General composition for Symmetric Net arc functions with applications

L. Capra[1], M. De Pierro[2], and G. Franceschinis[3]

[1] Dip. di Informatica, Università di Milano, Italy
[2] Dip. di Informatica, Università di Torino, Italy
[3] DISIT, Università del Piemonte Orientale, Alessandria, Italy

Abstract. Structural analysis of High-Level Petri Nets is a powerful technique, but it is less supported than in PNs. A symbolic calculus for Symmetric Nets (SNs) has been developed and implemented, which allows one to check structural properties directly on SNs without unfolding: however it is limited to a particular form of composition, restricted to functions that map to sets. To complete the calculus for more general applications the ability to solve the composition of general SN arc expressions in a symbolic way is required. In literature, a few papers show how to solve this operation for a restricted category of SN. In this paper, we formalize the algebraic composition of general SN bag-functions. Some applications are also discussed.

1 Introduction

The possibility of checking some properties of PN models on their structure instead of (or before) generating their state space is a strong point in favour of the PN formalism. The extension of the structural analysis techniques to High Level Petri Nets (HLPNs), without resorting to unfolding, has been considered in the literature and some interesting results have been published [11,13], however the applicability of the proposed methods is often limited to particular classes of HLPNs and not completely supported by software tools.

A contribution in this direction has been proposed in [6,4] for models represented with the Symmetric Net (SN) formalism: it consists of a symbolic calculus operating on the arc expressions of the formalism, allowing to derive several interesting structural properties in a symbolic and parametric form; the software tool SNexpression [7] (`http://www.di.unito.it/~depierro/SNexpression/`) implements the results developed so far in this direction. One limitation of such calculus concerned the composition operator, which could be applied only to the support of arc expressions (hence to functions mapping to sets): although this is sufficient for the computation of several symbolic structural relations (e.g. symbolic structural conflict or causal connection [5]) it is not enough e.g. for checking some invariant properties based on the definition of P and T-semiflows, or for applying model reduction by agglomeration of transitions, two techniques

that have been applied to concurrent programs verification [11]. This paper fills the gap by defining the theory allowing to symbolically apply the composition operator to general SN arc functions.
Paper organization: Some definitions and notations are introduced in Sec. 1.1 and 1.2. Sec.s 2 and 3 introduce some useful definitions and properties, while in Sec. 4 we describe the steps to solve the general composition of SN functions. In Sec. 5 two example applications are illustrated. Sec. 6 concludes the paper and outlines directions for future work.

1.1 Generalized Bags: basic definition, notation and properties

A (generalized) bag b over a domain A is a map $b : A \to \mathbb{Z}$. $Bag[A]$ is the set of bags over A. Let $a \in A$ and $b \in Bag[A]$: we write $a \in b$ if and only if $b(a) \neq 0$. The set $\overline{b} = \{a, a \in b\}$ is the *support* of b. The bag with an empty support, is denoted $\emptyset_A$ or just $\emptyset$ if its domain is clear from the context. Bags $b_1, b_2 \in Bag[A]$ are *disjoint* if and only if $\overline{b_1} \cap \overline{b_2} = \emptyset$. A *proper* bag is a map $b : A \to \mathbb{N}$. The set of proper bags over A is denoted $Bag^+[A]$. The *size* of $b \in Bag^+[A]$, $|b|$, is $\sum_a b(a)$. Bag b is *type-set* if $\forall a \in b\ b(a) = 1$. A bag $b \neq \emptyset$ may be represented as a formal sum $\sum_{a \in A} b(a).a$. Let $b_1, b_2 \in Bag[A]$, $k \in \mathbb{Z}$. The scalar product $k \cdot b_1$ and the sum (diff) $b_1 \pm b_2$ are operations in $Bag[A]$ defined as $(k \cdot b_1)(a) = b_1(a) \cdot k$ and $b_1 \pm b_2(a) = b_1(a) \pm b_2(a)$, $\forall a \in A$. Operator $+$ is associative, $-$ is associative on $Bag[A]$, not on $Bag^+[A]$, $+$ is commutative. $\emptyset$ is the neutral element for $+,-$.

The Cartesian product is defined as follows. Let $b_1 \in Bag[A]$, $b_2 \in Bag[B]$: $b_1 \times b_2 \in Bag[A \times B]$ is $b_1 \times b_2(\langle a, b \rangle) = b_1(a) \cdot b_2(b)$, $\forall a \in A, b \in B$. The notation $\langle b_1, b_2, \ldots \rangle$ is used in place of $b_1 \times b_2 \times \ldots$. The Cartesian product is associative and may be distributed over inner $+, -$: $\langle \ldots, b_1\ op\ b_2, \ldots \rangle$, $op \in \{+, -\} = \langle \ldots, b_1, \ldots \rangle\ op\ \langle \ldots, b_2, \ldots \rangle$. Finally, $\langle \ldots, k \cdot b_1, \ldots \rangle = k \cdot \langle \ldots, b_1, \ldots \rangle$.

Generalized bags allow to distribute composition over sum or difference: in our context it simplifies the symbolic treatment of general composition.

1.2 A brief introduction to Symmetric Nets

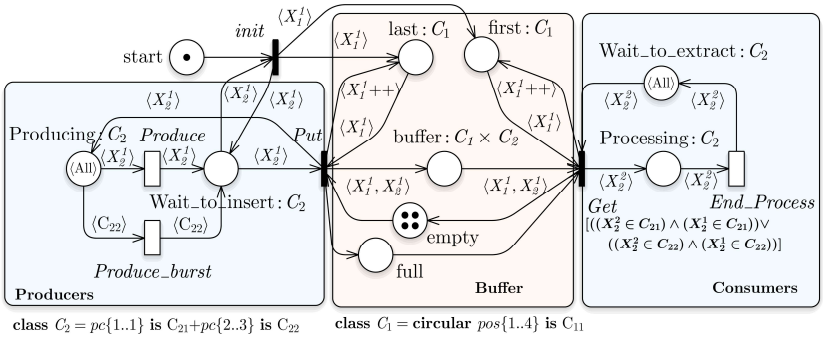


Fig. 1. A SN example: Producers and Consumers

The SN formalism belongs to the HLPN class: places and transitions are associated with a color domain $\mathcal{C}(.)$ expressing the possible *colors* of tokens and of the transition instances. Arcs are annotated by expressions, representing functions from $\mathcal{C}(t)$ to multisets $Bag^+[\mathcal{C}(p)]$; $W^-(p,t)$ and $W^+(p,t)$ denote t input and output arc expressions. The structure of a color annotation is based on the definition of color classes (finite not empty sets), which may be partitioned into (static) subclasses or be circularly ordered, and on a restricted set of basic functions: $S_i, S_{i,k}, X_i, !X_i$ namely diffusion/synchronization (on class C_i or subclass $C_{i,k}$), projection X_i^j and successor $!X_i^j$ (only for ordered classes). Color domains are defined as Cartesian products of n classes, where classes may be repeated. The arc expressions are weighted sums of tuples (Cartesian product) of basic functions (see Def. 4). The SN in Fig.1, represents a classical inter process communication example: producer and consumer processes communicating through a buffer (FIFO queue). There are two color classes, C_1 (circularly ordered, representing the position of messages in the buffer) and C_2 (id of producers/consumers), partitioned into two subclasses C_{21}, C_{22}. The places color domains are C_2, C_1, $C_1 \times C_2$ (tokens are tuples of one or two elements); some places and one transition in this net have *neutral* color (as in PNs, e.g. place empty and transition ProduceBurst). The transitions have color domain C_2, $C_1 \times C_2$, $C_1 \times C_2^2$, depending on the projection symbols appearing on their arcs; transition Get has a *guard*: a predicate restricting the allowed color instances. Arc expressions in Fig.1 are quite simple: $\langle X_2^1 \rangle$, $\langle !X_1^1 \rangle$, $\langle X_1^1, X_2^2 \rangle$, $\langle S_{2,2} \rangle$. A guard is a boolean expression whose terms can be either $(X_i^j = X_i^k)/(!X_i^j = X_i^k)$ or $X_i^j \in C_{i,k}$. An example of transition instance is $\mathsf{Get}(pos1, pc1, pc2)$: it satisfies the guard (the two colors $pc2, pc3 \in C_2$ belong to the same subclass C_{22}), and the projection $X_1^1(pos1, pc2, pc3) = pos1$, while $X_1^2(pos1, pc2, pc3) = pc2$, $X_2^2(pos1, pc2, pc3) = pc3$, finally $!X_1^1(pos1, pc2, pc3) = pos2$. A more formal definition of the SN arc expressions is deferred to Def. 4. Produce_burst has only one instance since the functions $\langle S_{22} \rangle$ on its arcs are constant, mapping on C_{22}.

An *incidence matrix* can be derived from the SN structure:

Definition 1 (Incidence Matrix). *The incidence matrix* $\mathbf{C}$ *of a HLPN model* N *is a* $P \times T$ *matrix of functions:* $\mathbf{C}[p,t] = W^+(p,t) - W^-(p,t)$. *If a transition instance* $t(c)$ *enabled in marking* $\mathbf{m}_i$ *fires, the corresponding state change can be defined in terms of the incidence matrix as:* $\mathbf{m}_j = \mathbf{m}_i + \mathbf{C}[.,t](c)$.

Among the structural analysis techniques that can be applied to SN models those based on the definition of P and T-semiflows allow to check interesting invariant properties of the model. The general composition proposed in this paper enables such possibility by allowing to compose the functions in $\mathbf{C}$ with T or P-indexed vectors of functions to check if they are semiflows, as discussed in Sec. 5.

