Symbolic structural techniques improving the analysis of Stochastic Symmetric Nets

Lorenzo Capra[1,*,†], Massimiliano De Pierro[2,*,†] and Giuliana Franceschinis[3,*,†]

[1]Università degli Studi di Milano, Italy
[2]Università di Torino, Italy
[3]Università del Piemonte Orientale, Alessandria, Italy

Abstract

Stochastic symmetric nets (SSN) represent a high-level Petri net formalism characterized by succinct annotations that encapsulate behavioral symmetries. These symmetries are utilized to enhance the efficiency of analysis, such as the construction of a symbolic reachability graph from which a lumped Markov chain is derived and the execution of symbolic discrete-event simulations. In the last two decades, powerful structural analysis techniques have also been developed for SSN. This article explores how these techniques can improve some performance analysis techniques, highlighting potential uses and recent progress.

Keywords

Stochastic Symmetric Nets, Structural techniques, Symbolic calculi, Performance analysis

1. Introduction

(Stochastic) Symmetric nets [1] are Colored Petri Nets with a specific syntax that allows us to exploit the models' symmetry in the analysis phase. SSN nodes (places and transitions) are associated with domains constituted by Cartesian products of finite basic color classes. A color class may be partitioned into static subclasses or, alternatively, circularly ordered. The colors within a subclass represent entities with similar behavior. SSN edges, connecting places and transitions, are annotated with functions that map transition domain instances to multisets within the place domain. These functions are composed of tuples of base color functions, including projections and constants that map to static subclasses or entire classes. Defining a symbolic initial marking and a symbolic firing mechanism, one can construct a Symbolic Reachability Graph (usually much smaller than the ordinary Reachability Graph) from which a lumped Markov chain is derived. Alternatively, symbolic discrete event simulations can be performed.

One distinctive feature of PNs is the possibility to derive some properties directly from their structure without (or before) building the state space; the efficient extension to Colored PNs, without resorting to unfolding, is not trivial. In the last 20 years, there has been an advance in the symbolic structural analysis of SSNs. Theoretical foundations are rooted in [2], which defines a language, denoted as $\mathcal{L}$, as a slight extension of the SSN arc functions. The elements of $\mathcal{L}$ are similar to the SSN arc functions but show expanded expressivity by using guards as prefixes (referred to as *filters*) and by allowing intersections of base color functions within tuples. The findings reported in [2] demonstrate the closure of $\mathcal{L}$ under a core set of functional operators (sum, difference, intersection, transpose, composition) used in structural analysis. Employing these operators enables the algebraic computation of fundamental structural relationships in SSNs, such as conflict, causality, and mutual exclusion, thereby obviating the need for unfolding. For example, the structural conflict between transitions $SC(t, t')$ symbolically represents the set of instances of t that can disable a generic instance of t'. In [3] further structural relations have been defined.

QualITA 2025: The Fourth Conference on System and Service Quality, June 25 and 27, 2025, Catania, Italy
*Corresponding author.
†These authors contributed equally.
lorenzo.capra@unimi.it (L. Capra); massimiliano.depierro@unito.it (M. De Pierro); giuliana.franceschinis@uniupo.it (G. Franceschinis)

The software tool SNexpression [4] (`http://www.di.unito.it/~depierro/SNexpression/`), composed of a command-line interface (CLI) built on an expandable Java library, supports the application of algebraic structural calculus to SSN models. SNexpression serves as a computer algebra tool that simplifies arbitrary user-defined structural expressions into comprehensible normal forms within $\mathcal{L}$. An entire SSN, encoded according to a designated format, can be imported into the CLI, facilitating the manipulation of its annotations. In the foreseeable future, the capability to verify symbolic (colored) invariants will be introduced.

In the next sections, we outline a few potential applications of SSN structural analysis related to performance evaluation: the novel results concern a complete formalization providing the basis for the efficient computation of the transitions enabled in a given marking of an SSN, particularly effective when the transition color domains are very large, and the definition of Symbolic Extended Conflict Sets, required in the assignment of priorities and weights to immediate transitions, taking into account all the mutual dependencies.

2. Symmetric Nets and Structural Relations

The SN formalism has been conceived to obtain a model representation whose intrinsic symmetries could be easily exploited. The Symbolic Structural Calculus goal is to derive symbolic representations of structural relations between net nodes: to this purpose, a language has been defined that resembles and extends the SN arc functions syntax. The SN formalism is presented first, followed by the definition of the language used to express the structural relations.

We assume that the reader is familiar with PN and HLPN, places, transitions, input, output, and inhibitor arcs, and marking. Let us succinctly introduce the color structure of an SN focusing on color classes, place and transition color domains, transition guards, and arc functions.

Definition 1 (Symmetric Net)**.** *A SN is a tuple:*

$$\mathcal{N} = (P, T, \Sigma, \mathcal{D}, var, \Phi, I, O, H, \pi, \mathbf{m}_0)$$

where P is the set of places, T is the set of transitions, Σ is the set of color classes, $\mathcal{D}$ assigns to each place and transition its color domain, I, O and H define the arcs connecting the net nodes, and the associated arc functions, $\pi : T \to \mathbb{N}$ is a function defining the priority of each transition; $\mathbf{m}_0$ is the initial marking, assigning to each place $p \in P$ a multiset of colors from its color domain $\mathcal{D}(p)$.

- Color classes in $\Sigma = \{C_i, i = 1 \ldots, n\}$ are finite and disjoint sets of elements called *colors*, each class C_i may be partitioned into static subclasses $\{C_{i,j}\}_j$ or a circular order relation can be defined on its elements;
- Place $p \in T$ color domain $\mathcal{D}(p)$ is defined as Cartesian products of color classes $\bigotimes C_i^{e_i}, e_i \geq 0, i = 1, \ldots, n$; $m(p)$, the marking of place p, is a multiset of elements from its color domain $\mathcal{D}(p)$;
- Transition $t \in T$ color domain $\mathcal{D}(t)$ defines the colored instances of t: a set of typed variables $var(t) = \{v_i^j : i = 1 \ldots n, j = 1 \ldots e_i\}$ is associated with each transition t, where the variable v_i^j type is the color class C_i in Σ, and a binding c assigning colors of appropriate type to t's variables defines a *transition instance* (that we denote (t, c));
- Transition t guard $\Phi(t)$ is a Boolean function expressed through a *standard predicate* (defined below) and evaluated on transition instances: the color domain $\mathcal{D}(t)$ of a transition includes all the instances of t satisfying the guard.
- A function labeling an arc connecting place p and transition t has domain $\mathcal{D}(t)$ and codomain $Bag[\mathcal{D}(p)]$, the set of the possible multisets on $\mathcal{D}(p)$. We use the notation ${}^\bullet t$ / ${}^\circ t$ as a shorthand notation for the set of places connected with transition t through an input/inhibitor arc.
  The general form of arc functions syntax is: $\sum_i \lambda_i . T_i[g_i],\ \lambda_i \in \mathbb{N}$, where T_i is a function-tuple $\langle f_1, \ldots, f_k \rangle$, i.e, a Cartesian product of *class functions* f_i, and g_i is a tuple-guard. Each class-function f is a linear function defined on a subset of variables of $var(t)$ of the same type. Let

$var_{C_i}(t)$ be the subset of $var(t)$ of type C_i, and $\widetilde{C_i}$ the set of static subclasses of C_i, then $f : \mathcal{D}(t) \to Bag[C_i]$ is defined as follows:

$$f = \sum_{v_k \in var_{C_i}(t)} \alpha_k.v_k + \sum_{v_k \in var_{C_i}(t)} \gamma_k.!v_k + \sum_{q \in \{1,\ldots,|\widetilde{C_i}|\}} \beta_q.All_{i,q} + \quad \eta.All_i$$

where $\alpha_k, \gamma_k, \beta_q, \eta \in \mathbb{Z}$. Symbol ! denotes the *successor* operator mapping the value of v_k to its successor (the type of v_k must be ordered). $All_{i,q}/All_i$ is a constant function mapping to the set $C_{i,q}/C_i$.
Boolean expressions g_i (guards) on $var(t)$ may be associated with transitions ($\Phi(t)$) or individual tuples; their terms are *standard predicates* checking whether two variables hold the same value ($v_{k1} = v_{k2}$), or if a variable "belongs" to a given static subclass ($v_k \in C_{i,j}$); if $\Phi(t)$ is false for a given instance of t, then that instance is not enabled, if a tuple-guard g_i is false for a given transition instance, the associated tuple evaluates to the empty-multiset. Class-functions must be such that no negative coefficients can result from their evaluation for any color satisfying the guard associated with the corresponding tuple/transition (if the guard is not explicit, then it is $true$).

