iterative process eventually reaches a fixed point: $M^{n+1} = M^n$, and the elements of M^n contain the information needed to symbolically express all the ECS of the model (the upper bound where the iterations necessarily stop if it did not stop earlier, is a matrix full of functions in the form $\langle S_{C_j}, S_{C_i}, ..., S_{C_n} \rangle$). A few examples of ECS computation are shown in Sec. 4.

4 SNexpression at work

In this section the structural property computation algorithms, as implemented in the SNexpression tool [5] are applied to three examples. All the results described in this section are obtained using SNexpression, a tool implementing the calculus presented in the previous section: it provides also direct computation of a few structural properties of SN models. With respect to the version presented in [5] the tool now has some new features, in particular the extension of all operators, except composition, to multisets. The tool can be downloaded from: `http://www.di.unito.it/~depierro/SNex/`. Tab. 2 in the appendix contains a summary of the commands syntax accepted by the SNexpression user interface and used to illustrate the examples in this section.

4.1 The relay race model

Let us check some structural properties of the relay race model of Fig. 1 introduced in Sec. 2: for example let us consider the structural causal connection between transitions `start` and `run`; there is only one place that connects the two transitions, i.e. `Running`. According to Tab. 1 the formula for computing this property is $AtP(start, Running) \circ \overline{W^-(run, Running)}$, and substituting AtP with its definition we obtain $Atp(start, run) = \overline{W^+(start, Running)}^t$. The

[5] This is due to the way conflicts among enabled immediate transitions are probabilistically solved, by normalization of their weights to obtain the probabilities.

tool can be exploited for the calculation of the property either by submitting directly the following commands:

```
f := @N <S_C,n>   g := @C,N <c,n>
sf(f') => <n>   this expression corresponds to AtP(start,Running)
s(f'.g) => <n>
```

(here the symbol => is used to indicate the result returned by the tool, the symbol := instead allows to assign expressions to symbols, the syntax $sf(expression)$ corresponds to a request to apply the operators), or reading the net (prepared in an appropriate textual format) and submitting the command for structural causal connection computation:

```
load "relayrace.sn"
SCC(start,run,Running) => <n>
```

The meaning of this result is: given an instance of transition `run`, e.g. with binding $c = IT, n = 1$, the instance of `start` that may enable is that with $n = 1$. Indeed, only when the race starts it happens that `run` becomes enabled due to the firing of `start`, since the next instance, until the race ends, are instead activated by the firing of transition `pass`:

```
SCC(pass,run,Running) = <c_1,!-1n_1,S-n_1>
```

The instances of `pass` (whose variables are c, n_1 and n_2) that can enable a given instance of `run` (variables c and n) can only be those involving the same team (c has the same value in `pass` and `run`) and the variables n_1 and n_2 of `pass` are the predecessor of variable n of `run`, and any element of N but n, respectively.

Let us now evaluate a situation of potential *confusion* when the relay race model is interpreted as a Stochastic SN, with stochastically timed transitions, the white rectangles, and immediate transitions, the black ones. The presence of confusion in such a kind of model is a symptom of an underspecified behaviour (from the point of view of a probabilistic characterization of conflicts resolution). In this case the potential *confusion* involves transitions `pass` and `win`: the folded structure of the high level model hides the structural conflict existing among the instances of `win` due to the presence of both an output arc and an inhibitor arc connecting this transition and place `Winner`. We want to compute the auto-conflicts of transition `win`: $SC(win, win, Winner) = AtP(win, Winner) \circ \overline{W^h(win, Winner)} - Id$

```
SC(win,win,Winner) =>  <S-c>
```

Indeed the firing of any instance of `win`, e.g. with $c = IT$, is in conflict with any other instance of the same transition (any $c \in C \setminus \{IT\}$) since only one team can win the race. Composing the function on the arc from place `Finished` to transition `win` with the outcome of the SC relation, we obtain the colored tokens that the conflicting instances withdraw from place `Finished`; finally composing the transpose of the function on the arc from `pass` to `Finished` to the result of the last operation provides the instances of `pass` that may enable some instance of `win` in conflict with the instance we started with (e.g. the instance with $c = IT$). Summarizing $f := \overline{W^+(pass, Finished)^t} \circ \overline{W^-(win, Finished)} \circ < S{-}c >$ is a function from $cd(win)$ to $cd(pass)$ indicating the possible presence of stochastic confusion in markings where both `pass` and `win` are concurrently enabled.