2 Bag-expressions: general properties of composition

Operators $op \in \{+, -\}$ on functions mapping to Bags are defined as follows, in terms of Bag-operations. Let $f, h : A \to Bag[D]$, $op \in \{+, -\}$: $(f \ op \ h)(a) =$

$f(a)$ *op* $h(a)$, $(k \cdot f)(a) = k \cdot f(a)$, $\forall a \in A$. The constant function mapping to the null bag is denoted $\epsilon_{A,D}$ (or just ϵ).

Let $\{f_i : A \to Bag[D_i]\}$ be a family of functions: the *function-tuple* $\langle f_1, f_2, \ldots\rangle$ (or $\otimes_i f_i$), is a map $A \to Bag[D_1 \times D_2 \times \ldots]$: $\langle f_1, f_2, \ldots\rangle(a) = \langle f_1(a), f_2(a), \ldots\rangle$.

The properties of operations on bags, as well as the type-set notion, apply to bag-functions by considering all function arguments. We call any expression built of bag-functions a bag-expression.

Definition 2 (composition). *Let* $f : A \to Bag[D]$, $h : C \to Bag[A]$. *Then* $f \circ h : C \to Bag[D]$ *is* $f \circ h(c) = f^*(h(c))$, $\forall c \in C$, *where* $f^* : Bag[A] \to Bag[D]$ *is the linear extension of* f *defined as* $f^*(b) = \sum_{a \in A} b(a) \cdot f(a)$, $\forall b \in Bag[A]$.

We shall use the same symbol for a function and its linear extension. When a function takes a bag as an argument we implicitly refer to its linear extension.

The possibility to distribute composition over a difference or sum during the symbolic calculus, thanks to generalized bags, is very helpful because a complex composition may be reduced to an algebraic sum of simpler ones.

Bag-expressions may be prefixed by *filters* and suffixed by *guards*, both expressed as predicates. A $[true]$ guard/filter is usually omitted. Let $p : A \to \{true, false\}$. A filter or guard $[p]$ is a function $A \to Bag[A]$ such that $[p](a) = 1 * a$ if $p(a) = true$, $[p](a) = \emptyset$ otherwise. Let $f : A \to Bag[D]$, and p' be a predicate on D. The expressions $f[p]$ and $[p']f$ stand for $f \circ [p]$ and $[p'] \circ f$, respectively. The following definition characterizes an important class of functions.

Definition 3 (constant-size function). $f : A \to Bag^+[D]$ *is constant-size iff* $\exists n \in \mathbb{N}^+$ *such that* $\forall a \in A$ $f(a) \neq \emptyset \Rightarrow |f(a)| = n$.

Hereafter, with f constant-size we mean $f \equiv f'[p]$, with $f'[p] = \epsilon$ iff $p = false$. The following two general properties of composition concern constant and constant-size functions.

Property 1 (composition of a constant and a constant-size function). Let $f : B \to Bag[D]$ be a constant function so defined: $f(b) = d, \forall b \in B$. And let $h[g] : A \to Bag[B]$ be a n constant-size function, where $h[g](a) = \emptyset$ iff $g(a) = false$. Then, the composition $f \circ h[g]$ is defined as follows: $\forall a \in A$, $f \circ h[g] = n \cdot f'[g]$, where $f'[g] : A \to Bag[D]$ is such that $\forall a \in A, g(a) = true \Rightarrow f'(a) = d$.

Property 2 (composition of a tuple including a constant). Let f' be a constant function. Then $\langle f, f'\rangle \circ h = \langle f \circ h, f'\rangle$.

Since f and f' might be tuples in turn, Property 2 applies (up to a permutation of tuple positions) to any $T \circ h$, where $T := \langle f_1, \ldots, f_m\rangle$ contains some constant components and others non-constant. We may thus reduce such a composition to $T' \circ h$, where T' is built of all and only the non-constant components f_i of T. In the sequel, we focus on this case.

3 A language for composition of SN functions

Let us anticipate the main result presented in this paper by introducing the language used to calculate and express the composition of SN functions.

Definition 4 (Language $\mathcal{L}^c$). *Let*

- $D = C_1^{e_1} \times C_2^{e_2} \times ... \times C_n^{e_n}, e_* \in \mathbb{N}$, *be any color domain, i.e., a Cartesian product of colour classes ($e_i = 0$ means that C_i doesn't occur on D).*
- $\mathcal{B}_i^D = \left\{X_i^j, S_i, S_{i,k}, !^s X_i^j : D \to Bag[C_i]\right\}, i : 1, \ldots, n,\ j : 1, \ldots, e_i,\ k : 1, \ldots, ||C_i||,\ s \in \mathbb{Z}$; *be the set of elementary functions, where $!^s$ denotes the s-th $mod_{|C_i|}$ successor on C_i ($!^s X_i^j \equiv !^s \circ X_i^j$, $!^0 = id$)* [4].
- $T_j : D \to Bag[D']$, $T_j = \langle f_1, \ldots f_l \rangle$, *and* $\forall r : 1, \ldots, l$, $f_r : D \to Bag[C_i]$, $f_r = \sum_m \beta_m \cdot h_m$, $\beta_m \in \mathbb{Z}$, $h_m \in \mathcal{B}_i^D$; *f_r is said a class-function.*
- g'_j *and* g_j *be SN predicates on D' and D, respectively, such that all class functions appearing in $T_j[g_j]$ map to proper bags when g_j is true.*

$$\mathcal{L}^c = \left\{E : D \to Bag[D'], E = \sum_j \lambda_j [g'_j] T_j[g_j], \forall D, D'\right\},\ \lambda_j \in \mathbb{Z}$$

Any class-function f_r has as implicit guard: the guard of the tuple it belongs to.

Theorem 1. *$\mathcal{L}^c$ is closed under composition.*

The properties and lemmas presented in the rest of the paper justify the claim above[5]. Before that, let us point out a few interesting facts about $\mathcal{L}^c$.
$\mathcal{L}^c$ includes SN arc-functions, where scalars are such that they map to $Bag^+[D']$. The possibility of prefixing function-tuples with filters makes $\mathcal{L}^c$ slightly more expressive than the language of SN arc-functions, and has recently been included in the GreatSPN GUI. With respect to the language ($\mathcal{L}$) of SN structural relations defined in [4,5] and implemented in `SNexpression` [7] the main difference is that class-functions belong to SN legacy: $S_i - X_i^j$ is a difference between elementary functions (often treated as an idiom), not a new symbol; the intersection operator (though helpful) is not part of the language; scalars are in $\mathbb{Z}$; class-functions and, consequently, function-tuples are themselves bag-expressions, whereas both in $\mathcal{L}$ and the new `GreatSPN` GUI [2] they are set-expressions. As long as we restrict to expressions mapping to proper bags, however, $\mathcal{L}$ and $\mathcal{L}^c$ are equivalent. We use $\mathcal{L}^c$ for the sake of convenience/efficiency during the symbolic calculus.

The next important properties of $\mathcal{L}^c$ directly follow from the definition of $\mathcal{L}^c$, the Cartesian product and filter/guard properties, and the basic predicate reductions listed in [8], Appendix A.2; these properties ensure that a few assumptions, prerequisites for the application of the lemmas presented in next section, can always be met, possibly after the transformation of the involved expressions into equivalent ones satisfying such assumptions. The proof of all relevant properties and lemmas of this paper are reported in [8], Appendix A.1.

[4] symbols $!^s X_i^j$, with $s \neq 0$, and $S_{i,k}$ are mutually exclusive because we assume a color class is either partitioned or circularly ordered.

[5] showing the closure of $\mathcal{L}^c$ under all the other operations on bag-expression (sum, difference, intersection, transpose), though much simpler, is not the focus here.

Property 3 (conjunctive form equivalence). Any $E \in \mathcal{L}^c$ can be rewritten into $E' \in \mathcal{L}^c$, $E' \equiv E$, in which the predicates of filters/guards are conjunctive forms exclusively composed of (in)equality/membership clauses.

Property 4 (type-set equivalence). Any $E \in \mathcal{L}^c$ can be rewritten into $E' \in \mathcal{L}^c$, $E' \equiv E$, in which class-functions (therefore, function-tuples) are type-set.

Property 5 (constant-size function). Any class-function f_r is constant-size when considering its implicit guard. $|f_r|$ is the weighted algebraic sum of sizes of elementary terms of f_r. A function-tuple not prefixed by a filter is constant-size. Its size is the product of sizes of tuple's components.

For instance, $|(S_1 - X_1^1 - X_1^2)|$, with the implicit guard $X_1^1 \neq X_1^2$, is equal to $|C_1| - 2$. This expression is also type-set.

A function-tuple prefixed by a filter may not be constant-size. An example is $[X_1^1 = X_1^2]\langle S_1 - X_1^2, S_1 - X_1^1\rangle$: the size of this function-tuple depends on whether the 1st and 2nd element of a color-tuple argument are the same or not, it is $|C| - 1$ when applied to $\langle c, c\rangle$ and $|C| - 2$ when applied to $\langle c, c'\rangle, c \neq c'$. Observe that the projection symbols appearing in the filter refer to the elements of the tuple (X_1^j is the j-th element of class C_1 in the tuple), and should not be confused with the projection symbols appearing within the tuple or the guard.

Property 6 (constant-size equivalence). Any $E \in \mathcal{L}^c$ can be rewritten into $E' \in \mathcal{L}^c$, $E' \equiv E$, uniquely composed of constant-size terms.

Thus, if *needed*, we can transform any expression into an equivalent sum of constant-size (and/or type-set) terms. We will return to this in Section 4.1.

When calculating a Cartesian product, we have to take care of possible filters. Since we consider domains ordered by color, we only have to deal with two cases.

Property 7 ($\mathcal{L}^c$ tuple Cartesian product with filters adaptation).