A transition instance (t, c) is *enabled* in marking $\mathbf{m}$ if the following conditions are satisfied:

- $\Phi(t)(c) = \mathsf{true}$
- $\forall p \in {}^\bullet t, I(t,p)(c) \leq \mathbf{m}(p)$
- $\forall p \in t^\circ, H(t,p)(c) > \mathbf{m}(p)$
- there isn't any transition instance $(t', c') : \pi(t') > \pi(t)$ enabled in $\mathbf{m}$.

The *enabling degree* of an enabled transition instance (t, c) is defined as follows:

$$ed(t,c) = min_{p \in {}^\bullet t : I(t,p)(c) \neq 0, d \in I(t,p)(c)} \left\lfloor \frac{\mathbf{m}(p)(d)}{I(t,p)(c)(d)} \right\rfloor$$

The enabling degree of (t, c) is not defined[1] for (t, c) if $\forall p \in P, I(t,p)(c) = 0$.

The Stochastic Symmetric Nets (SSN) formalism comprises timed and immediate transitions: the former have priority level 0 and a random firing time (with negative exponential distribution) that determines the occurrence time of the transition firing and solves the possible conflicts through a race, the latter have priority level greater than 0, fire in zero time, and the conflicts between transitions at the same priority level are solved through probabilities obtained by normalizing the weights of transitions that are in a (direct or indirect) *dependency relation.* The appropriate specification of immediate transition weights and priorities in GSPNs (e.g., in the net obtained by unfolding an SSN) has been studied and discussed in [5, 6, 7] in order to allow a correct net level specification from which the underlying CTMC could be automatically derived; further reflections on this line have been proposed in [8], where the underlying stochastic model was instead a Markov Decision Process.

The language for structural relations The language for expressing structural relation has a syntax very similar to that used for SN arc functions, with some extension, and a (non limiting) constraint on the form of basic class functions. The resulting language is closed under a set of operators, namely intersection, transpose, sum, difference, composition, required to define the symbolic structural relations.

Definition 2 (Language $\mathcal{L}$). *Let $\Sigma = \{C_1, C_2, \ldots, C_n\}$ 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, ($\mathcal{D} = C_1^{e_1} \times C_2^{e_2} \times \ldots \times C_n^{e_n}, e_k \in \mathbb{N}$). Let $T_i : \mathcal{D} \to Bag(\mathcal{D}')$ and $[g_i']$ and $[g_i]$ be standard predicates in $\mathcal{D}'$ and $\mathcal{D}$, respectively. The set of expressions:*

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

[1] Hereafter we shall consider models where the enabling degree is never undefined, considering such situation as a modeling error which can be easily fixed to either prevent the enabling, or to force a fixed enabling degree in those cases.

Table 1
Computing structural properties in SN.

Structural Relation	Formula
$SC(t,t')$	$\bigcup_{p \in {}^\bullet t \cap {}^\bullet t'} \overline{I(t,p) - O(t,p)}^t \circ \overline{I(t',p)} + \bigcup_{p \in t^\bullet \cap {}^\circ t'} \overline{O(t,p) - I(t,p)}^t \circ \overline{H(t',p)}$
$SC(t)$	$SC(t,t) - \mathbf{1}$, where $\mathbf{1}$ is the identity on $\mathcal{D}(t)$
$SCC(t,t')$	$\bigcup_{p \in t^\bullet \cap {}^\bullet t'} \overline{O(t,p) - I(t,p)}^t \circ \overline{I(t',p)} + \bigcup_{p \in {}^\bullet t \cap {}^\circ t'} \overline{I(t,p) - O(t,p)}^t \circ \overline{H(t',p)}$
$SME_{simple}(t,t')$	$\bigcup_{p \in {}^\bullet t \cap {}^\circ t'} \overline{I(t,p)}^t \circ \overline{H(t',p)} + \bigcup_{p \in {}^\circ t \cap {}^\bullet t'} \overline{H(t,p)}^t \circ \overline{I(t',p)}$

is the language used to express SN structural relations, where T_i are function-tuples formed by class functions f_j, defined in turn as intersections of language elementary functions $\{c, !^k c, All_C, All_C - c, All_C - !^k c, \emptyset_C\}$ (projection, k^{th} successor, constant function corresponding to all elements of basic class C, projection/successor complement and the empty function; where C represents any class and c any variable of type C).

We can observe that, with respect to the arc function syntax, there may be a predicate left-composed with a tuple (that we call filter), the set of allowed elementary class functions is a bit different, with the *complement* $All - v$ among the elementary functions and the introduction of intersection of elementary functions. Any arc function may be expressed through an equivalent expression of this language.

Conventions In the following, we use capital letters $A, B, \ldots, Z$ to denote basic color classes in Σ (the set of *types*) and the corresponding lowercase letters (possibly with a subscript) to denote variables of the respective type: for example, if $\mathcal{D}(t) = M^2 \times N$, then $var(t) = \{m_1, m_2, n\}$. We use sans-serif lowercase letters to denote the elements (colors) of a class, for instance if $N = \{\mathsf{n}_1, \mathsf{n}_2, \ldots, \mathsf{n}_k\}$ and $M = \{\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_r\}$, then $(\mathsf{m}_3, \mathsf{m}_7, \mathsf{n}_4) \in M^2 \times N$.

Symbolic structural relations A structural relation between the SN nodes n, n' is symbolically expressed as a function $\mathcal{R}(n', n) : \mathcal{D}(n) \to 2^{\mathcal{D}(n')}$ and formalized using the language syntax $\mathcal{L}$. Its semantics is that, given an instance c of n, $\mathcal{R}(n', n)(c)$ results in instances c' of n' such that $c' \; \mathcal{R} \; c$ in the unfolding of the SN. We use the symbol $\mathcal{R}$ alone to denote a family of relations of the same type. We say that $\mathcal{R}$ is symmetric if and only if $\mathcal{R}(n', n) \equiv \mathcal{R}(n, n')^t$ for each pair n, n' on which $\mathcal{R}$ is defined. A relation may be made symmetric.

Table 3.1 summarizes a set of useful structural relations commonly used in structural analysis with the formulae for their calculation. The SC (structural conflict), SCC (structural causal connection) and SME (structural mutual exclusion) are defined on a pair $(t, t') \in T \times T$ have domain $\mathcal{D}(t')$ and codomain $2^{\mathcal{D}(t)}$.

An illustrative example of a SSN model and structural relation Consider the SN in Fig. 1. The transitions and places are $T = \{t_1, t_2, t_3\}$, $P = \{P_0, P_1\}$. There is only one color class C, that is, $\Sigma = \{C\}$. The color domains and variables of the three transitions are: $\mathcal{D}(t_1) = C, var(t_1) = \{c_1\}$, $\mathcal{D}(t_2) = \mathcal{D}(t_3) = C \times C$, $var(t_2) = \{c_1, c_2\}$, $var(t_3) = \{c_1, c_2\}$. The color domains of the places are: $\mathcal{D}(P_1) = C$, and $\mathcal{D}(P_0) = C$. Let us describe the function I for t_2: $I(t_2, P_1) = \langle c_2 \rangle [c_1 = c_2]$. $I(t_2, P_1) : C \times C \to 2^{Bag[C]}$: transition instances of t_2 with identical color components c_1 and c_2, say a, consume one token of the same color, a, from place P_1; the remaining instances do not consume tokens from P_1.

As an example of structural relation, consider the conflict of t_1 and t_2 on the tokens in the shared place P_0: $SC(t_1, t_2) : \mathcal{D}(t_2) \to 2^{\mathcal{D}(t_1)}$. According to Table 3.1: $SC(t_1, t_2) = \overline{I(t_1, P_0) - O(t_1, P_0)}^t \circ \overline{I(t_2, P_0)} = \langle c_1 \rangle^t \circ \langle c_1 \rangle [c_1 \neq c_2] = \langle c_1 \rangle [c_1 \neq c_2]$

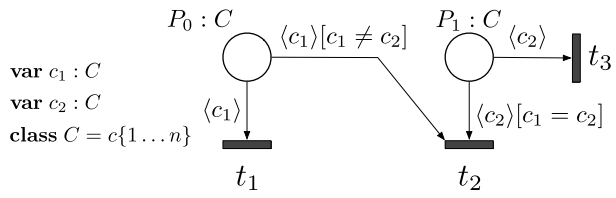


Figure 1: Example of conflicting colored transition in SN.

