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

Deriving Symbolic and Parametric Structural Relations in Symmetric Nets: Focus on Composition Operator

L. Capra, M. De Pierro, G. Franceschinis (inserire indirizzo e-mail, inserire indirizzo e-mail, giuliana.franceschinis@uniupo.it)

TECHNICAL REPORT TR-INF-2019-03-01-UNIPMN
(March 2019)

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

Deriving Symbolic and Parametric Structural Relations in Symmetric Nets: Focus on Composition Operator

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

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

Giuliana Franceschinis,
DiSIT, Istituto di Informatica
Università del Piemonte Orientale
Alessandria, Italia
giuliana.franceschinis@uniupo.it

1. Introduction

The work presented in this report concerns the most recent developments of a calculus used to derive (mainly structural) properties of Symmetric Nets (SNs), formerly known as Well-Formed Nets (WN)[6], a kind of High Level Petri Net formalism introduced about 30 years ago, for which several efficient analysis algorithms and tools have been developed. The most relevant feature of SNs is the possibility to represent in a quite natural way systems with symmetric structure and behavior, and to exploit such characteristics to efficiently analyse SN models. This feature extends to Stochastic SNs (SSNs), an extension of the formalism allowing to generate a stochastic process (precisely a Continuous Time Markov Chain - CTMC) from an high level SSN model representation, and to automatically reduce the state space exploiting the model symmetries.

The calculus that is being extended in this report was introduced about 15 years ago[7, 4, 3], with the aim of efficiently computing useful structural properties of SNs, representing them in a symbolic and compact form. A SN model consists of places (modeling the state) and transitions (modeling activities and events) connected by arcs decorated with arc expressions. The calculus proposed in this report allows to manipulate and combine the arc expressions through operators like the transpose, the composition or the intersection, in order to derive new expressions representing various structural properties in symbolic form[5]. The calculus is parametric in the size of color classes: this allows to obtain results that are valid for a whole family of models. The underlying language resembles the arc expressions language and extends it by including *filters* (similar to guards but left-composed with tuples) and intersection of basic SN functions. The calculus has been implemented in a library whose functions can be accessed through a command-line interface: the tool[1] is called SN-expression[8, 5]. Finally it has been shown that this calculus can also be useful when solving models with very large state spaces using a set of ordinary differential equations: the intrinsic symmetries of these models can be exploited to generate a reduced set of equations allowing to derive the same results that could be obtained from the complete set of equations [2, 1] with a reduced computation effort.

The present work completes the rules to solve the composition of any pair of function tuples mapping on sets and in some case also on multisets. It is organized as follows: in Section 2 the main definitions an notations used throghout the report and the language used to compute structural relations are introduced, in Section 3 some preliminary notions on the method used to transform expressions through appropriate rewriting rules is discussed, and some expression forms with specific properties (needed in the next section) are introduced. Section 4 is the core of the paper and discusses the different possible cases arising when composing two tuples (with both guards and filters): the rewriting rules may in some case require an operation of tuple expansion before composing and successive projection to obtain the final result. In Section 5 we summarize the results presented in the report, provide hints on its applicative relevance and discuss possible future works. In the Appendix some rewriting rules already presented in previous publications are reported, to make the report self-contained, moreover a new rewriting rule is proposed that may conveniently replace a set of other rules illustrated in the report; the choice of the rule to apply in practice can then be driven by efficiency considerations.

2. Main definitions and notations

2.1. Symmetric Nets

In this section the SN formalism is briefly presented through the example in Fig.1 and the notation used throughout the paper is introduced (the syntax and notation is summarized in Table 1). The SN formalism is an High-level Petri Net (HLPN) formalism with syntax constraints that allow to automatically discover and exploit symmetries in the model behavior. Being a HLPN formalism, it features parametric and compact modeling of systems composed of several similarly behaving elements, as well as an intuitive graphical representation in the form of a bipartite graph. A SN model state is represented through the *colored marking* of *places*, while the possible state changes are represented by *transition instances* (which are pairs: transition, binding). Places are graphically represented by circles, annotated with the corresponding *color domain* (denoted $\mathcal{C}(p)$). Transitions are represented by (black or white)

[1]SNexpression can be downloaded from www.di.unito.it/$\sim$depierro/SNex.