Let $F_1, F_2 \in \mathcal{L}^c$, $F_1 : D \to D'$, $F_1 = [g_1]T_1$; $F_2 : D \to D''$, $F_2 = [g_2]T_2$, such that D', D'' are either a) disjoint or b) $D' = C_i^m$, $D'' = C_i^n$, $n, m \in \mathbb{N}^+$.

Then $\langle F_1, F_2\rangle = [g_1 \wedge {g_2}^*]\langle T_1, T_2\rangle$, where ${g_2}^* = g_2$ in case a), ${g_2}^*$ is obtained by replacing each X_i^j in g_2 with X_i^{j+m} in case b).

All results we are presenting hold modulo a permutation of tuple elements.

3.1 Conventions/notations used in the sequel

Lower-case letters f, g, h, p denote class-functions and predicates, if enclosed between angular and square brackets, respectively, otherwise represent any bag-expression (like in Section 2). Upper case F denotes any expression in $\mathcal{L}^c$ whereas T any (guarded) function-tuple in $\mathcal{L}^c$. $Var(f)$ denotes the set $\{X_j^i\}$ of variables (projections) appearing in f: if $f = [g]f'[g']$ then $Var(f) = Var(f') \cup Var(g')$.

Let $\overline{X}$ be a non-empty set of typed variables: $f(\overline{X})$ denotes a function such that $Var(f) = \overline{X}$. We may list all function variables, e.g., $f(X_i^j, X_h^w)$ meaning that $Var(f) = \{X_i^j, X_h^w\}$. The subset $\overline{X}_i \subseteq \overline{X}$ holds the class-C_i variables.

Index restriction ^{r} Let D: $C_1^{e_1} ... \times C_n^{e_n}$, $f = f(\overline{X}) : D \to Bag[D']$, and $e'_j = |\overline{X}_j|$ (by the way, $e'_j \leq e_j$). The index *restriction* of f is a function f^r: $C_1^{e'_1} ... \times C_n^{e'_n} \to Bag[D']$, obtained from f by replacing symbols in $X_j^i \in \overline{X}_j$, in superscript order, with $X_j^1, \ldots, X_j^{e'_j}$, for each j, $e'_j < e_j$.

For example, let $T = \langle X_1^1 + 2X_1^3, X_1^3, X_2^2 \rangle$: ${C_1}^3 \times {C_2}^2 \to {C_1}^2 \times C_2$, then $T^r = \langle X_1^1 + 2X_1^2, X_1^2, X_2^1 \rangle : {C_1}^2 \times C_2 \to {C_1}^2 \times C_2$.

Let $\overline{X}$ be a set $\{X_j^i\}$, $X_j^i : D \to C_j$.

Domain projection $\mathbf{\Pi}$ By convenience, we assume $\overline{X}$ ordered, first by class index then by superscript. We say the tuple $\mathbf{\Pi}_{\overline{X},D} : D \to D' = \otimes_{X_j^i \in \overline{X}} X_j^i$ projection of D (on D') induced by $\overline{X}$. D is omitted when clear from the context.

Projection image $_{\overline{X}}$ Let $T' = \langle h_1, \ldots, h_m \rangle$ with codomain D. The image of $\overline{X}$ on T' is the subtuple $T'_{\overline{X}} = \otimes_{i:\exists X_j^i \in \overline{X}} h_i$, with the same domain as T'. We denote by $T'_{\neg \overline{X}}$ the residual sub-tuple of T'.

4 Composing $\mathcal{L}^c$ expressions

Let $T : D \to Bag[D']$, $T' : D'' \to Bag[D]$, $D = C_1^{e_1} ... \times C_n^{e_n}$, $\overline{X} = Var(T)$ and $Var(T) \neq \emptyset$. Tuple T may have a guard, i.e., $T = \langle f_1, \ldots, f_n \rangle [g]$. Solving $T \circ T'$ means being able to rewrite this expression into $F \in \mathcal{L}^c$.

The algorithm to solve composition operates top-down. Each step is described in detail in this section. We proceed by making assumptions more and more stringent on T, T', until we get a base form.

Figure 2 outlines the main steps of the composition procedure on an example. At each step some rewriting rule is applied, possibly involving a sub-term (highlighted using under-braces): it is described in the legend along with the reference(s) to the corresponding lemma(s).

We hereafter assume to have preliminarily operated all generic reductions described in Sec.2 (see e.g. the first step in the example of Fig. 2), thus we let T be $\langle f_1, \ldots, f_n \rangle [g]$, where $\forall i \; Var(f_i) \neq \emptyset$, and $T' = \langle h_1, \ldots, h_m \rangle$.

Property 8. $T = T^r \circ \mathbf{\Pi}_{\overline{X}}$.

A first basic result says that we can solve $T \circ T'$ by considering T^r and the image of $Var(T)$ on T'.

Lemma 1. *Let $T = T(\overline{X})$ and $T'_{\overline{X}} \neq T'$. If $T'_{\neg \overline{X}}$ is of constant-size k, then:*

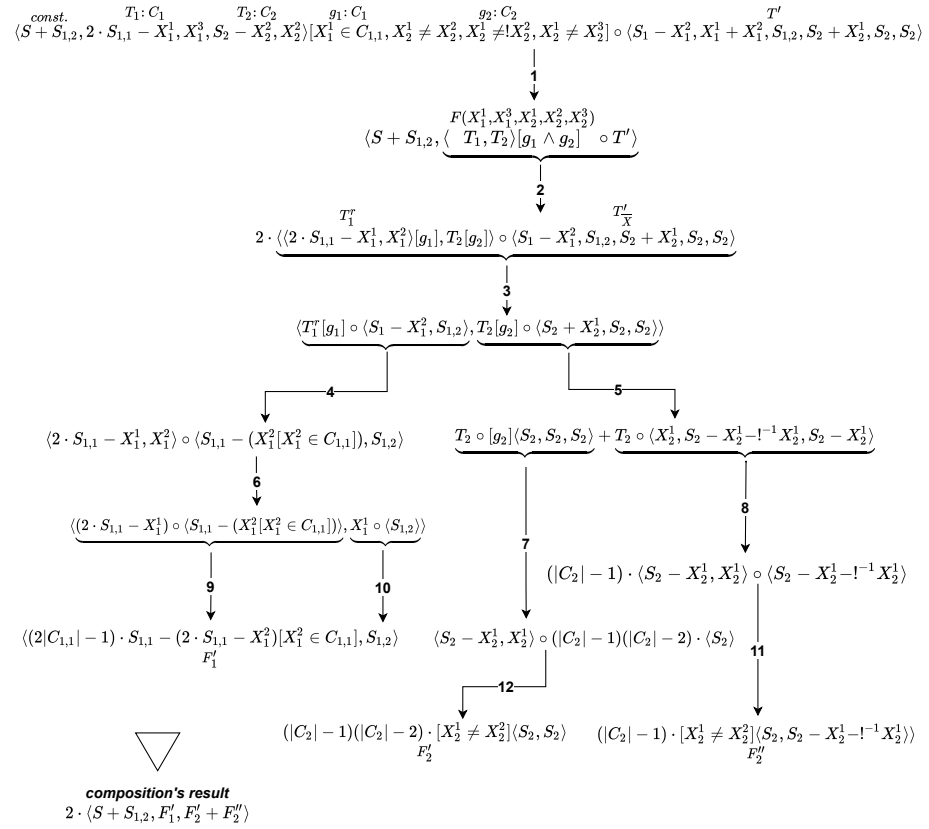
$$T \circ T'[g'] = k \cdot T^r \circ T'_{\overline{X}}[g']$$

Lemma 1 is applied twice in Fig. 2; follows a third example ($|2S_1 - X_1^2| = 2|C_1| - 1$):

$$\langle X_1^1, X_1^1, S_1 - X_1^3 \rangle \circ \langle S_1, 2S_1 - X_1^1, S_{1,1} + X_1^2 \rangle = (2|C_1| - 1) \cdot \langle X_1^1, X_1^1, S_1 - X_1^2 \rangle \circ \langle S_1, S_{1,1} + X_1^2 \rangle$$

Based on Lemma 1 (and Property 6), we focus on tuple compositions where the image of left tuple's variables coincides with the entire right tuple.

A second basic result allows one to distribute a composition over the independent parts of the left operand (up to a permutation of tuple elements).



rule	short description	reference(s)
1	constant pulled-out of the left tuple	Property 2
2,8	left tuple restriction and projection on the right tuple	Lemma 1
3,6	distribution of composition on independent left sub-tuples	Lemma 2
4	reduction (move) of a membership clause of g	see [8], Appendix A.2
5	distribution property, reduction to a simple tuple-prefix	Claim 1, Def. 5
7	elimination of an in-between guard built of inequalities	Corollary 2, Lemma 5
9	distribution property and composition of a left constant	Property 1
10	elementary composition	Sect. 4.1 (table)
11, 12	basic composition (variable repetitions on the left tuple)	Property 14

Fig. 2. A complete composition example (C_1 is a partitioned class, C_2 is ordered).

Lemma 2.
Let $T = \langle F_1(\overline{X}_1), \ldots, F_h(\overline{X}_h)\rangle \wedge \forall i,j,\ i \neq j, \overline{X}_i \cap \overline{X}_j = \emptyset,\ \wedge T'_{\overline{X}} = T'$.

$$T \circ T' = \langle {F_1}^r \circ T'_{\overline{X}_1}, \ldots, {F_h}^r \circ T'_{\overline{X}_h}\rangle$$

In other words, one can separately compose independent sub-tuples of T with their images on T', whatever form they have. As a consequence, due to color independence, one can partition T (and T') in sub-tuples based on color-classes.