The meaning is that if we consider an instance $t_2 : \langle a, b \rangle$ of t_2 (where $\langle a, b \rangle \in C \times C, a \neq b$) then the conflicting instance of t_1 is $t_1 : \langle a \rangle$ obtained by applying SC to $\langle a, b \rangle$: $\langle c_1 \rangle [c_1 \neq c_2](a, b) = \langle a \rangle$; this can also be easily verified on the SN unfolding. If we consider $SC(t_2, t_1) : \mathcal{D}(t_1) \rightarrow 2^{\mathcal{D}(t_2)}$ instead, we obtain $(\langle c_1 \rangle [c_1 \neq c_2])^t \circ \langle c_1 \rangle = [c_1 \neq c_2] \langle c_1, All \rangle \circ \langle c_1 \rangle = \langle c_1, All - c_1 \rangle$ so that if we take an instance $t_1 : \langle a \rangle$ the set of conflicting t_2 instances is $\{t_2 : \langle a, b \rangle, b \in C \backslash a\}$. In the same net, we could also derive $SC(t_3, t_3)$, that is, conflicting instances of the same transition t_3 that result in $SC(t_3, t_3) = \langle All - c_1, c_2 \rangle$ hence, given an instance $t_3 : \langle a, b \rangle$, the set of conflicting instances of the same transition is $\{t_3 : \langle d, b \rangle, d \in C \backslash a\}$. Finally, $SC(t_3, t_2) = \langle S_C, c_1 \rangle [c_1 = c_2]$.

3. Efficient transition enabling test

Finding transition instances that can fire in a given marking is the core operation in both state-space-based approaches and discrete-event simulations. The effectiveness or lack thereof of this operation significantly influences the overall methodology within which it is utilized.

Empirical findings using the package GreatSPN [9] have indicated that this operation becomes onerous when the transition color domains are intricate or in the presence of free variable symbols, which appear exclusively in the transition output arcs.

Drawing on the concept presented in [10], computational efficiency can be significantly improved by formulating a concise representation that directly correlates with the potential set of instances enabled in a certain marking. Using an example, we illustrate here an extension of the method, which allows us to determine the exact set of enabled transition instances. A structural expression will offer a symbolic representation of the enabled set for each transition; when this expression is applied to a specific marking description, it yields a precise set of enabled transition instances.

Basic notation and definitions The fundamental operations used for the efficient transition enabling test are [2] the transpose of an arc function, the support, and the intersection of symbolic expressions. The support and intersection are straightforward functional extensions of the corresponding operations on multisets. The definition of the transpose operator is recalled in the following.

Definition 3. *The transpose of* $W(p, t) : \mathcal{D}(t) \rightarrow Bag[\mathcal{D}(p)]$ *is the function* $W'(t, p) : \mathcal{D}(p) \rightarrow Bag[\mathcal{D}(t)]$*, characterized by* $\forall c \in \mathcal{D}(p), c' \in \mathcal{D}(t)\ W'(t, p)(c)(c') = W'(p, t)(c')(c)$.

For example, when $W = I$, its transpose with respect to a given colored token within the domain of p yields the multiset of instances of t responsible for its withdrawal, where multiplicities correspond to the number of tokens withdrawn. As indicated previously, the transpose is an operator in language $\mathcal{L}$.

Multisets are extended to include the value ∞ in the set of scalars $\mathbb{N}$, thus if $Bag[A]$ is the set of multisets built over a set A then $Bag_\infty[A]$ is an extension in which multiset elements may have ∞ multiplicity, for instance multiset $3.\mathsf{a}_1 + 4.\mathsf{a}_3 + \infty.\mathsf{a}_2 \in Bag_\infty[A]$; Multiset operations are accordingly extended in a straightforward way according to the scalar Algebra rules.

If $\mathbf{m} \in Bag[A]$, with $\mathbf{m} = \sum_i s_i.\mathsf{a}_i$, and $r \in \mathbb{N}$ then the division of $\mathbf{m}$ by r is defined as follows:

$$\frac{\mathbf{m}}{r} = \sum_i \left\lfloor \frac{s_i}{r} \right\rfloor .\mathsf{a}_i$$

Definition 4 (Simple Function). *A function $F \in \mathcal{L}, F : D \to Bag[D']$ is said simple if:*

$$\forall d \in D, d' \in D' : \ f(d)(d') \leq 1$$

In other words, a simple function always results in multisets where elements are not repeated.

Definition 5 (Normal Form). *A function $F \in \mathcal{L}, F : D \to Bag[D']$, such that $F := \sum_i \lambda_i T_i$, is said to be in Normal Form if*

- *$\forall i \ T_i$ is simple, $\lambda_i \in \mathbb{N}^+$ and $\exists k_i \in \mathbb{N}^+ : \forall c \in D, \ T_i(c) \neq 0 \implies |T_i(d)| = k_i$*
- *$\forall i, j \ \ i \neq j \implies T_i \cap T_j = 0$*

Proposition 1. *Any function $F \in \mathcal{L}$ has a Normal Form. The Normal Form of F is not unique in $\mathcal{L}$.*

The proposition states that an arc function can be rewritten as a weighted sum of pairwise disjoint terms. Each term T_i thus results, when evaluated on the domain values, in multisets that retain identical cardinality k_i, and where all elements possess the same multiplicity of one, except for those arguments rendering the term null. We call k_i the size of the term T_i and use the notation $k_i = |T_i|$.

Figure 2 illustrates an SN transition t that encapsulates a sort of synchronization. Two color classes, N and M, are used to represent the processing nodes and the messages, respectively. The color domains of the places are explicitly defined, while $\mathcal{D}(t) = N \times M$ corresponds to the variables (projection functions) n and m present in the incident arcs. The enabling of t depends on the marking of the places p_1 (input) and p_2 (input and inhibitor).

The function $I(p_2, t)$ in Figure 2, left part, can be rewritten in normal form as shown in Figure 2, right part: The two terms (function tuples) have cardinality $|N| - 1$ and 1, respectively.

An intuitive lemma, which follows directly from Proposition 1, establishes the final form of input and inhibitor arc functions.

Corollary 1. *Let $I(p, t) \not\equiv 0$ and $H(p, t) \not\equiv 0$. These functions can be expressed in a Normal Form, $\sum_i \lambda_i I_i$ and $\sum_j \gamma_j H_j$, respectively, such that $I_i \equiv H_j \vee I_i \cap H_j = 0, \forall i, j$.*

We will assume that the input and inhibitor arc functions are written in a normal form and meet the Corollary 1. The function $H(p_2, t)$ in Figure 2, left part, is accordingly rewritten as shown in Figure 2, right part: the two different function tuples in $H(p_2, t)$ and $I(p_2, t)$ are disjoint. As discussed earlier, we assume that there cannot be transition instances that make $I(t, p) = 0$ for all input places of t: in other words, the enabling condition of all instances must involve at least one input place.

The subsequent corollary arises directly from the definition of a transpose.

Corollary 2. *Let $F := \sum_i \lambda_i T_i$ be a normal form. Its transpose $F^t = \sum_i \lambda_i T_i^t$ is in normal form.*

Consider Figure 2, where

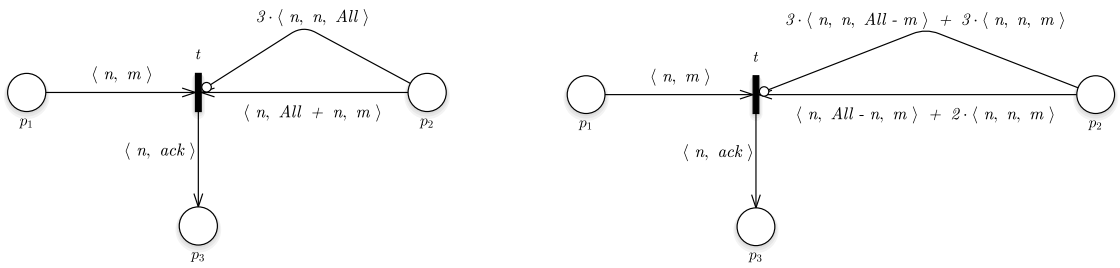
$$I(t, p_2) = 2.\langle n, n, m\rangle + \langle n, All - n, m\rangle$$
$$H(t, p_2) = 3.\langle n, n, m\rangle + 3.\langle n, n, All - m\rangle$$