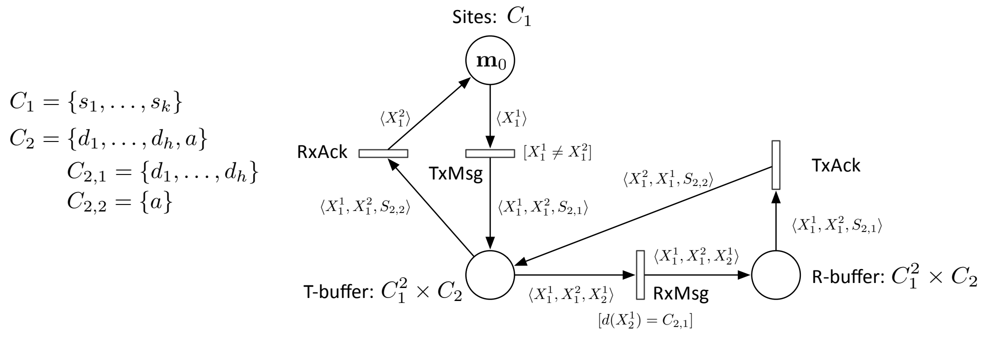

$C_1 = \{s_1, \dots, s_k\}$
$C_2 = \{d_1, \dots, d_h, a\}$
$C_{2,1} = \{d_1, \dots, d_h\}$
$C_{2,2} = \{a\}$
Sites: C_1
$\mathbf{m}_0$
$\langle X_1^2 \rangle$
$\langle X_1^1 \rangle$
RxAck
TxMsg
$[X_1^1 \neq X_1^2]$
TxAck
$\langle X_1^2, X_1^1, S_{2,2} \rangle$
$\langle X_1^1, X_1^2, S_{2,2} \rangle$
$\langle X_1^1, X_1^2, S_{2,1} \rangle$
$\langle X_1^1, X_1^2, S_{2,1} \rangle$
$\langle X_1^1, X_1^2, X_2^1 \rangle$
T-buffer: $C_1^2 \times C_2$
$\langle X_1^1, X_1^2, X_2^1 \rangle$
RxMsg
R-buffer: $C_1^2 \times C_2$
$[d(X_2^1) = C_{2,1}]$

Figure 1: A simple SN example.

rectangles; each transition has an associated set of typed *variables* and a *predicate* (called transition *guard*) which together define its *color domain*. Places and transitions are nodes of a graph, and are connected through arcs annotated with functions, represented in the form of expressions whose terms are (weighted and guarded) tuples of *basic functions* (the tuple notation stands for the Cartesian product of its elements). Transition guards are boolean expressions built on *standard predicate* terms. Place and transition color domains are defined in terms of *basic color classes* which are disjoint, finite not empty sets. Let us consider the model in Fig.1: it comprises place Sites with color domain C_1, T-buffer and R-buffer with color domain $C_1 \times C_1 \times C_2$, where $C_1 = \{s_1, \dots, s_k\}$ and $C_2 = \{d1, \dots, d_h, a\}$ are basic color classes: the former represents a set of sites that can exchange messages, the latter represents messages exchanged by sites, and is partitioned into two *static subclasses*, $C_{2,1} = \{d1, \dots, d_h,\}$ representing data messages and $C_{2,2} = \{a\}$ representing an acknowledge message. The marking of a place is a multiset of colored tokens (i.e. a set with possible repetition of *tokens* with same *color*) from the corresponding color domain (the set of all multisets that may be markings of p is denoted $Bag[\mathcal{C}(p)]$). The transitions of the SN in Fig.1 are TxMsg, whose variables are X_1^1 and X_1^2 both of type C_1 (indicated by the subscript 1, and distinguished by means of the superscript), and with guard $[X_1^1 \neq X_1^2]$, RxMsg, whose variables are X_1^1, X_1^2 both of type C_1 and X_2^1 of type C_2, and with guard $[X_2^1 \in C_{2,1}]$. Transitions TxAck and RxAck have a constant guard $true$ and two variables of type C_1. Summarizing, $\mathcal{C}(\mathsf{TxMsg}) = \{\langle s_i, s_j \rangle \in C_1 \times C_1\}$ with the constraint $i \neq j$ due to the predicate $\Phi(\mathsf{TxMsg}) = [X_1^1 \neq X_1^2]$; $\mathcal{C}(\mathsf{RxMsg}) = \{\langle s_i, s_j, m \rangle \in C_1 \times C_1 \times C_2\}$ with the constraint $m \in C_{2,1}$ due to the predicate $\Phi(\mathsf{RxMsg}) = [d(X_2^1) = C_{2,1}]$. Finally $\mathcal{C}(\mathsf{TxAck}) = \mathcal{C}(\mathsf{RxAck}) = C_1 \times C_2$.