We therefore focus on function-tuples $C_i^e \rightarrow C_i^{e'}$ and hereafter omit class index in basic functions: X^j replaces X_i^j, S_j replaces $S_{i,j}$, S replaces S_i. We assume that monochromatic tuples do not include independent sub-tuples.

Lemma 2 exploits a nice property of linear extension of Cartesian product of bag-functions by which, if an argument is in turn a product of bags, it is possible to evaluate the product-function in a modular way.

In Fig. 2 this Lemma is applied twice; follows another non-trivial example:

$$\langle X^1, X^3, S - X^2, X^2\rangle[X^1 \neq X^3 \wedge X^2 \neq X^4] \circ \langle S, 2S - X_1^1, S, S_1 + 2X_2\rangle =$$
$$\langle\langle X^1, X^3\rangle[X^1 \neq X^3], \langle S - X^2, X^2\rangle[X^2 \neq X^4]\rangle \circ \langle S, 2S - X^1, S, S_1 + 2X_2\rangle \overset{Lem.2}{=}$$
$$\langle\langle X^1, X^2\rangle[X^1 \neq X^2] \circ \langle S, S\rangle, \langle S - X^1, X^1\rangle[X^1 \neq X^2] \circ \langle 2S - X^1, S_1 + 2X_2\rangle\rangle$$

The two compositions that we ended up with are representatives of the base cases of tuple composition we have to solve.

4.1 Tuple composition's base cases

We can identify a few base cases of function-tuple composition as a result of Lemma 2 application. One in which the left-hand tuple is a one-variable function and the right tuple is a singleton. The others in which there is an infix predicate (guard/filter) that cannot be eliminated. The presence of an infix predicate complicates the solution of a composition. Based on properties of filters and filter reduction rules we may reduce such expressions to a particular form. The following statements assume that the right operand of a composition (T') is possibly followed by a guard.

Definition 5 (tuple prefix/composition simple form(s)).
Let $T' = \langle h_1, \ldots, h_m\rangle$. *A tuple-prefix* $[g]T'$ *is simple if and only if*

1. *g is a conjunction of (in)equalities, such that for each clause* $(X^i \overset{\neq}{=}!^s X^j)$ *in g:* h_i *is type-set,* $|h_i| > 1$, *and* $h_i =!^s h_j$
2. *let* $g_= \cup g_{\neq}$ *be the partition of g in equalities and inequalities*
   (a) *if* $g_{\neq} \neq \emptyset$, *each partition* $g_1 \cup g_2$ *of* $g_{\neq}$ *is such that* $Var(g_1) \cap Var(g_2) \neq \emptyset$
   (b) *if* $g_= \neq \emptyset$, *then* $g_=$ *may be partitioned in* $g_=^i, \ldots, g_=^w$, $w \in \mathbb{N}^+$, *such that*
   $\forall i,j,\ i \neq j,\ Var(g_=^i) \cap Var(g_=^j) = \emptyset$
   $\forall g_=^i \quad Var(g_=^i) \cap Var(g_{\neq}) = \{X^j\}$

$T[g] \circ T'$ *is a simple form of composition if and only if* $[g] \circ T'$ *is a simple tuple-prefix and* $\forall i\ |Var(g_=^i) \cap Var(T)| \leq 1$.

Claim 1 *We can rewrite any composition $T[g] \circ T'$ that cannot be decomposed according to Lemma 2 in terms matching Definition 5 (see [8], Appendix A.2)*

In other words, if required we may assume that infix predicates are made of (in)equalities between projections which point to equal (modulo-successor) type-set class-functions of T' of size > 1. Equalities define equivalence classes of projections, one representative of each class must appear in inequalities, and at most one occur on T. In Fig. 2 steps 4 and 5 lead to a form satisfying Def.5; follows another example: Let ($|C_2| > 1$):

$$\langle S_2 - X^3, X^1\rangle[X^1 \neq X^2 \wedge X^1 \in C_2 \wedge X^1 = X^3] \circ \langle S, S, S + X^1\rangle \rightarrow$$
$$\langle S_2 - X^1, X^1\rangle[X^1 \neq X^2 \wedge X^1 = X^3] \circ \langle S_2, S_2, S_2\rangle +$$
$$\langle S_2 - X^1, X^1\rangle[X^1 = X^3] \circ \langle S_2, S - S_2, S_2\rangle + \langle S_2 - X^1, X^1\rangle \circ \langle X^1, S - X^1, X^1\rangle[X^1 \in C_2]$$

Tuples prefixed by a filter matching Definition 5 have an important property.

Property 9. A simple tuple-prefix $[p]T : D \rightarrow Bag[D']$ is constant-size.

Four base cases have to be considered, they are listed below.

1. $T(X^1) \circ \langle h\rangle$
2. $T(\overline{X})[g] \circ T'$, with $|\overline{X}| > 1$ and $\overline{X} \supseteq Var(g)$
3. $T(\overline{X})[g] \circ T'$, with $\overline{X} \subset Var(g)$
4. $T[g] \circ T'$, with $Var(T) \cap Var(g) = \emptyset$

Case 1: $T(X^1) \circ \langle h\rangle$ In absence of infix predicates, we can always reduce to one such form due to the distribution property of sum/diff. For example:

$$\langle S - X^1 + 2X^2, X^2\rangle \circ \langle h_1, h_2\rangle = \langle (S - X^1) \circ \langle h_1\rangle, X^1 \circ \langle h_2\rangle\rangle + 2|h_1| \cdot \langle X^1, X^1\rangle \circ \langle h_2\rangle$$

Case 1.a: $|h| = 1$; this is the simplest situation.

Property 10. Let $|h| = 1$. Then $F(X^1) \circ \langle h\rangle$ is obtained by replacing each occurrence of symbol X^1 in F with h.

It directly follows from the definition of composition.

For example, $\langle S + X^1, X^1\rangle \circ \langle h_2\rangle$, $|h_2| = 1 \longrightarrow \langle S + h_2, h_2\rangle$.

Case 1.b: $|h| > 1$. The basic compositions are summarized in the following table (omitting tuple notation),the first one is applied in Fig. 2, step 10:

Basic composition rules and some useful identities			
$X^1 \circ h = h$	$!^s X^1 \circ h = !^s h$	$S - X^1 \circ h = \lvert h\rvert \cdot S - h$	$S - !^r X^1 \circ h = \lvert h\rvert \cdot S - !^r h$
$!^r S = S$	$!^s!^r X^i = !^{s+r} X^i$	$!^r(h_1 \pm h_2) = !^r h_1 \pm !^r h_2$	

If $T = \langle f_1(X^1)\rangle$ the above basic rules (and property 2) are enough. For example, letting $|C_1| = 3$:

$\langle 2S - X^1 - !X^1\rangle \circ \langle S - X^2\rangle \rightarrow \langle 4S - S + X^2 - S + !X^2\rangle \equiv \langle 2S + X^2 + !X^2\rangle$.

Repetition of a projection in T Let $T = \langle f_1(X^1), \ldots, f_m(X^1)\rangle$, where (without loss of generality due to Property 4) $f_i(X^1) = !^{s_i}X^1$, $\forall i$ (symbol $!^sX^i$ from now on denotes a projection *possibly* prefixed by the s-th successor). We consider first an unordered color class, then we generalize.

Property 11. Let h be type-set. $\underbrace{\langle X^1, \ldots, X^1\rangle}_{m>1} \circ \langle h\rangle = [\bigwedge_{i:2\ldots m} X^1 = X^i]\,\underbrace{\langle h, \ldots, h\rangle}_{m}$

Here are some non-elementary, type-set class-functions[6]:

$S - X^1$; $\quad S - X^1 - X^2[X^1 \neq X^2]$; $\quad S_i - X^1[X^1 \in C_i]$; $\quad S - X^1 - !X^1$.

We can extend the above outcome to any color class.

Property 12. Let h be type-set.

$$\langle !^{r_1}X^1, \ldots, !^{r_m}X^1\rangle \circ \langle h\rangle = [\bigwedge_{i:2\ldots m} X^1 = !^{(r_1 - r_i)}X^i]\langle !^{r_1}h, \ldots, !^{r_m}h\rangle$$

As an example:

$\langle !X^1, X^1, !X^1\rangle \circ \langle S - X^2\rangle = [X^1 = !X^2 \wedge X^1 = X^3]\langle S - !X^2, S - X^2, S - !X^2\rangle$.

In some cases *may* treat the idiom $S - X^1$ as a single function for convenience. Here are two situations that are more efficiently processed applying the following properties, although they could be solved applying the previous rules. Without loss of generality, in both cases $S - X^1$ is the last element of T.

Property 13. Let h be a type-set function.

$\underbrace{\langle X^1, \ldots, X^1, S - X^1\rangle}_{m} \circ \langle h\rangle = [X^1 \neq X^m \bigwedge_{i:2\ldots m-1} X^1 = X^i]\,\underbrace{\langle h, \ldots, h, S\rangle}_{m}$

If there are no repetitions of X^1 then h may be any function.