Let $T_1 = \langle n, All - n, m\rangle$, $T_2 = \langle n, n, m\rangle$, and $T_3 = \langle n, n, All - m\rangle$, thus $I(t, p_2) = T_1 + 2 \cdot T_2$ and $H(t, p_2) = 3 \cdot T_2 + 3 \cdot T_3$. The sizes of the terms involved are: $|T_1| = N - 1$, $|T_2| = 1$, and $|T_3| = N - 1$. After transposing them, we obtain the following expression:

$$\begin{aligned} I^t(t, p_2) &= \overbrace{1.\langle n_1, m\rangle[n_1 \neq n_2]}^{T_1^t = \langle n, All-n, m\rangle^t} + \overbrace{2.\langle n_1, m\rangle[n_1 = n_2]}^{2.T_2^t = 2.\langle n, n, m\rangle^t} \\ H^t(t, p_2) &= \overbrace{3.\langle n_1, m\rangle[n_1 = n_2]}^{3.T_2^t = 3.\langle n.n.m\rangle^t} + \overbrace{3.\langle n_1, All - m\rangle[n_1 = n_2]}^{3.T_3^t = 3.\langle n, n, All-m\rangle^t} \end{aligned}$$

The normal form of the result allows us to write the *enabling conditions* locally to place p_2 and inferable from the transposes:

$$[1, \infty)\underbrace{\langle n_1, m\rangle[n_1 \neq n_2]}_{T_1^t}, \ [2, 3)\underbrace{\langle n_1, m\rangle[n_1 = n_2]}_{T_2^t}, \ [0, 3)\underbrace{\langle n_1, All - m\rangle[n_1 = n_2]}_{T_3^t}$$



$$\textbf{class}\, N = \{\mathsf{n}_1, \ldots, \mathsf{n}_4\}\ \ \textbf{class}\, M = \{\mathsf{m}_1, \ldots, \mathsf{m}_K\} \cup \{\mathsf{ack}\}\quad \textbf{vars}\ \ n : N\ m : M$$
$$\mathcal{D}(p_1) = \mathcal{D}(p_3) = N \times M\ \ \mathcal{D}(p_2) = N \times N \times M$$

Figure 2: Arc function rewriting for enabling test

The above expression conveys most of the information needed for the enabling test: for example, let $\mathsf{c} \in \mathbf{m}(p_2)$ and r its multiplicity, then $[2,3)T_2^t(\mathsf{c}) = T_2^t(\mathsf{c})$ iff $2 \leq r < 3$, otherwise $[2,3)T_2^t(\mathsf{c}) = 0$. $[2,3)T_2^t(\mathsf{c})$ constitutes a superset of the enabled instances of t when the color c is in the current marking of p_2. Obviously, for a given c, the terms $T_1^t(\mathsf{c})$, $T_2^t(\mathsf{c})$, and $T_3^t(\mathsf{c})$ are disjoint sets of instances.

Definition 6 (Enabled Super-Set Fragment)**.** *Assume $I(t,p)$ and $H(t,p)$ to be in normal form and term $m.T$ to be in $I(t,p)$ or term $n.T$ to be in $H(t,p)$, where $m, n \in \mathbb{N}$. Then:*

$$[\lambda, \gamma)T^t$$

is a function $Bag[\mathcal{D}(p)] \to Bag[\mathcal{D}(t)]$ so defined:

$$\lambda = m,\ \textit{and}\ \gamma = \infty\ \textit{iff}\ n = 0\ \textit{otheriwise}\ \gamma = n$$

and:

$$\forall \boldsymbol{d} \in \mathbf{m}(p),\quad [\lambda,\gamma)T^t(\boldsymbol{d}) = \begin{cases} T^t(\boldsymbol{d}) & \textit{iff}\ \lambda \leq \mathbf{m}(p)(\boldsymbol{d}) < \gamma \\ \emptyset & elsewhere \end{cases}$$

$$[\lambda,\gamma)T^t(\mathbf{m}(p)) = \sum_{\boldsymbol{d} \in \mathbf{m}(p)} [\lambda,\gamma)T^t(\boldsymbol{d})$$

Intersecting all the Enabled Superset Fragments applied to marking $\mathbf{m}(p_2)$ further refines the resulting superset of the enabled instances of t locally to p_2:

$$\bigcap_i [\lambda_i, \gamma_i)T_i^t(\mathbf{m}(p_2))$$

Definition 7 (Enabled Super-Set)**.** *Given the Enabled Superset Fragments $\left\{[\lambda_i,\gamma_i)T_i^t\right\}$ for pair (t,p), the function $En(t,p)$, which applies to a marking, is defined:*

$$En(t,p) = \bigcap_i [\lambda_i,\gamma_i)T_i^t$$

For the SN of the running example:

$$En(t,p_2) = [1,\infty)\underbrace{\langle n_1, m\rangle[n_1 \neq n_2]}_{T_1^t} \cap [2,3)\underbrace{\langle n_1, m\rangle[n_1 = n_2]}_{T_2^t} \cap [0,3)\underbrace{\langle n_1, All - m\rangle[n_1 = n_2]}_{T_3^t}$$

and similarly, but less complex:

$$En(t, p_1) = [1, \infty)\langle n, m\rangle$$

In Def. 6 looking for tokens that in current marking $\mathbf{m}$ satisfy the constraint $\lambda \leq \mathbf{m}(p) < \gamma$, when $\lambda = 0$, is not practicable because it will lead us to evaluate T^t also for tokens that are not in the marking (they have multiplicity 0 in $\mathbf{m}$). Thus, when $\lambda = 0$ for some T^t we will compute the superset of the enabled instances for that T^t via the complement of the disabled instances:

$$[0, n)]T^t = ([n, \infty)T^t)^{\complement}$$

When $\lambda \neq 0$, $En(t, p)$ is in general a proper superset of the enabled instances. We will use the information conveyed by the cardinality k_i of T_i to calculate the actual set of enabled instances, as well as their enabling degree: If, due to the application of term $[\lambda, \gamma)T^t$ to the current marking of p, (t, c) is in $En(t, p)(\mathbf{m}(p))$ then we know that this instance will require from p a sub-multiset[2] with $|T|$ distinct colors. We can compute how many of the tokens required by (t, c) are actually in p applying $[\lambda, \gamma)T^t(\mathbf{m}(p))$ and counting the scalar multiplicity of (t, c), we will call this value the Connection Degree of (t, c) with regard to $[\lambda, \gamma)T^t$, and in the current marking.

For example, consider a very simple net with a transition t and place p with $\mathcal{D}(p) = C = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}$ and $I(t, p) = \langle All - c\rangle$. Then, $I^t(t, p) = \langle All - c\rangle$, and the unique Enabled Super-Set Fragment is $[1, \infty)\langle All - c\rangle$, thus

$$En(t, p) = [1, \infty)\langle All - c\rangle$$

Assume first that $\mathbf{m}_1(p) = 2.\mathsf{a} + 2.\mathsf{b} + 2.\mathsf{c} + \mathsf{d}$. In this marking, it is straightforward to see that all instances of t are enabled, moreover the one of color $c = \mathsf{d}$ has the enabling degree 2, while all the others have the enabling degree 1. By Def. 6:

$$En(t, p)(\mathbf{m}(p)) = 3.\mathsf{a} + 3.\mathsf{b} + 3.\mathsf{c} + 3.\mathsf{d}$$

Thus, the obtained Enabled Super-Set matches the actually enabled instances. Observe that in the above expression, the scalar 3 points out that each instance will withdraw by marking $\mathbf{m}(p)$ 3 tokens with a distinct color (for example, instance (t, a) will withdraw a multiset $\mathsf{b} + \mathsf{c} + \mathsf{d}$). Now let us consider a different marking: $\mathbf{m}_2(p) = \mathsf{a} + \mathsf{b} + \mathsf{c}$; in this case only instance (t, d) is enabled, however

$$En(t, p)(\mathbf{m}(p)) = 2.\mathsf{a} + 2.\mathsf{b} + 2.\mathsf{c} + 3.\mathsf{d}$$

In this case, the Enabled Super-Set is a proper superset of the actually enabled instances because instances $(t, \mathsf{a}), (t, \mathsf{b})$, and (t, c) are not enabled. However, the multiplicity 2 indicates that these instances were derived from only two distinct colors in $\mathbf{m}(p)$, while three are actually required; instead, instance (t, d) is derived from $3 = |C| - 1 = |All - c|$ distinct colors in $\mathbf{m}(p)$. We shall call this multiplicity *Connection Degree* because, in a certain sense, it represents the number of connections from a given instance to distinct token colors in $\mathbf{m}(p)$. We will see later how to use this information to refine the Enabled Super-Set function, but first let us formalize the above concepts.