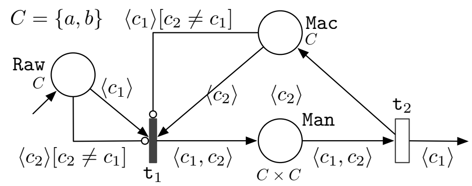

Fig. 3. A subnet from a FMS model.

```
f1 := @C <c>  f2 := @C <S-c>  f3 := @C,N^2 <c>[!n_1=n_2]
sf(f1.f2) => <S-c>
s(f3'.f1.f2) => [n_1 = !-1n_2]<S-c,S_N,S_N>
```

The interpretation of the result, a language expression with filter, is that given an instance of `win`, the instances of `pass` that may produce confusion, if enabled together with the former, are those involving a different team $(S - c)$ and such that n_1 is the predecessor of n_2. In other words, n_1 is the identifier of the athlete running the last section of the race (otherwise the arc function from `pass` and place `Finish` would represent an empty set, and the instance of `pass` would not enable any instance of `win`).

4.2 Machines scheduling policy in a Flexible Manufacturing System

Let us consider the model in Fig. 3, which is a small portion of a model representing a Flexible Manufacturing System (FMS) producing two types of parts. In [1] (Chapter 8) a GSPN representing such system is presented and studied. Here we concentrate only on the part of the model representing the scheduling policy for two machines that can process both part types. Place `Raw` represents a buffer of raw parts: the colors in class C allow to distinguish between the parts of type a and b. There are two machines $M2$ and $M3$ that can process both part types, however machine $M2$ processes parts of type a more efficiently than $M3$, on the other hand machine $M3$ processes parts of type b more efficiently than $M2$, for this reason the scheduling policy for parts waiting in place `Raw` tries to allocate as much as possible parts of type a on $M2$ and parts of type b on $M3$ (but without leaving a machine idle if there is at least one waiting part in place `Raw`). Place `Mac` represents the idle machines: colors a and b are used to identify also the machines, since there is a natural association of each machine with a part type for efficiency reasons. The scheduling policy is hidden in transition `t1`, since its instances correspond to the possible scheduling choices. Using the calculus it is possible to discover the structural relations existing among the possible instances of `t1`. Structural conflict among `t1` instances is computed through the following formula: $SfP(t1, Raw) \circ \overline{W^-(t1, Raw) - Id} \cup SfP(t1, Mac) \circ \overline{W^-(t1, Mac)}$. The two terms can be computed by SNexpression through the following commands:

```
load "FMS.sn"
SC(t1,t1,Raw) => <c_1,S-c_2>        SC(t1,t1,Mac) => <S-c_1,c_2>
sf(@C^2 <c_1,S-c_2> + <S-c_1,c_2>) => <c_1,S-c_2> + <S-c_1,c_2>
```

Hence the instances of `t1` potentially in conflict with a given instance of the same transition are those with variable `c_1` bound to the same value as the first instance, and variable c_2 bound to a different value, or viceversa, different value for variable `c_1` and same value for c_2. Since there are also inhibitor

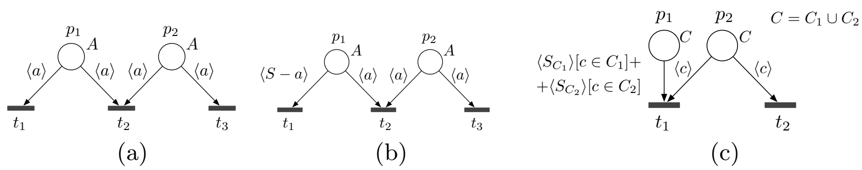

Fig. 4. ECS computation examples.