Property 14. $\langle X^1, S - X^1\rangle \circ \langle h\rangle = [X^1 \neq X^2]\langle h, S\rangle$

Case 2: $\mathbf{T}(\overline{\mathbf{X}})[\mathbf{g}] \circ \mathbf{T}', |\overline{\mathbf{X}}| > \mathbf{1}, \overline{\mathbf{X}} \supseteq \mathbf{Var}(\mathbf{g})$ In this and in the next case we may assume, without loss of generality, that g is composed of (in-)equalities and $T = \langle f_1, \ldots, f_m\rangle$, where $f_i = !^{s_i}X^j$, $\forall i$. Consider, for example:

$$\langle X^2, !X^2, X^3, !^2X^1\rangle[X^1 \neq X^3 \wedge X^1 \neq !X^2] \circ \langle h_1, h_2, h_3\rangle$$

This expression doesn't match $\mathcal{L}^c$. However, we can transform the guard in between into a filter prefixing the left tuple through an index substitution, i.e., an injective map $\phi : Var(T) \to \{1 \ldots m\}$ associating each X^i to the position of any of its occurrences in T. The picture below illustrates the idea.

$$\langle X^2, !X^2, X^3, !^2X^1\rangle[X^1 \neq X^3 \wedge X^1 \neq !X^2].$$

The following rule formalizes the move of a clause of a guard g to a filter prefixing T ($X^i \stackrel{\not\equiv}{=} !^kX^j \equiv X^j \stackrel{\not\equiv}{=} !^{-k}X^i$).

[6] if b is any bag, $\langle X^1, \ldots, X^1\rangle(b)$ is obtained from $[\bigwedge_{i:2\ldots m} X^1 = X^i]\langle b, \ldots, b\rangle$ by applying the m^{th} root to multiplicities of bag elements

Lemma 3 (transforming the infix predicate into a filter). *Let $\phi(i)$, $\phi(j)$ be any two occurrences of variables X^i, X^j, $i \neq j$, in tuple T.*

$$\underbrace{\langle \ldots, !^r X^{\overset{\phi(i)}{i}}, \ldots, !^s X^{\overset{\phi(j)}{j}}, \ldots \rangle}_{T}[X^i \overset{\neq}{=} !^k X^j, \ldots] \longrightarrow [X^{\phi(i)} \overset{\neq}{=} !^{k+r-s} X^{\phi(j)}]T[\ldots]$$

Corollary 1. *Let $T[g]$, with $T = \langle f_1, \ldots, f_m \rangle$ and g be a set of (in)equalities. If $\forall X^i \in Var(g)$ there is f_j, $f_j = !^s X^i$, then $\exists p\ T[g] \equiv [p]T$.*

Therefore, if $Var(T) \supseteq Var(g)$ the reiterated application of Lemma 3 eventually removes the infix guard from $T(\overline{X})[g] \circ T'$.

Applying Lemma 3 to the last example,we can proceed with Lemma 2.

$$\overset{Lem.3}{\longrightarrow} \quad [X^4 \neq !^2 X^3 \wedge X^4 \neq !^2 X^2]\langle X^2, !X^2, X^3, !^2 X^1 \rangle \circ \langle h_1, h_2, h_3 \rangle$$

$$\overset{Lem.2}{\longrightarrow} \quad [X^4 \neq !^2 X^3 \wedge X^4 \neq !^2 X^2]\langle \langle X^1, !X^1 \rangle \circ \langle h_2 \rangle, X^1 \circ \langle h_3 \rangle, !^2 X^1 \circ \langle h_1 \rangle \rangle$$

Case 3: $T(\overline{X})[g] \circ T'$, $\overline{X} \subset Var(g)$ This situation is the most complex one. The reason is that we must project on $\overline{X} = Var(T)$ the application of the filter g on T', in a symbolic way.

We assume that g is no further reducible, the composition to solve matches Definition 5 and none of lemmas presented so far applies (in particular Lemma 1, thus $T'_{Var(g)} = T'$).

Let $A \subseteq Var(g)$: we say $g_A = \{(X^i \overset{\neq}{=} !^r X^j) \in g | X^i, X^j \in A\}$ the restriction of g to variables A.

A first simplification comes from the fact that we may take out equalities of g. We recall that f^r denotes the index-restriction of f.

Property 15. Let $[g]T'$ meet Definition 5, $g_= \neq \emptyset$, $\overline{X}' = \overline{X} \cup Var(g_{\neq})$. Then

$$T(\overline{X})[g] \circ T' = (T[g_{\neq}])^r \circ T'_{\overline{X}'}$$

This nice property stems from the fact that (by Definition 5) T' components are equal (modulo successor) and in T only one symbol per equivalence class is used. Here are some examples of application of Property 15. For simplicity, in all the following examples $\overline{X} = \{X^1, \ldots, X^m\}$ ($T = T^r$).

1) $\langle X^1, S - X^1 \rangle [X^1 = X^2, X^1 = !X^3] \circ \langle S - !X^2, S - !X^2, S - X^2 \rangle \overset{Prop.15}{\longrightarrow}$
$\langle X^1, S - X^1 \rangle \circ \langle S - !X^2 \rangle \overset{Prop.14}{\longrightarrow} [X^1 \neq X^2]\langle S - !X^2, S \rangle$

2) $\langle X^1, !X^2 \rangle [X^1 \neq !X^2, X^2 = X^3] \circ \langle S - !X^2, S - X^2, S - X^2 \rangle \overset{Prop.15}{\longrightarrow}$
$\langle X^1, !X^2 \rangle [X^1 \neq !X^2] \circ \langle S - !X^2, S - X^2 \rangle \overset{Lem.3,Lem.2}{\longrightarrow} [X^1 \neq X^2]\langle S - !X^2, S - !X^2 \rangle$

3) $\langle X^2, X^2 \rangle [X^1 \neq X^2, X^2 \neq X^4, X^1 = X^3] \circ \langle S_1, S_1, S_1, S_1 \rangle \overset{Prop.15}{\longrightarrow}$
$(\langle X^2, X^2 \rangle [X^1 \neq X^2, X^2 \neq X^4])^r \circ \langle S_1, S_1, S_1, S_1 \rangle_{\{X^1, X^2, X^4\}} \equiv$
$\langle X^2, X^2 \rangle [X^1 \neq X^2, X^2 \neq X^3] \circ \langle S_1, S_1, S_1 \rangle$ (non-reducible with the available rules)

Main sub-case: $g = g_{\neq}$ We may therefore assume that g is a non-empty set of inequalities and focus on $\mathbf{\Pi}_{\overline{X}} \circ [g]T'$.
In general, we figure out that it holds

$$\overline{\mathbf{\Pi}_{\overline{X}} \circ [g]T'} \subseteq \overline{[g_{\overline{X}}]^r \ T'_{\overline{X}}} \tag{1}$$

Equation 1 says that, disregarding bag multiplicity, the projection of $[g]T'$ on $Var(T)$ is included in the restriction of $[g]$ to $Var(T)$ which applies to the image of $Var(T)$ on T'. It follows from our assumptions, by which: $g \Rightarrow g_{\overline{X}}$.
In the sequel, we characterize particular forms $[g]T'$ verifying

$$\mathbf{\Pi}_{\overline{X}} \circ [g]T' = k \cdot [g_{\overline{X}}]^r \ T'_{\overline{X}}, \ k \in \mathbb{N} \tag{2}$$

showing that (modulo some rewriting) we can always reduce our expression to such a form. An immediate corollary of (2) is:

Corollary 2. *Let $T = T(\overline{X})$, $\overline{X} \subset Var(g)$. If (2) holds and $[g] \circ T'$ is a simple tuple-prefix (Definition 5). Then:*
$T(\overline{X})[g] \circ T' = k \cdot T^r[g_{\overline{X}}]^r \circ T'_{\overline{X}}$, *where* $k = \frac{|[g]T'|}{|[g_{\overline{X}}]^r T'_{\overline{X}}|}$ *if* $|[g_{\overline{X}}]^r T'_{\overline{X}}| \neq 0$, *otherwise* $k = 0$

Corollary 2 outlines that we bring a composition to a form where the variables of the left tuple are a super-set of those of the infix predicate.
Consider this simple but interesting case where $T[g] : C^3 \to C^2$, $T' : C \to C^3$.

$$\langle X^1, X^2 \rangle [X^1 \neq X^3 \underset{g}{\wedge} X^2 \neq X^3] \circ \langle S - X^1, S - X^1 S - X^1 \rangle \underset{T'}{} \qquad |C| > 2$$

In this case condition (2) is not verified: $\mathbf{\Pi}_{\{X^1, X^2\}} \circ [g]T'(c)$ turns out to be, for any $c \in C$, $|C| > 2$:

$$(|C| - 2) * \sum_{c' \in C, c' \neq c} \langle c', c' \rangle + (|C| - 3) * \sum_{c', c'' \in C, c' \neq c, c'' \neq c, c' \neq c''} \langle c', c'' \rangle$$

The restriction of $g[T']$ to $\overline{X} = \{X^1, X^2\}$ used on the right-hand side of (2), instead, is $\langle S - X^1, S - X^1 \rangle$ (in this particular case, $g_{\overline{X}} = \{\}$). As the example suggests, we should distinguish (in g) the case $X^1 = X^2$ from the case $X^1 \neq X^2$.

Since we are assuming that the requirements of Definition 5 are met, we may conveniently study $[g]T'$ as a system of inequalities (in case of ordered classes, the successors are all expressed modulo-$|C|$) among $Var(g)$ variables, with the implicit constraints $X^i \in h_i(c)$, $\forall X^i \in Var(g)$, h_i being the i-th component of T'. Thus, $[g]T'(c)$ is a type-set bag which corresponds to the system's solutions, while $\mathbf{\Pi}_{\overline{X}} \circ [g]T'(c)$ is a bag representing their projection on subset $\overline{X}$.

Due to the symmetry of g and to the fact that functions h_i are equal (modulo successor), we can abstract from both c and i and consider any h_i as a (parametric) set of known size.

We put some conditions on g to ensure that the projection of the inequality system's solutions on $\overline{X}$ is the parametric multi-set consisting of k instances of the solutions of the *sub-system* restricted to $\overline{X}$, i.e., $[g_{\overline{X}}]^r \ T'_{\overline{X}}$.