Definition 8 (Connection Degree)**.** *Given the pair (t, p), let $[\lambda, \gamma)T^t$ be an Enabled Superset Fragment of $En(t, p)$, and $\mathbf{m}(p)$ the marking of p. Let $\mathbf{b} \in Bag[\mathcal{D}(t)]$ such that $En(t, p)(\mathbf{m}(p)) = \mathbf{b}$. Let the value $s \in \mathbb{N}$ be the multiplicity of the color instance $\mathsf{c} \in \mathcal{D}(t)$ in multiset $\mathbf{b}$. Then, s is said Connection Degree of instance (t, c) in $\mathbf{m}$ locally to place p and Super-set $[\lambda, \gamma)T^t$.*

The Connection Degree s provides us with how many distinct tokens, of those needed by a given instance (t, c) belonging to $[\lambda, \gamma)T^t(\mathbf{m}(p))$, are actually in $\mathbf{m}(p)$. If $s = |T|$, then instance (t, c) is enabled locally to place p and Superset Fragment$[\lambda, \gamma)T^t$. When this holds, let r be the minimum multiplicity of the distinct tokens required by (t, c) actually in $\mathbf{m}(p)$:

$$r = min_{\mathsf{d} \in T(c)}\{\mathbf{m}(p)(\mathsf{d})\}$$

[2]This sub-multiset is part of the whole multiset $I(t, p)(\mathsf{c})$.

then $\lfloor r/\lambda \rfloor$ is the Enabling Degree of the instance (t, c), relative to Fragment $[\lambda, \gamma)T^t$. Referring to the marking $\mathbf{m}_1$ of the previous example, it holds $r = 1$ for all instances of t except (t, d) , due to the single token of color d in place p; instead, for example (t, d) it holds $r = 2$ because each needed token has multiplicity 2.

Again, if for all i the *Connection Degree* s_i is equal to $k_i = |T_i|$, then we can say that instance (t, c) is actually enabled in $\mathbf{m}(p)$. The actual enabling degree is $min_i\{\lfloor r_i/\lambda_i \rfloor\}$.

Definition 9 (Enabling Degree)**.** *Given $En(t, p)$, and a term T_i appearing in the normal form of $I(t, p)$, then $ed_i(t, p)$ for marking $\mathbf{m}$ is a function mapping to a multiset of colors of t so defined:*

$$ed_i(t,p)(\mathbf{m}) = \sum_{c \in En(t,p)(\mathbf{m}(p))} \frac{min_{d \in T_i(c)}\{\mathbf{m}(p)(d)\}}{\lambda_i}.c$$

and $ed(t, p)$ is a function so defined:

$$ed(t,p)(\mathbf{m}) = min_i\{ed_i(t,p,\mathbf{m})\}$$

Definition 10. *Given pair (t, p), and $En(t, p)$ the Super-set function of the enabled instances. Then, the actually enabled instances of t locally to p are given by the following function:*

$$EN(t,p) = \bigcap_{\forall i:\lambda_i \neq 0} \frac{[\lambda_i, \gamma_i)T_i^t}{k_i} \bigcap_{\forall i:\lambda_i = 0} \infty\big([\gamma_i, \infty)T_i^t\big)^{\complement}$$

where $k_i = |T_i|$.

We observe that in Def. 10 the division by k_i puts to 0 the multiplicities of the instances in $[\lambda_i, \gamma_i)T_i$ such that the Connection Degree is less than k_i. The instances remaining in $EN(t, p)$ will have multiplicity 1.

The actual enabled instances are finally computed by intersecting the locally enabled instances in all places.

Definition 11 (Enabled instances)**.** *Let $EN(t) \in Bag[\mathcal{D}(t)]$ represent the instances of t enabled in a marking $\mathbf{m}$. Then,*

$$EN(t,\mathbf{m}) = \bigcap_{p \in {}^\bullet t \cup t^\circ} EN(t,p)(\mathbf{m}(p))$$

and the Enabling Degree function is:

$$ed(t) = \bigcap_p ed(t,p)$$

For example, consider the following marking of the SN shown in Figure 2 (right part), where $N = \{\mathsf{n}_1, \ldots, \mathsf{n}_4\}$, $M = \{\mathsf{m}_1, \ldots, \mathsf{m}_K\} \cup \{\mathsf{ack}\}$, $K > 1$:

$$\begin{aligned}
\mathbf{m}(p_1) &= 2\langle \mathsf{n}_1, \mathsf{m}_1\rangle + \langle \mathsf{n}_2, \mathsf{m}_2\rangle \\
\mathbf{m}(p_2) &= 4\langle \mathsf{n}_1, \mathsf{n}_2, \mathsf{m}_1\rangle + 2\langle \mathsf{n}_1, \mathsf{n}_3, \mathsf{m}_1\rangle + 3\langle \mathsf{n}_1, \mathsf{n}_4, \mathsf{m}_1\rangle + 2\langle \mathsf{n}_1, \mathsf{n}_1, \mathsf{m}_1\rangle + 3\langle \mathsf{n}_2, \mathsf{n}_2, \mathsf{m}_1\rangle \\
&\quad + 2\langle \mathsf{n}_2, \mathsf{n}_1, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_2, \mathsf{n}_3, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_2, \mathsf{n}_4, \mathsf{m}_2\rangle + 3\langle \mathsf{n}_3, \mathsf{n}_1, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_3, \mathsf{n}_2, \mathsf{m}_2\rangle
\end{aligned}$$

Let us compute $EN(t, p_2)(\mathbf{m}(p_2))$, the enabling function locally to place p_2. Thus, we first compute the Enabled Super-set Fragments $[1, \infty)T_1^t(\mathbf{m}(p_2))$,$[2, 3)T_2^t(\mathbf{m}(p_2))$, and $[0, 3)T_3^t(\mathbf{m}(p_2))$ (Def. 6). Let us look at the term associated to T_3^t:

$$[0,3)\underbrace{\langle n_1, All - m\rangle[n_1 = n_2]}_{T_3^t}, \text{ with } n_1, n_2 \in N, m \in M$$

The transition instances belonging to this parametric set have no input from p_2 but only inhibition. Since $\lambda_3 = 0$ we will compute these instances using the complemented formula:

$$[3,\infty)T_3^t(\mathbf{m}(p_2))^{\mathtt{C}}$$

By applying the function $[3,\infty)T_3^t$ to marking $\mathbf{m}$, we obtain $\langle \mathsf{n}_2, All - \mathsf{m}_1\rangle$, the result is due to the only token $\langle \mathsf{n}_2, \mathsf{n}_2, \mathsf{m}_1\rangle \in \mathbf{m}(p_2)$ such that the first and second components are equal (guard $n_1 = n_2$) and whose multiplicity is at least 3 in $\mathbf{m}(p_2)$. This tuple identifies the instances of t inhibited in the current marking and therefore excluded from $EN(t,p_2)$. Complementing it provides the superset of the instances we are looking for:

$$[3,\infty)T_3^t(\mathbf{m}(p_2))^{\mathtt{C}} = \infty\langle All - \mathsf{n}_2, All\rangle + \infty\langle \mathsf{n}_2, \mathsf{m}_1\rangle$$

As second Fragment to evaluate in $\mathbf{m}(p_2)$, let us consider the term associated with T_1^t:

$$[1,\infty)\underbrace{\langle n_1, m\rangle[n_1 \neq n_2]}_{T_1^t} \text{ with } n_1 \in N, n_2 \in N, m \in M$$

This parametric set represents a group of instances of t consuming tokens from p_2 but not inhibited (upper range ∞). All of these instances need at least one instance of the consumed colors. The evaluation in the current marking results in (see Def. 10):

$$[1,\infty)\langle n_1, m\rangle[n_1 \neq n_2](\mathbf{m}(p_2)) = \langle \mathsf{n}_1, \mathsf{m}_1\rangle + \langle \mathsf{n}_2, \mathsf{m}_2\rangle$$