arcs connecting places `Raw` and `Mac` to transition `t1` we should check also the structural mutual exclusion relation. By applying the algorithm presented in Sec. 3 the following result can be computed (here we show the commands to obtain them through SNexpression):

```
SME(t1,t1,Raw) => <S-c_1,c_1>[c_1 = c_2] + <c_2,S_C>[c_1 != c_2]
SME(t1,t1,Mac) => <S-c_2,c_1>[c_1 != c_2] + <c_2,S-c_2>
sf( <S-c_1,c_1>[c_1 = c_2] + <c_2,S_C>[c_1 != c_2] +
 <S-c_2,c_1>[c_1 != c_2] + <c_2,S-c_2>) => <S-c_1,c_1>[c_1 = c_2] +
 <S-c_2,c_1>[c_1 != c_2] + <c_1,S-c_1>[c_1 = c_2] + <c_2,S_C>[c_1 != c_2]
```

From this result we can infer that the scheduling policy is deterministic, hence the weights associated with the instances of `t1` are irrelevant for the characterization of the stochastic process associated with the model. Let us consider the two cases: `c_1=c_2` and `c_1!=c_2`. In the former case any other instance with the two variables bound to different values (represented by the two terms `<S-c_1,c_1>[c_1=c_2] + <c_1,S-c_1>[c_1=c_2]`) are in mutual exclusion with the considered instance; the only instance which is not mutually exclusive is the one with `c_1=c_2` and both variables bound to a different value w.r.t. the reference instance, however we can verify that the two instances with `c_1` and `c_2` bound to the same value are not in conflict, so they are independent (if both enabled they can fire in any order without interfering with each other). Let us now consider the case `c_1!=c_2`, the `t1` reference instance is in mutual exclusion with all other instances, hence again there is no conflict to solve.

4.3 ECS computation

In this section the ECS computation technique is illustrated through three examples, depicted in Fig. 4. The first example in Fig. 4(a) is very simple: the SC and SSC relations computation gives the following results: $SSC(t_1,t_2) = SC(t_1,t_2) = \langle a\rangle$, $SSC(t_2,t_3) = SC(t_2,t_3) = \langle a\rangle$ (there is no structural conflict between t_1 and t_3, due to the net structure, nor among the instances of the same transition, due to the arc functions). Hence matrix M^0 and M^1 are as follows:

$$\mathbf{M}^0 = \begin{pmatrix} 0 & a & 0 \\ a & 0 & a \\ 0 & a & 0 \end{pmatrix} \qquad \mathbf{M}^1 = \mathbf{M}^2 = \begin{pmatrix} 0 & a & a \\ a & 0 & a \\ a & a & 0 \end{pmatrix}$$

The elements of matrix M^1 can be computed by applying the formulae presented in Sec. 3; so for example $M^1(t_3,t_1)$ is derived by computing the result of the following expression: $ID_a + ID_a \circ 0_A + ID_a \circ ID_a + 0_A \circ ID_a = ID_a$ where

$ID_a = @A\langle a\rangle$ and 0_A is the empty function (defined on domain A). In fact, only instances with same color $a \in A$ of t_1 and t_2 withdraw the same colored tokens from $P1$, and only instances with same color $a \in A$ of $t2$ and $t3$ withdraw the same colored tokens from $P2$. By transitive closure the instances with same color $a \in A$ of t_1 and t_3 are in relation SSC^* (through $t2$). Hence there is one ECS for each distinct color $a \in A$, including the instances of t_1, t_2 and t_3 with color a. This can be derived from any column of matrix M^1: in fact all functions in the column of transition t have a common domain, which is $cd(t)$. Given an element $a \in cd(t)$, the expression appearing in the row corresponding to transition t' provide the instances of t' that are in the same ECS as (t, a).