We consider first an unordered color-class C. In that case, $h_i = h$, $\forall i$, and we may represent g as an undirected simple graph whose vertices are $Var(g)$ and whose edges connect vertices corresponding to variables that are required to be different by a term of g. Abusing notation, we denote the same way g and the corresponding graph, leaving the context to disambiguate. Thus, we may interpret $g_{\overline{X}}$ as a (proper) subgraph of g.

A few basic results of graph-colouring theory turn out to be useful for our purposes (refer to [10] for all the details). Given a graph $G = (V_G, E_G)$, where V_G is a non-empty finite set and E_G a set of unordered pairs xy, $x, y \in V(G)$, $x \neq y$, and $\lambda \in \mathbb{N}$, we define a λ-colouring of G as a map $\varphi : V_G \to \{1, 2, \ldots, \lambda\}$ that assigns adjacent vertices of G different values. We are interested in the number of different λ-colourings of G, denoted $P(G, \lambda)$. The minimum value of λ such that G admits a λ-colouring is the *chromatic number of* G, denoted χ_G. The value $|V(G)|$ is the order of G. We say that G is empty if $E(G) = \emptyset$, complete (or clique) if any two vertices are adjacent. A clique of order n is denoted K_n. We use a few operations on graphs: let x, y be two vertices of G, $G \cdot xy$, $xy \notin E(G)$, is the graph obtained from G by merging x and y and leaving one occurrence of possible resulting multiple edges; $G - xy$, $xy \in E(G)$, the graph obtained by removing edge xy; $G + xy$, $xy \notin E(G)$, the graph obtained by adding edge xy.

$P(G, \lambda)$ can be expressed as a polynomial in λ, called chromatic polynomial of G. This is interesting, since it gives us the possibility to express a composition's result in a parametric way. Computing χ_G is around $O(2^n)$, and computing $P(G, \lambda)$ is at least as complex. [10] shows, however, that for large classes of graphs you can compute $P(G, \lambda)$ very efficiently, e.g., exploiting their modular structure. How to compute the chromatic polynomial is out of the scope of the paper, however, here are a few intuitive properties used in the sequel.

- (Fundamental reduction theorem) let x, y be two non-adjacent vertices of G: $P(G, \lambda) = P(G \cdot xy, \lambda) + P(G + xy, \lambda)$.
- Let G include K_r and G' be obtained from G by adding a new vertex x which is (uniquely) linked to all vertices of K_r. Then $P(G', \lambda) = P(G, \lambda)(\lambda - r)$
- Some known polynomials: $P(K_r, \lambda) = \lambda(\lambda - 1) \cdots (\lambda - r + 1)$; if G is an empty graph of order n, $P(G, \lambda) = \lambda^n$; ...

Any solution of the inequality system $[g]T'$ (g, from now on) corresponds in fact to a λ-colouring of g with $\lambda = |h|$, therefore, denoting with $Sol(g, \lambda)$ the number of solutions of the inequality system, $P(g, \lambda) = Sol(g, \lambda)$.

Lemma 4. *Let $[g]T'$ be a simple tuple-prefix, $\overline{X} \subset Var(g)$, and $g_{\overline{X}}$ be a clique. Then, equation (2) holds and (Corollary 2) $k = \frac{P(g,\lambda)}{P(K_{|\overline{X}|},\lambda)}$ if $\lambda \geq |\overline{X}|$, otherwise $k = 0$, where $\lambda = |h_i|$, for any h_i in T'.*

It is sufficient to observe that for any two distinct λ-colourings of $g_{\overline{X}}$ there are (obviously) the same number of λ-colourings of g that include them. Moreover, we know that $\chi_{K_r} = r$ and $\chi_g \geq \chi_{g_{\overline{X}}}$.

We are always able to reduce a composition to the condition that meets Lemma 4 by linking non-adjacent vertices of $g_{\overline{X}}$. Two such vertices are two unrelated variables $X^i, X^j \in \overline{X}$, therefore we rewrite $T \circ [g][T']$ into the equivalent

sum $T \circ [g \wedge X^i \neq X^j]T' + T \circ [g \wedge X^i = X^j]T'$. Note that $g \wedge X^i = X^j$, after symbol replacement and removal of redundant inequalities (according to Definition 5) exactly corresponds to $g \cdot X^i\ X^j$.

By recursively rewriting and applying variable substitutions accordingly in T we eventually get sub-compositions solvable with the lemmas above. Let us instantiate this simple procedure and Lemma 4 on the last example.

$$\langle \underset{T}{X^1, X^2} \rangle [X^1 \neq X^3 \underset{g}{\wedge} X^2 \neq X^3] \circ \langle S - X^1, \underset{T'}{S - X^1 S}, -X^1 \rangle \quad \lambda = |C| - 1\ (> 1)$$
$$\equiv \langle X^1, X^2 \rangle [X^1 \neq X^3 \wedge X^2 \neq X^3 \wedge X^1 \neq X^2] \circ T' + \langle X^1, X^1 \rangle [X^1 \neq X^3] \circ T'$$
$$\equiv \langle X^1, X^2 \rangle \circ (\lambda - 2)[X^1 \neq X^2] \langle S - X^1, S - X^1 \rangle + \langle X^1, X^1 \rangle \circ (\lambda - 1) \langle S - X^1 \rangle$$
$$\equiv (\lambda - 2)[X^1 \neq X^2] \langle S - X^1, S - X^1 \rangle + (\lambda - 1)[X^1 = X^2] \langle S - X^1, S - X^1 \rangle$$

where $\lambda - 2 = \frac{P(K_3,\lambda)}{P(K_2,\lambda)}$ and $\lambda - 1 = \frac{P(K_2,\lambda)}{P(K_1,\lambda)}$. *Dealing with an ordered class C.* What if predicate g of the filter prefixing tuple T' is defined on an ordered class? In theory, the situation is even simpler because we can always rewrite any inequality into a disjunction of equalities: for example, if $|C| = 3$: $X^1 \neq !^2 X^2 \equiv X^1 =!^2 X^2 \vee X^1 = X^2$. So, by expanding $T \circ [g]T'$ accordingly we eventually get a solvable form.

This approach, however, may be inefficient and is not parametric. Therefore, we slightly extend the technique based on graph-representation of g to ordered classes. The graph representing g now has as vertices the symbols $Symb(g) = \{!^r X^j\}$ occurring in g and as edges, in addition to inequalities in g, those implied by them: for any two symbols $!^r X^j, !^s X^j \in Symb(g)$, there is a corresponding edge in the graph[7]. In this case, a λ-colouring of g doesn't necessarily match a solution of the inequality system, i.e., $P(g, \lambda) \geq Sol(g, \lambda)$,

Due to the circularity of C there are a number of equivalent representations for g. A first result concerns those cases where g may be expressed in a form such that $|Var(g)| = |Symb(g)|$, i.e., for each variable there is one corresponding symbol. In this case the previous results, in particular Lemma 4, still hold. When $|Var(g)| < |Symb(g)|$ instead, an extension of Lemma 4 is needed.

Lemma 5. *Let $[g]T'$ be a simple tuple-prefix and $\overline{X} \subset Var(g)$. If a) for each $X^i \in Var(g) \backslash \overline{X}$ any two inequalities between $\overline{X}$ and X^i use the same symbol $!^r X^i$ and b) $g_{\overline{X}}$ is a clique, then equation (2) holds and $k = \frac{Sol(g,\lambda)}{Sol(g_{\overline{X}},\lambda)}$ if $Sol(g_{\overline{X}}, \lambda) > 0$, otherwise $k = 0$, where $\lambda = |h_i|$, for any h_i in T'.*

We can always rewrite any predicate g so that it meets condition a) of Lemma 5. Condition a) is redundant if $|\overline{X}| = 1$.

Observe that for computing $Sol(g, \lambda)$, for any inequality graph g, we can use similar basic techniques as for the chromatic polynomial. See Fig. 2, step 7 and [8], Appendix A.3 for a few examples of application of the last two lemmas.

[7] Let $succ_{min}$ and $succ_{max}$ be the smallest and largest successor index in the formula; we assume that $succ_{max} - succ_{min} < |C|$, so that for each two symbols $!^r X^j, !^s X^i$, and for each c, $!^r X^j(c) \neq !^s X^j(c)$; as a consequence we may have consider different cases, depending on the size of C

Case 4: $\mathbf{T[g]} \circ \mathbf{T'},\ \mathbf{Var(T)} \cap \mathbf{Var(g)} = \emptyset$. We may reduce this case to one solvable with Lemma 1. Indeed, we can express (modulo a tuple permutation) $[g] \circ T'$ as a Cartesian product $\langle [g]^r T'_{Var(g)}, T'_{\neg Var(g)} \rangle$, where $T'_{Var(g)}$ is the image of g on T'. If $[g]^r T'_{Var(g)}$ is simple we can directly use Lemma 1. For instance:

$\langle X^1 \rangle [X^2 \neq X^3] \circ \langle S - X^1, S, S \rangle \rightarrow \langle X^1 \rangle \circ \langle S - X^1, [X^1 \neq X^2] \langle S, S \rangle \rangle \rightarrow$
$\lambda(\lambda - 1) \cdot \langle S - X^1 \rangle \quad (\lambda = |C|)$

5 Applications

This section presents two application examples: the symbolic verification of invariants in SN models, and the reduction of nets by agglomeration of transitions, enabling more efficient qualitative analysis (as in behavior preserving reductions [3,11]) and quantitative analysis (e.g. elimination of immediate transitions in Stochastic SN models, extending the technique defined for GSPNs in [1]).