In detail follows the calculus that leads to this result. The representation below highlights in the underbrace the instances of t each color in $\mathbf{m}_2$ maps to, consistently with the multiplicity of the color in the marking:

$$\begin{aligned}\mathbf{m}(p_2) = \underbrace{\underbrace{4\langle \mathsf{n}_1, \mathsf{n}_2, \mathsf{m}_1\rangle}_{\substack{\text{since } 4\geq\lambda_1 \text{ then}\\ [1,\infty)T_1^t(\mathsf{n}_1,\mathsf{n}_2,\mathsf{m}_1)=\langle \mathsf{n}_1,\mathsf{m}_1\rangle}} + \underbrace{2\langle \mathsf{n}_1, \mathsf{n}_3, \mathsf{m}_1\rangle}_{\substack{\text{since } 2\geq\lambda_1 \text{ then}\\ [1,\infty)T_1^t(\mathsf{n}_1,\mathsf{n}_3,\mathsf{m}_1)=\langle \mathsf{n}_1,\mathsf{m}_1\rangle}} + \underbrace{3\langle \mathsf{n}_1, \mathsf{n}_4, \mathsf{m}_1\rangle}_{\substack{\text{since } 3\geq\lambda_1 \text{ then}\\ [1,\infty)T_1^t(\mathsf{n}_1,\mathsf{n}_4,\mathsf{m}_1)=\langle \mathsf{n}_1,\mathsf{m}_1\rangle}}}_{[1,\infty)T_1^t\left(\langle \mathsf{n}_1,\mathsf{n}_2,\mathsf{m}_1\rangle+\langle \mathsf{n}_1,\mathsf{n}_3,\mathsf{m}_1\rangle+\langle \mathsf{n}_1,\mathsf{n}_4,\mathsf{m}_1\rangle\right)=3\langle \mathsf{n}_1,\mathsf{m}_1\rangle} + \\ + \underbrace{2\langle \mathsf{n}_1, \mathsf{n}_1, \mathsf{m}_1\rangle + 3\langle \mathsf{n}_2, \mathsf{n}_2, \mathsf{m}_1\rangle}_{\substack{\text{since } n_1=n_2 \text{ then}\\ [1,\infty)T_1^t(\langle \mathsf{n}_1,\mathsf{n}_1,\mathsf{m}_1\rangle+\langle \mathsf{n}_2,\mathsf{n}_2,\mathsf{m}_1\rangle)=0}} + \\ + \underbrace{2\langle \mathsf{n}_2, \mathsf{n}_1, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_2, \mathsf{n}_3, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_2, \mathsf{n}_4, \mathsf{m}_2\rangle}_{3\langle \mathsf{n}_2,\mathsf{m}_2\rangle} + \underbrace{3\langle \mathsf{n}_3, \mathsf{n}_1, \mathsf{m}_2\rangle + 2\langle \mathsf{n}_3, \mathsf{n}_2, \mathsf{m}_2\rangle}_{2\langle \mathsf{n}_3,\mathsf{m}_2\rangle}\end{aligned}$$

There exist two instances, namely $\langle \mathsf{n}_1, \mathsf{m}_1\rangle$ and $\langle \mathsf{n}_2, \mathsf{m}_2\rangle$, whose Connection Degree is $|T_1| = 3$; both instances have enabling degree 2, since those are the minimal multiplicities common to the involved tokens present in $\mathbf{m}(p_2)$: respectively to the first three tokens in $\mathbf{m}(p_2)$ above, $ed_1(t,p_2)(\langle \mathsf{n}_1, \mathsf{m}_1\rangle) = min\{4,2,3\}$, and respectively to the tokens from 6th to 8th in $\mathbf{m}(p_2)$ above, $ed_1(t,p_2)(\langle \mathsf{n}_2, \mathsf{m}_2\rangle) = min\{2,2,2\}$. There are instead only 2 tokens in $\mathbf{m}(p_2)$ (the last two), resulting in instance $\langle \mathsf{n}_3, \mathsf{m}_2\rangle$, thus its Connection Degree is lower than $|T_1|$. Consequently, this term of $EN(t,p_2)$ leads to the result $\langle \mathsf{n}_1, \mathsf{m}_1\rangle + \langle \mathsf{n}_2, \mathsf{m}_2\rangle$.

The last Fragmentto evaluate is the one associated with T_2^t, namely

$$[2,3)\langle n_1, m\rangle[n_1 = n_2](\mathbf{m}(p_2))$$

Let us show below only the tokens in $\mathbf{m}(p_2)$ that do not map to the empty multiset:

$$\mathbf{m}(p_2) = \underbrace{2\langle \mathsf{n}_1, \mathsf{n}_1, \mathsf{m}_1\rangle}_{[2,3)T_2^t(\langle \mathsf{n}_1,\mathsf{n}_1,\mathsf{m}_1\rangle)=\langle \mathsf{n}_1,\mathsf{m}_1\rangle \text{ since } \lambda_2\leq 2<\gamma_2} + \underbrace{3\langle \mathsf{n}_2, \mathsf{n}_2, \mathsf{m}_1\rangle}_{[2,3)T_2^t(\langle \mathsf{n}_2,\mathsf{n}_2,\mathsf{m}_1\rangle)=0 \text{ since } 3\geq\gamma_2}$$

Table 2
Summary of the functions needed to obtain the set of enabled instances of t.

$[\lambda, \gamma)Ti^t$ where T_i is a term of the normal form of $I(t,p)$ and/or $H(t,p)$	Enabled Superset Chunk	Def 6: expresses the range of token multiplicity used to select a superset of potentially enabled instances of a given transition t based on the marking of a given input/inhibitor place p
$En(t,p)$	Enabled Superset of t instances, relative to input or inhibitor place p	Def. 7: intersection of Enabled Superset Chunks derived from the Input/Inhibitor arcs connecting p and t. When applied to a marking of p provides a superset of enabled instances of t relative to p.
$EN(t,p)$	Connection Degree and enabled instances of t relative to p.	Def. 8 and 10: The Connection Degree is the multiplicity of transition t instances in the multiset obtained from the application of $En(t,p)$ to a given marking of p. For the instances obtained from the chunk $[\lambda, \gamma)Ti^t$, the multiplicity is compared with $k_i = \lvert T_i \rvert$: if it is less than k_i the instance is discarded, otherwise it is kept.
$EN(t, \mathbf{m})$	Enabled instances	Def. 11: the set of enabled instances of t in marking $\mathbf{m}$: obtained combining the result of $EN(t,p)$ for all input places of t.
$ed(t,p)$ / $ed(t)$	Enabling Degree relative to input place p / to all input places of t.	Def. 9: A function of the marking that provides the enabling degree of the enabled instances of t with respect to the marking of p / to the marking of all input places of t.

Thus, $[2,3)T_2^t(\mathbf{m}(p_2)) = \langle \mathsf{n}_1, \mathsf{m}_1 \rangle$ and $ed_2(t,p_1)(\langle \mathsf{n}_1, \mathsf{m}_1 \rangle) = 2/\lambda_2 = 1$.
Intersecting the three Fragments (Def. 10) yields the final result:

$$EN(t,p_2)(\mathbf{m}(p_2)) = \langle \mathsf{n}_1, \mathsf{m}_1 \rangle$$

with $ed(t,p_2)(\langle \mathsf{n}_1, \mathsf{m}_1 \rangle) = min\{2, 1, \infty\} = 1$.

Similarly, it can be computed that $EN(t,p_1)(\mathbf{m}(p_2)) = 2\langle \mathsf{n}_1, \mathsf{m}_1 \rangle + 1\langle \mathsf{n}_2, \mathsf{m}_2 \rangle$ with the respective enabling degree equal to 2 and 1. In conclusion, $EN(t, \mathbf{m}) = \langle \mathsf{n}_1, \mathsf{m}_1 \rangle$ with enabling degree 1.

Throughout various stages of computation, we have implicitly applied certain operators of structural calculus, including composition, difference, and intersection. These operators enable us to manipulate symbolic representations efficiently.

3.1. Efficient state space generation

The approach just described may be integrated using a traditional method based on a proficient update of the set of enabled transition instances. Structural relations inherently encapsulate all potential dependencies that are locally induced by the firing of a transition instance. Such an instance has the capability to disable instances with which it is in (structural) conflict and to enable those to which it is (structurally) causally connected. Hence, exploiting the transpose of structural conflicts and causal connections (see Table) computed on the net structure it is possible to incrementally update the set of enabled instances after the state change due to enabled transition instances.

4. Extended Conflict Sets

The most recent version of SNexpression [11], grounded in the theoretical framework of [12], equips modelers with advanced capabilities to perform calculations specified by the user across the entire SSN, rather than limit calculations to pairs of nodes. A net relationship R between nodes within an SN is represented by a two-dimensional matrix, where each element $R[i,j]$ matches the associated relationship between the color instances of nodes i-th and j-th. The fundamental algebraic operations

Table 3
SD^*_{eq} relation for the Utility Control Room model.