Now, let us consider the second example depicted in Fig. 4(b). In this case we can observe the ability of the tool to compute results that are parametric in the size of the classes. In particular, when computing the structural conflict relation between the instances of transition $t1$ we obtain the following result:

```
SC(t1,t1,p1) = <0_A> : |A| = 2          SC(t1,t2,P1) = <S-a_1> : 2<=|A|<=n
               <S-a> : 3 <= |A|<= n     SC(t2,t2,Pi) = <0_A> : 2<=|A|<=n
SC(t2,t3,P2) = <a> : 2 <= |A|<= n       SC(t3,t3,P2) = <0_A> : 2<=|A|<=n
```

The above SC relations are already symmetric (i.e. $SC(t_i, t_j) = SC(t_j, t_i)^t$), hence they lead directly to the elements of M^0. The transitive closure becomes stable after three steps leading to the following result:

$$\mathbf{M}^3 = \mathbf{M}^4 = \left\{ \begin{pmatrix} 0 & S-a & S-a \\ S-a & 0 & a \\ S-a & a & 0 \end{pmatrix} |A| = 2, \quad \begin{pmatrix} S-a & S & S \\ S & S-a & S \\ S & S & S-a \end{pmatrix} |A| \geq 3 \right.$$

In this case if $|A| \geq 3$ all instances of the three transitions end up in a unique ECS. If instead $|A| = 2$ there is one ECS for each element $a \in A$ including instance $(t1, a)$ and $(t2, A - a), (t3, A - a)$, in other words there are two ECSs, comprising the instances of $t2$ and $t3$ with same color, and the instance of $t1$ with the other color in A.

Finally let us consider the example in Fig. 4(c). In this case class C is partitioned into two static subclasses denoted C_1, C_2 (that for technical reasons and to simplify the discussion we assume have both cardinality > 2). The starting point is again the computation of the SC relation.

```
SC(t1,t1)= <S-c * S_C{1}>[c in C{1}] + <S-c * S_C{2}>[c in C{2}] =
         = <S_C{1}-c>[c in C{1}] + <S_C{2}-c>[c in C{2}]
SC(t2,t2) = <0_C>, SC(t1,t2,P2) = SC(t2,t1,P2) = <c_1>
```

Also in this case the relation is already symmetric (so that $SSC(t_i, t_j) = SC(t_i, t_j)$) and the above relations define the entries of matrix M^0. The final result is the following:

$$\mathbf{M}^2 = \mathbf{M}^3 = \begin{pmatrix} \langle S_{C_1} - c\rangle[c \in C_1] + \langle S_{C_2} - c\rangle[c \in C_2] & \langle S_{C_1}\rangle[c \in C_1] + \langle S_{C_2}\rangle[c \in C_2] \\ \langle S_{C_1}\rangle[c \in C_1] + \langle S_{C_2}\rangle[c \in C_2] & \langle S_{C_1} - c\rangle[c \in C_1] + \langle S_{C_2} - c\rangle[c \in C_2] \end{pmatrix}$$

That leads to the conclusion that there are two ECS in the model: the first one contains all the instances of t_1 and t_2 whose color belongs to static subclass C_1 while the second contains all the instances of t_1 and t_2 whose color belongs to static subclass C_2. Indeed if we consider a generic instance (t_1, c) of $t1$ the functions in the first column show us that if $c \in C_i$ all instances of t_1 with a color in the same static subclass (see the expression in $M^2(t_1, t_1)$) belong to the same ECS, and the same is true for all instances of $t2$ with color in the same static subclass (see the expression in $M^2(t_2, t_1)$). The same information can be derived from the second column (due to the symmetry of the involved relations).

5 Conclusion and future work

In this paper an approach for the computation of structural properties of SNs in a symbolic and parametric form has been proposed, extending previous works on the subject. The formulae for expressing the structural properties are based on a language $\mathcal{L}$, that extends the one used to specify SN arc functions, and on some operators. The language is closed w.r.t. such operators, and rules have been defined to transform an expression with operators into an expression of $\mathcal{L}$. This calculus has been implemented in the SNexpression tool; examples of application have been shown in the paper, highlighting also some directions for future work: extension of the composition operator to (some type of) multisets useful in several applications, e.g. for place and transition invariants verification; extension of the structural relations directly implemented in the tool, e.g. the symmetric and transitive closure of structural conflicts; user defined constraints on class cardinalities, and the possibility to import the SN models generated by other tools, e.g. GreatSPN.