Verification of P and T-invariants. P and T-semiflows inducing invariant properties in PNs have been introduced in mid 80s and extended to HLPN in 90s [13]. Algorithms to compute a generating family of P and T-semiflows of PN exist [9] and are implemented in several tools; they can be applied to the unfolding of an HLPN model, while the automatic derivation of high-level (symbolic) P or T-semiflows is still an open problem, unless restrictions are imposed on the HLPN formalism (as in [12]). Often the modeler is aware of which invariant properties should satisfy a given model, thus the ability to automatically check whether a P or T-indexed vector of functions is a P or T-semiflow is interesting. The symbolic calculus implemented in the SNexpression tool [7], extended with the composition presented in this paper, allows to implement such automatic check. Let us illustrate how this can be done on the example net in Fig. 1; for the sake of space only T-invariants are shown, while P-invariants are discussed in [8].

Definition 6 (T-semiflow of HLPN models).

Let $\mathbf{y}$ *be a T-indexed vector of functions with color domain* $\mathcal{C}(\mathbf{y})$, $\mathbf{y}[t] : \mathcal{C}(\mathbf{y}) \rightarrow Bag[\mathcal{C}([t)], t \in T$; $\mathbf{y}$ *is a T-semiflow if* $\mathbf{C} \circ \mathbf{y} = \mathbf{null_y}$. *A T-semiflow defines a (parametric) set of transition instances; given a color* $c \in \mathcal{C}(\mathbf{y})$ *if all the transition instances in* $\mathbf{y}(c)$ *can fire (in any order) starting from marking* $\mathbf{m}$, *they bring back the model to the same marking* $\mathbf{m}$.

Where $\mathbf{null_y}$ is a P-indexed vector of functions such that $\mathbf{null_y}[p]$ is a constant function mapping any $c \in \mathcal{C}(\mathbf{y})$ into a constant empty Bag on $\mathcal{C}(p)$. In Table 1 two T-semiflows of the producers-consumers SN are listed. Their interpretation is quite simple[8]: In order to reproduce a marking the pointers first and last should be increased by one $|C1|$ times, hence the producers should produce $|C1|$ items and put them into the buffer, and the consumers should get $|C1|$ items from the buffer. In T1 the same producer fills completely the buffer and a consumer with the same id of the producer empties it (it represents $|C2|$ invariants in the unfolded net). In T2 any combination of $|C1|$ producers and $|C1|$ consumers (satisfying the invariant guard g') is possible. The expressions

[8] Initialization transition init cannot belong to any T-invariant, while additional invariants involving ProduceBurst exist, but are not included for the sake of space.

Transition (Domain)	T1 C_2	T2 (Hp: $\|C1\| = 4$) C_2^8, C_1 $[g']$
Produce (C_2)	$\|C_1\|\langle X_2^1\rangle$	$\langle X_2^1\rangle + \langle X_2^3\rangle + \langle X_2^5\rangle + \langle X_2^7\rangle$
Put (C_1, C_2)	$\langle S_{C_1}, X_2^1\rangle$	$\sum_{i=0}^{3}\langle !i\ X_1^1, X_2^{2i+1}\rangle$
Get $[g](C_1, C_2^2)$	$\langle S_{C_1}, X_2^1, X_2^1\rangle$	$\sum_{i=0}^{3}\langle !i\ X_1^1, X_2^{2i+1}, X_2^{2(i+1)}\rangle$
End_Process (C_2)	$\|C_1\|\langle X_2^1\rangle$	$\langle X_2^2\rangle + \langle X_2^4\rangle + \langle X_2^6\rangle + \langle X_2^8\rangle$

$p(k) = \bigwedge_{i=0}^{k}\left(\bigvee_{j=1}^{2} X_2^{2i+1} \in C_{2j} \wedge X_2^{2(i+1)} \in C_{2j}\right);\ g = p(0);\ g' = p(|C_1| - 1)$

Table 1. Two T-semiflows of the Producers-Consumers SN

allowing to verify that vectors T_j correspond to parametric T-semiflows are: $\forall p \in P, \sum_{t\in T} \mathbf{C}[p,t] \circ T_j[t] = \epsilon_{\mathcal{C}(p)}$; the function $T_j[t]$ has domain $\mathcal{C}(T_i)$ and codomain $\mathcal{C}(t)$, so that the domains of the functions to be composed are coherent. Observe that T_2 represents 625 T-invariants in the unfolded net when $|C_1| = 4$, $|C_{21}| = 1$ and $|C_{22}| = 2$.

Let's check whether the transitions in T1 produce a null change on all places.

Producing: W^+(Put,Producing) $\circ\, T1$[Put] - W^-(Produce, Producing) $\circ\, T1$[Produce] $= \langle X_2^1\rangle \circ |C_1| \langle X_2^1\rangle - \langle X_2^1\rangle \circ |C_1| \langle X_2^1\rangle = \epsilon_{C_1}$ (the equations for Wait_to_insert, Wait_to_extract and Processing are very similar).

buffer: W^+(Put, buffer) $\circ\, T1$[Put] $- W^-$(Get, buffer) $\circ\, T1$[Get] $= \langle X_1^1, X_2^1\rangle \circ \langle S_{C_1}, X_2^1\rangle - \langle X_1^1, X_2^1\rangle[g] \circ \langle S_{C_1}, X_2^1, X_2^1\rangle = \langle S_{C_1}, X_2^1\rangle - \langle S_{C_1}, X_2^1\rangle = \epsilon_{C_1,C_2}$

full: W^+(Put, full) $\circ\, T1$[Put] $- W^-$(Get, full) $\circ\, T1$[Get] $= \langle S_{C_\bullet}\rangle \circ \langle X_1^1, X_2^1\rangle - \langle S_{C_\bullet}\rangle[g] \circ \langle X_1^1, X_2^1, X_2^1\rangle = \epsilon_{C_\bullet}$ (the equation for empty is very similar)

first: W^+(Get, first) $\circ\, T1$[Get] $- W^-$(Get, first) $\circ\, T1$[Get] $= \langle !X_1^1\rangle[g] \circ \langle S_{C_1}, X_2^1, X_2^1\rangle - \langle X_1^1\rangle[g] \circ \langle S_{C_1}, X_2^1, X_2^1\rangle = \langle S_{C_1}\rangle - \langle S_{C_1}\rangle = \epsilon_{C_1}$ (the equation for last is very similar)

T2 has color domain C_1, C_2^8 where $8 = 2|C_1|$, completed by guard g' (see Table1); and similar formulae as for T1 apply to it, let us consider the expressions for just a few places to illustrate the type of composition to be solved.

buffer: W^+(Put, buffer) $\circ\, T2$[Put] $- W^-$(Get, buffer) $\circ\, T2$[Get] $= \langle X_1^1, X_2^1\rangle \circ (\langle X_1^1, X_2^1\rangle[g'] + \langle !X_1^1, X_2^3\rangle[g'] + \langle !^2X_1^1, X_2^5\rangle[g'] + \langle !^3X_1^1, X_2^7\rangle[g']) - (\langle X_1^1, X_2^1\rangle[g] \circ (\langle X_1^1, X_2^1, X_2^2\rangle[g'] + \langle !X_1^1, X_2^3, X_2^4\rangle[g'] + \langle !^2X_1^1, X_2^5, X_2^6\rangle[g'] + \langle !^3X_1^1, X_2^7, X_2^8\rangle[g']) = \epsilon_{C_1,C_2}$

Wait_to_insert: W^+(Produce, Wait_to_insert) $\circ\, T2$[Produce] $- W^-$(Put, Wait_to_insert) $\circ\, T2$[Put] $= \langle X_2^1\rangle \circ (\langle X_2^1\rangle + \langle X_2^3\rangle + \langle X_2^5\rangle + \langle X_2^7\rangle)[g'] - (\langle X_2^1\rangle \circ (\langle X_1^1, X_2^1\rangle[g'] + \langle !X_1^1, X_2^3\rangle[g'] + \langle !^2X_1^1, X_2^5\rangle[g'] + \langle !^3X_1^1, X_2^7\rangle[g']) = \epsilon_{C_2}$

Observe that in all cases where transition Get is involved we have an infix guard g whose terms check which static sublcass X_2^1 and X_2^2 belong to: it is easily handled both in T1 (where the involved elements of the right tuple are equal) and in T2 (where the T-semiflow guard g' implies that g is satisfied).

Finally, note that depending on the actual binding of the T-psemiflow variables to colors, functions T_j[Produce] and T_j[End_Process] map to a $Bag^+[C_2]$ that may have coefficients greater than one. Hence the composition presented in this paper is required to verify the invariant.

Reduction of transitions. Another application of multiset composition in coloured PN is structural reduction. For instance, in [11] a symbolic reduction technique (based on the capability to syntactically solve the composition) is applied to models used for software verification. To allow for a symbolic treatment, the authors propose Quasi-well formed Nets, a tight restriction of SNs. Moreover,

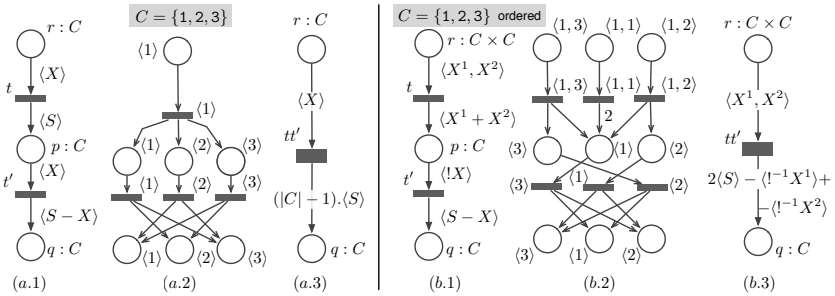


Fig. 3. Agglomeration: .1 Original SN; .2 Unfolded subnet; .3 Reduced folded net.

the syntactical arc function composition requires further restrictions on the shape of manageable arc functions.