SD^*_{eq}	assignSameZone	assignNearZone	assignFar	assignMunicipality
assign SameZone	$\langle S_C, z_1\rangle$	·	·	·
assign NearZone	·	$\langle c_1, z_1, z_2\rangle + \langle S_C, S_{Z_2}, z_2\rangle[g_1] + \langle S_C, S_{Z_3}, z_2\rangle[g_2]$	·	·
assign Far	·	·	$(\langle S_C, S_{Z_4}, S_{Z_1}\rangle + \langle S_C, S_{Z_1}, S_{Z_4}\rangle)[g_3] + \langle c_1, z_1, z_2\rangle$	$\langle S_C, S_{Z_4}, S_{Z_1}\rangle + \langle S_C, S_{Z_1}, S_{Z_4}\rangle$
assign Municipality	·	·	$\langle S_C, S_Z\rangle[g_3]$	$\langle S_C, S_Z\rangle$

$$g_1 = [z_1 \in Z_2, z_2 \in Z_3], g_2 = [z_1 \in Z_3, z_2 \in Z_2], g_3 = [z_1 \in Z_1, z_2 \in Z_4 + z_1 \in Z_4, z_2 \in Z_1], \quad S = All$$

available in previous versions (such as sum, difference, transpose, and composition) remain accessible at the matrix level, supplemented by some newly introduced operations typical of the matrix framework, including the transitive closure.

In conjunction with the established symbols representing matrices for net adjacency lists and fundamental relationships, namely Structural Conflict, Structural Causal Connection, and Structural Mutual Exclusion, a novel symbol is introduced to denote a more comprehensive and intricate form of dependency between transitions, termed Structural Dependency (SD). SD was first introduced for GSPN in [7] and extended to SSN in [12]: said briefly, two transition instances are *mutually dependent* if their firing order may influence the future behavior (different reachable states or different probability to reach the states). Detecting SD is even more crucial in SSN models that feature immediate transitions with *priorities*, because partitioning the set of transitions according to their priority can lead to having *indirect* forms of dependency.

An immediate application of the SD relation is in the calculation of the well-known ECS (Extended Conflict Sets) [13]. In fact, ECS, in their original definition, are *independent classes* of *dependent* immediate transition instances. Therefore, they can be calculated using SD as the basic dependence relation. In order to compute the equivalence classes of dependent transition instances, the SD relation is closed transitively SD^+, and made reflexive: SD^*=$SD^+ + \mathbf{1}$; and then restricted to pairs of transitions with the same priority level leading to SD^*_{eq}.

The algorithm for computing the ECS in SN, as equivalence classes of the SD^*_{eq} relationship, has been presented in [12] and is currently being developed in the SNexpression suite.

SCS enables the consistent assignment of probabilistic weights to immediate transition instances. Furthermore, it may be leveraged to construct the ordinary or symbolic reachability graph of an SSN with enhanced efficiency and efficacy by minimizing the interleaving of immediate transitions, since instances pertaining to two independent sets can be executed in arbitrary sequence.

Computing ECS in SSN: an application example We illustrate the derivation of the Symbolic ECS (SCS) by applying the method to an SSN inspired by a model of the Model Checking Contest[3]. This SSN models a Utility Control Room [14]. The model represents a system where the customers of a utility can call a customer support center in case of need (see Figure 3 in the appendix). A pool of technicians is available to provide the required service. Customers are spread over a certain number of zones. When a customer calls, the system looks for available technicians and finally it assigns one of them trying to minimize his travel time: it assigns with higher priority a technician who is already in the customer's zone and then, in decreasing priority order, a technician in a nearby zone or, finally, in a far zone. The color classes in this model are C, to identify the customers and Z to represent the zones; class Z is partitioned into four static subclasses used to distinguish zones that are near of far to each other. The

[3] https://mcc.lip6.fr/

analysis in this case is concentrated on four immediate transitions: `assignMunicipality` (priority 1) indicating where the customer is located, and `assignSameZone` (priority 3), `assignNearZone` (priority 2), and `assignFar` (priority 1).

SD^*_{eq} for this net is a block matrix M (see Tab. 3) composed of three blocks, M_1,M_2 and M_3, since there are three priority levels. Transitions in different priority groups are in different SCS. The derivation of SCS for the groups M_1 and M_2 is easier because they are both composed by a single colored transition. Let us illustrate the computation of the SCS for transition instances in the three groups.

SCS for transitions in M_1: matrix block M_1 is composed of transition `assignSameZone` only, instances of this transition can still be partitioned in several SCS. To compute them, consider the parametric instance $\texttt{assignSameZone}(c, z_1)$, where $\{c, z_1\} = Var(\texttt{assignSameZone})$ and start checking which other instances of the same transition are in the same class using the dependency relation in M_1. Thus, symbolically instantiating $SD^*_{eq}(\texttt{assignSameZone}, \texttt{assignSameZone}) = \langle All_C, z_1\rangle$ on such a color parameter $\langle All_C, z_1\rangle(c, z_1)$ allows obtaining the requested dependent instances, thus belonging to the same SCS: $\langle All_C, z_1\rangle(c, z_1) = \langle C, z_1\rangle$. SECS is updated as follows:

$$\text{SCS}(c, z_1) = \{\texttt{assignSameZone}(c, z_1), \texttt{assignSameZone}(C, z_1) : c \in C, z_1 \in Z\}$$

and after simplification (in fact SD^*_{eq} is reflexive then the first instance is contained in the second group of instances)

$$\text{SCS}(z_1) = \{\texttt{assignSameZone}(C, z_1), z_1 \in Z\}$$

This symbolic expression for the SCS covers all the instances of `assignSameZone`, that is all instances are assigned to one SCS. To check this condition, we can apply again the symbolic calculus:

$$\sum_{z_1 \in Z} \langle C, z_1\rangle \rightarrow \underbrace{\langle All_C, z_1\rangle \circ \langle All_Z\rangle = \langle All_C, All_Z\rangle}_{\substack{\text{equivalent calculus} \\ \text{performed in functional form}}} \rightarrow \langle C, Z\rangle$$

SCS for the transition `assignNearZone` in M_2: as for M_1, one can compute SCS instantiating the function in M_2 on the colors of the parametric instance $\texttt{assignNearZone}(c, z_1, z_2)$, which results in two SCSs due to mutually exclusive guards $[g_1]$ and $[g_2]$:

$$\begin{array}{ll} \text{SCS}(c, z_1, z_2) = & \{\texttt{assignNearZone}(\langle C, Z_2, z_2\rangle), c \in C, z_1 \in Z_2, z_2 \in Z_3\} \overset{\text{it simplifies}}{\longrightarrow} \\ \text{SCS}(z_2) = & \{\texttt{assignNearZone}(\langle C, Z_2, z_2\rangle), z_2 \in Z_3\} \end{array}$$

and

$$\begin{array}{ll} \text{SCS}(c, z_1, z_2) = & \{\texttt{assignNearZone}(\langle C, Z_3, z_2\rangle), c \in C, z_1 \in Z_3, z_2 \in Z_2\} \overset{\text{it simplifies}}{\longrightarrow} \\ \text{SCS}(z_2) = & \{\texttt{assignNearZone}(\langle C, Z_3, z_2\rangle), z_2 \in Z_2\} \end{array}$$

SCS for the transitions in M_3: M_3 is made up of transitions $\{\texttt{assignFar}, \texttt{assignMunicipality}\}$. The method first categorizes all the instances of `assignFar` iterating on the functions in the first column of M_3; we obtain the instances respectively of `assignFar` and `assignMunicipality` in some SCS with $\texttt{assignFar}(c, z_1, z_2)$. Due to the guard $[g_3]$ on the function in $M_3[1, 1]$, a first SCS is determined according to the parameter values satisfying g_3. Thus, instantiating $M_3[1, 1]$, and then $M_3[2, 1]$ the following SCS is determined:

$$\begin{aligned} \text{SCS}(c, z_1, z_2) = \{&\texttt{assignFar}(c, z_1, z_2), \texttt{assignFar}(\langle C, Z_4, Z_1\rangle + \langle C, Z_1, Z_4\rangle), \\ &\texttt{assignMunicipality}(\langle C, Z\rangle), c \in C, g\} \end{aligned}$$

and after some simplification:

$$\text{SCS} = \{\texttt{assignFar}(\langle C, Z_4, Z_1\rangle + \langle C, Z_1, Z_4\rangle), \texttt{assignMunicipality}(\langle C, Z\rangle)\}$$

Each remaining instance of $\texttt{assignFar}(c, z_1, z_2)$, whose color does not satisfy g_3, is in its own separate SCS. It remains to classify the instances of `assignMunicipality` not classified in the previous step, and in relation only with itself, because due to the symmetric nature of M_3 `assignFar` instances in some SCS with `assignMunicipality` were considered in the previous step. However, in this particular case all the instances of `assignMunicipality` have already been classified because the previous calculated $\texttt{assignMunicipality}(\langle C, Z\rangle)$ contain all.