References

1. M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.
2. M. Beccuti, C. Fornari, G. Franceschinis, S.M. Halawani, O. Ba-Rukab, AbRahman Ahmad, and G. Balbo. From Symmetric Nets to Differential Equations exploiting Model Symmetries. *The Computer Journal*, 58(1):23–39, 2015.
3. L. Capra, M. De Pierro, and G. Franceschinis. Deriving symbolic and parametric structural relations in symmetric nets: Focus on composition operator. Technical report, Computer Science Inst., DiSIT, Univ. del Piemonte Orientale. In preparation.
4. L. Capra, M. De Pierro, and G. Franceschinis. A High Level Language for Structural Relations in Well-Formed Nets. In *Proc. of the 26th Int. Conf. ATPN 2005*, volume LNCS 3536, pages 168–187. Springer, 2005.
5. 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.
6. 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.
7. 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/.
8. C. Dutheillet and S. Haddad. Conflict Sets in Colored Petri Nets. In *proc. of Petri Nets and Performance Models*, pages 76–85, 1993.
9. S. Evangelista, S. Haddad, and J.F. Pradat-Peyre. Syntactical Colored Petri Nets Reductions. In *ATVA'2005*, volume LNCS 3707, pages 202–216. Springer, 2005.

Appendix This appendix has been included for convenience of the referees, and will be removed from the final version of the paper: the information in the following sections can be found in the SNexpression tool web page.

A SNexpression

The `SNexpression` tool integrates a library that implements the core of the structural calculus and a command line interface (CLI) whose syntax is just summarized in the table below. Both components are written in `java` and have a highly modular structure. The library has been designed as a pure rewriting system: it allows arbitrary terms of the language used to express structural properties to be built and progressively reduced, until a given normal form in sub-language $\mathcal{L}$ is matched. Thanks to its particular blueprint the library can be easily extended to cover new operators and syntactical features. The CLI interfaces to the library via a quite sophisticated parser. It accepts two kinds of inputs: basic items of the structural language (domains, function-tuples, and operators defined on them), and higher level forms representing structural formulae. The computation of structural formulae requires that matrix-representations of SN models to be pre-loaded. Advanced functionalities such as definition of parametric color domains and naming of sub-expressions are provided. A simple notebook helps the user save and reuse intermediate results of the calculus.

Class definition	$set\ N\ ordered$
(static subclasses)	$set\ M :=\ \{1, [2, n]\}$
Function tuples on sets	$@N\ < S\_C, n >$
	$@C\hat{}2, D\hat{}2 < c\_1, c\_2, d\_1, d\_2 > [c\_2! = c\_1 + d\_2! = d\_1]$
Function tuples on multisets	$@A\hat{}2\ mset : 2 < a, S - a > +3 < a, a >$
Function symbols	$f := @N < S\_C, n >; g := @N\hat{}2 < n_2 >$
(can be used in expressions)	$f.g$
Simplification function	$s(A\hat{}2 < S - a\_1 {*}!a\_2 >))$
(applies the operators and gives as a result a language expression)	
Operators:	
intersection	Symbol: $*$; $@D\hat{}2 < S - d\_1 * S - d\_2, d\_1 > [d\_1! = d\_2]$
difference	Symbol: $-$;
transpose	Symbol: $'$; $@C\hat{}2, L(< c\_1, l > + < c\_2, l >)'$
composition	Symbol: .; $@A\hat{}2, M < a\_1, m\_1 > . < a\_1, a\_2, S\_M\{2\} >$
support	Symbol:$<<$ ... $>>$; $@A << 4 < a > [a \in A2] >>$
Output:	
Parametric results	$h := s(@A\hat{}2 < S - a\_1 {*}!a\_2 >)$
(result)	$<!a\_1 > [a\_1 = a\_2][\vert A\vert = 2]$
	$<!a\_2 > [a\_1! =!a\_2][3 <= \vert A\vert <= n]$
Access to elements	$h\{1\}$: $<!a\_1 > [a\_1 = a\_2]$
in parametric results	$h\{2\}$: $<!a\_2 > [a\_1! =!a\_2][3 <= \vert A\vert <= n]$

Table 2. A summary of the SNexpression comands syntax.