Transitions agglomeration is a useful reduction: it consists of merging causally connected transitions. The aim is to reduce transition instances interleaving, preserving specific system properties. A number of techniques have been proposed, with different properties and applicability conditions. Common to most proposals is this applicability scenario: a set H of transitions put tokens into a place p, leading to a new marking of p from which it is possible to restore its initial state *only* through the firing of a newly enabled transition t'; in that case, the firing of any transition in H may immediately cause the firing of t', without interfering with any other transition firing. In this paper, we do not propose a general agglomeration technique for SN, rather, we illustrate how to perform agglomeration by symbolically deriving the arc functions for the new transition. We also provide an intuitive structural condition on SN arc functions, which ensures the applicability of agglomeration in a limited set of cases.

Figure 3(a.1) shows a simple example where the transitions t and t' can be agglomerated. Figure 3(a.2) depicts a part of the net unfolding connected to the instance $\langle 1\rangle$ of t. It is possible to see that, if p is initially empty, after the firing of instance $\langle 1\rangle$ of t, we can safely fire immediately all the newly enabled instances of t' restoring the p empty state. This is true for any instance $\langle i\rangle$ of t. Figure 3(a.3) is the reduced colored subnet, where the size of class C is a parameter.

The reduced subnet can be obtained without unfolding it: indeed, for this type of agglomeration, the following formula allows to symbolically compute the arc functions of the reduced subnet[9]:

$$W^-(tt', r) = W^-(t, r); \quad W^+(tt', q) = W^+(t', q) \circ W^-(t', p)^t \circ W^+(t, p)$$

In $W^+(tt', q)$ the rightmost composition provides the instances of t' enabled by the firing of an instance of t, through p. Finally composing $W^+(t', q)$ with this result (firing t' for all those instances) we obtain a function providing the multiset of tokens to be added in q. For the example of Fig.3(a) $W^-(tt', r) = \langle X\rangle$, while:

$$W^+(tt', q) = \langle S - X\rangle \circ \langle X\rangle^t \circ \langle S\rangle = \langle S - X\rangle \circ \langle S\rangle = (|C| - 1).\langle S\rangle \overset{|C|=3}{=} 2.\langle S\rangle$$

Figure 3(b.1) shows another example of reduction. The formula for the agglomeration can be also used here. The agglomeration is valid for any $|C| \geq 2$:

[9] $W^-(t', p)^t : \mathcal{C}(p) \to Bag[\mathcal{C}(t)]$ denotes the transpose of $W^-(t', p)$; the rules to symbolically compute the transpose of an arc function are defined in [4].

$W^+(tt',q) = \langle S-X\rangle \circ \langle !X\rangle^t \circ \langle X^1+X^2\rangle = \langle S-X\rangle \circ \langle !^{-1}X\rangle \circ \langle X^1+X^2\rangle = \langle S-X\rangle \circ \langle !^{-1}X^1 + !^{-1}X^2\rangle = \langle S - !^{-1}X^1\rangle + \langle S - !^{-1}X^2\rangle = 2.\langle S\rangle - \langle !^{-1}X^1\rangle - \langle !^{-1}X^2\rangle$

For these examples where p is the only input place of t' and is the only output place of t, we can confidently state that if (1) the instances of transition t' are not in conflict with each other (a situation called *auto-conflict*, which can be checked symbolically on the net structure) and (2) if all the tokens put into p by t can be completely consumed by (one or more instances of) t', then we can aggregate t and t'. To verify the second condition it is sufficient to check the following equality: $W^+(t,p) = W^-(t',p) \circ W^-(t',p)^t \circ W^+(t,p)$. The following property characterizes the form for the input function $W^-(t',p)$ which guarantees the agglomeration conditions(s). Let the identity on D be $Ide_D(d) = 1 \cdot d$, $\forall d \in D$.

Property 16. Let $\mathcal{C}(p)' = \overline{W^+(t,p)(\mathcal{C}(t))}$ be the (support of the) image of $W^+(t,p)$. $W^+(t,p) = W^-(t',p) \circ W^-(t',p)^t \circ W^+(t,p) \Leftrightarrow \forall c \in \mathcal{C}(p)' W^-(t',p) \circ W^-(t',p)^t(c) = 1 \cdot c$

(i.e. the restriction of $W^-(t',p) \circ W^-(t',p)^t$ to $\mathcal{C}(p)'$ is the identity).

Observe that with the assumption that p is the unique intermediate place between t and t', this property includes also the condition of no autoconflicts among t' instances ($= \overline{W^-(t',p)^t \circ W^-(t',p)} - \overline{Ide}$, simplified formula from [5]).

The condition holds for both the examples discussed above, and this can be checked symbolically exploiting the composition presented in this paper.
Finally, let us consider a third example, fully exploiting the composition technique presented in this paper. The subnet schema is the same but for t', which has one more output place, q'. The arc functions for this example are:

$$W^-(t,r) = \langle X\rangle \quad W^+(t,p) = \langle S - !X, S, S-X\rangle \quad W^+(t',q) = \langle !X^2, X^2\rangle[X^1 = X^3]$$
$$W^-(t',p) = \langle !X^3, X^2, X^1\rangle \qquad W^+(t',q') = 2 \cdot \langle X^1, X^2\rangle[X^1 \neq X^2 \wedge X^2 \neq X^3]$$

The arc functions for the agglomerated transition tt' are ($\lambda = |C|$ with $|C| > 2$):

$$W^-(tt',r) = \langle X\rangle \qquad W^+(tt',q) = (\lambda - 1)[X^1 = !X^2]\langle S, S\rangle$$
$$W^+(tt',q') = 2(\lambda-1)\langle S-X, X\rangle + 2(\lambda-2)[X^1 \neq X^2]\langle S-X, S-X\rangle$$

The detailed explanation of the steps leading to the result, involving the application of Lemmas 1 to 4, is in Appendix A.3 of [8]. Observe that the result requires to extend the SN formalism, to allow arc expressions with filters.

6 Conclusions and future work

This paper reaches the goal of completing the symbolic manipulation of SN arc functions defining a procedure for the composition of functions mapping on Bags. This enables a number of applications that couldn't be treated with the operations defined so far. The implementation of a new version of SNexpression including this kind of composition is ongoing. The approach suggests an extension to the SN formalism itself, to allow filters in arc expressions: this feature is now implemented in the GreatSPN Graphical User Interface. The presented results are often parametric in color class cardinality, possibly with some constraints on cardinality lower bounds.

Future work is devoted to extend the possible applications of the results presented in this paper, in particular as concerns a generalization of net reduction methods proposed in literature to a wider class of nets: for instance, we expect that our proposal may allow to relax some restriction posed in [11]. Another interesting topic is the elimination of immediate transitions in Stochastic SNs (in order to apply analysis algorithms which do not work in presence of immediate transitions): this is a challenging issue since in general it may involve marking dependent sequences of immediate transition firings triggered by one timed transition, as in the case of the example in Fig. 1.

References

1. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley —& Sons Ltd, 1995.
2. E. G. Amparore, G. Balbo, M. Beccuti, S. Donatelli, and G. Franceschinis. *30 years of greatSPN*, pages 227–254. Springer London, 2016.
3. G. Balbo and M. Silva, editors. *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques*. Editorial Kronos, Zaragoza, Spain, 1998. Available online.
4. L. Capra, M. De Pierro, and G. Franceschinis. A high level language for structural relations in Well-Formed Nets. In G. Ciardo and P. Darondeau, editors, *Int. Conf. on Applications and Theory of Petri Nets 2005*, pages 168–187. Springer, 2005.
5. L. Capra, M. De Pierro, and G. Franceschinis. Computing structural properties of Symmetric Nets. In *Proc. of the 15th International Conference on Quantitative Evaluation of Systems*, QEST 15, Madrid, ES, 2015. IEEE CS.
6. L. Capra, M. De Pierro, and G. Franceschinis. Deriving symbolic and parametric structural relations in Symmetric Nets: Focus on composition operator. Technical Report TR-INF-2019-03-01-UNIPMN, DiSIT,UPO, Alessandria, Italy, 2019.
7. L. Capra, M. De Pierro, and G. Franceschinis. SNexpression: A symbolic calculator for Symmetric Net expressions. In *Proceedings PN2020*, volume 12152 of *LNCS*, pages 381–391, Cham, CH, june 2020. Springer.
8. L. Capra, M. De Pierro, and G. Franceschinis. General composition for symmetric net arc functions with applications. Technical Report TR-INF-2021-05-01-UNIPMN, Computer Science Inst., DiSIT, Univ. del Piemonte Orientale, 2021.
9. J.M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. a comparative study of algorithms for computation of minimal p-semiflows. In Rozenberg G., editor, *Advances in Petri Nets 1990. ICATPN 1989*, volume 483 of *LNCS*. Springer, Berlin, Heidelberg, 1991.
10. F M Dong, K M Koh, and K L Teo. *Chromatic Polynomials and Chromaticity of Graphs*. WORLD SCIENTIFIC, 2005.
11. S. Evangelista, S. Haddad, and J. F. Pradat-Peyre. Syntactical colored Petri nets reductions. In *Automated Technology for Verification and Analysis*, pages 202–216, Berlin, Heidelberg, 2005. Springer.
12. S. Evangelista, C. Pajault, and J. F. Pradat-Peyre. A simple positive flows computation algorithm for a large subclass of colored nets. In *FORTE 2007*, pages 177–195, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
13. K. Jensen. An introduction to the theoretical aspects of coloured Petri nets. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium, The Netherlands, June 1-4, 1993*, volume 803 of *LNCS*, pages 230–272. Springer, 1993.