5. Conclusions and future work

The symbolic structural relation calculus for (Stochastic) Symmetric Nets developed in the last two decades and implemented in the SNexpression tool is the basis for enabling several tasks that can be useful when analyzing an SSN model. In this paper, we have presented a novel method exploiting the operators of the structural calculus to derive the transition instances enabled in a given marking, applying functions derived from the structure directly to the marking, instead of enumerating all possible color instances and testing their enabling conditions: this is a useful feature which is common to state space construction and discrete event simulation of SSN. We have also illustrated the operators recently introduced in SNexpression and operating on matrices of functions derived from the model that, in particular, can express in a symbolic form the dependency relations between transitions, useful to support the definition of the probabilities to be used for conflict resolution between immediate transitions. Among the other features implemented in SNexpression and not discussed in this paper, we recall the automatic generation of a system of Symbolic Ordinary Differential Equations (SODE), which can provide approximate average results in systems that have very large state spaces. Using the symmetry of the model, a reduced number of equations is generated, usually much smaller than the total number of place instances. The adaptation of the method to the parametric size of color classes is ongoing. Another useful feature is the ability to verify that a given P-invariant holds, which is also required to verify the applicability of a SODE-based solution.

Future work will integrate additional features into the tool to improve it in several directions: (1) expanding the calculus to include general composition (of functions mapping on multisets), (2) improving the rules for the simplification of the resulting formulas, and (3) achieving a seamless integration with other tools, in particular with GreatSPN.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, Stochastic well-formed coloured nets for symmetric modelling applications, IEEE Tran. Comput. 42 (1993) 1343–1360.

[2] L. Capra, M. De Pierro, G. Franceschinis, A high level language for structural relations in stochastic well-formed nets, in: Applications and Theory of Petri Nets 2005, volume 3536, Springer-Verlag, 2005, pp. 168–187. doi:10.1007/11494744_11.

[3] L. Capra, M. De Pierro, G. Franceschinis, Computing structural properties of symmetric nets, in: Quantitative Evaluation of Systems, Proc. of the 12th Int. Conf., QEST 2015, volume 9259, Springer, 2015, pp. 125–140.

[4] L. Capra, M. De Pierro, G. Franceschinis, SNexpression: A symbolic calculator for symmetric net expressions, in: Proc. of the 41st Int. Conference Petri Nets 2020, volume 12152 of *LNCS*, Springer, 2020, pp. 381–391.

[5] G. Chiola, M. Ajmone Marsan, G. Balbo, G. Conte, Generalized stochastic petri nets: A definition at the net level and its implications, IEEE Trans. Software Eng. 19 (1993) 89–107. URL: https://doi.org/10.1109/32.214828.

[6] E. Teruel, G. Franceschinis, M. De Pierro, Clarifying the priority specification of gspn: Detached priorities, in: Proceedings of the 8th International Workshop on Petri Nets and Performance Models - PNPM'99, IEEE, 1999, pp. 114–123. doi:10.1109/PNPM.1999.796558, contributo in Atti di convegno.

[7] E. Teruel, G. Franceschinis, M. De Pierro, Well-defined generalized stochastic petri nets: A net-level method to specify priorities, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 29 (2003) 962–973. doi:10.1109/TSE.2003.1245298, articolo in rivista.

[8] J. Katoen, GSPNs revisited: Simple semantics and new analysis algorithms, in: 12th Int. Conf. on Application of Concurrency to System Design, ACSD 2012, Hamburg, Germany, June 27-29, 2012, IEEE Computer Society, 2012, pp. 6–11. URL: https://doi.org/10.1109/ACSD.2012.30. doi:10.1109/ACSD.2012.30.

[9] E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, G. Franceschinis, 30 years of GreatSPN, in: Principles of Performance and Reliability Modeling and Evaluation, Springer, 2016, pp. 227–254.

[10] L. Capra, M. D. Pierro, Efficient enabling test in simulation of SWN, in: The 2006 European Simulation and Modelling Conference - Modelling and Simulation 2006 - ESM'06, EUROSIS-ETI, 2006, pp. 367–374.

[11] L. Capra, M. De Pierro, G. Franceschinis, SNexpression: a new component for SN matrix-based structural analysis, in: Proc. of the 45th International Conference on Formal Techniques for Distributed Objects, Components, and Systems - FORTE 2025, Lille, France, 2025.

[12] L. Capra, M. De Pierro, G. Franceschinis, Symbolic dependency relations calculation in Symmetric Nets, Transactions on Petri Nets and Other Models of Concurrency (2025). Accepted for publication.

[13] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, Modelling with Generalized Stochastic Petri Nets, Wiley Series in Parallel Computing, John Wiley and Sons, 1995.

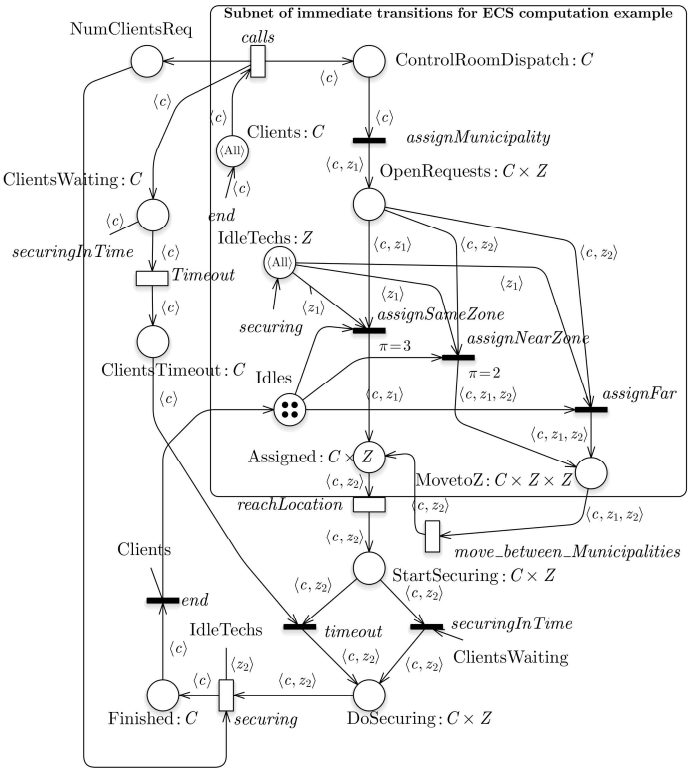
[14] E. G. Amparore, S. Donatelli, E. Landini, Modelling and evaluation of a control room application, in: W. van der Aalst, E. Best (Eds.), Application and Theory of Petri Nets and Concurrency, Springer International Publishing, Cham, 2017, pp. 243–263.

A. Online Resources

The SNexpression tool can be downloaded from http://www.di.unito.it/~depierro/SNexpression/. Two releases are currently available: the latest, integrating the calculation of structural properties on a whole SN using a matrix calculus, and the previous one, implementing the structural properties computation for specific pairs of nodes in the net, specified by the user. Several example models illustrated in the publications are also available online. A translator from the GreatSPN net description format (.PNPRO) (https://github.com/greatspn) to the SNexpression format (.sn) can be downloaded from the home page.

B. The utility control room SSN

The Utility Control Room model used in Section 4 is depicted in Figure 3: the SCS computation concerns the subnet of immediate transitions in the frame.



class $C = \{cli_1 \dots cli_k\}$, **class** $Z = \{zone_1\}$ **is** Z_1,$\{zone_2\}$ **is** Z_2, $\{zone_3\}$ **is** Z_3, $\{zone_4\}$ **is** Z_4
Guards:
$\Phi(\texttt{assignNearZone}) = [z_1 \in Z_2, z_2 \in Z_3 + z_1 \in Z_3, z_2 \in Z_2]$
$\Phi(\texttt{assignFar}) = [z_1 \in Z_1, z_2 \in Z_4 + z_1 \in Z_4, z_2 \in Z_1]$

Figure 3: The SSN model of the Utility Control Room