A binding of a given transition is an association of colors to its variables (from the appropriate basic color class), it leads to a valid transition instance only if it satisfies the guard. A pair transition-binding is called transition instance: an example of transition instance is (TxMsg,$\langle X_1^1 = s_1, X_1^2 = s_2 \rangle$), which is valid because it satisfies the transition guard $[X_1^1 \neq X_1^2]$. Arc functions in this example have all only one term which is a tuple of basic functions: projection X_i^j or diffusion/synchronization S_{C_i}, $S_{C_{i,j}}$ (which is a constant function returning a whole color class or static subclass). The tuple notation should be interpreted as Cartesian product of the tuple elements: hence the evaluation of a function tuple is the Cartesian product of the evaluation of its elements. The domain of an arc function is the color domain of the transition connected to the arc, its codomain is $Bag[\mathcal{C}(p)]$ where p is the place connected to the arc. The arc function $\langle X_1^1, X_1^2, S_{C_{2,1}} \rangle$ on the output arc from TxMsg evaluated on the binding $\langle X_1^1 = s_1, X_1^2 = s_2 \rangle$ returns the set $\{\langle s_1, s_2, d_1 \rangle + \ldots + \langle s_1, s_2, d_h \rangle\}$ $(= \{s_1\} \times \{s_3\} \times C_{2,1})$. SN arc functions and guards syntax is formally defined in Sec. 2.2.

Any SN model can be *unfolded* into an equivalent PN model, it is thus possible to apply any existing PN analysis algorithm to the unfolded model, however this approach has three drawbacks: (1) the unfolded model may be huge, (2) the unfolding requires to fix the color class size, hence it is inherently less parametric, and (3) the unfolded model does not allow to exploit the symmetries possibly present in the model to improve the analysis efficiency.

The dynamic behavior of a SN model can be expressed through its Reachability Graph (RG): the latter can be finite or infinite. Starting from the initial marking the RG can be obtained by following the possible execution paths that originate from the initial marking m_0: the set of transition instances that are enabled in m_0 is first established, then one of them is fired (according to some selection criteria) leading to a new marking. The set of enabled transition instances in any marking and the effect of firing a given enabled transition instance (i.e. the marking reached after such firing) depend on the marking of the input and inhibitor places and on the functions appearing on all arcs connected to the transitions. For instance, assuming that initially place Sites contains the set C_1 while the others are empty, there are $|C_1|(|C_1|-1)$ instances of TxMsg enabled in the initial marking, with binding $X_1^1 = s_i, X_1^2 = s_j, \forall i, j \in 1, \ldots, m, i \neq j$. After the firing of one such instance, say (TxMsg,$\langle X_1^1 = s_1, X_1^2 = s_3\rangle$), one token with color s_1 is withdrawn from place Sites (due to function $\langle X_1^1\rangle$ on the arc) and $|C_2| - 1$ tokens with colors $\langle s_1, s_3, d_1\rangle, \ldots, \langle s_1, s_3, d_h\rangle$ are added in place T-buffer. Observe that in the new marking m_1 only a subset of the instances of TxMsg initially enabled are still enabled: in fact all the instances with X_1^1 bound to s_1 are no more enabled because they were in (effective) conflict with the instance that fired in m_0. In the new marking, $(|C_1| - 1)(|C_1| - 1)$ instances of TxMsg, and $|C_{2,1}|$ instances of RxMsg are enabled, with binding $X_1^1 = s_1, X_1^2 = s_3, X_2^1 = d_j$ one for each token $\langle s_1, s_3, d_j\rangle$ in T-buffer. If all the instances of RxMsg enabled in m_1 fired, one after the other, then the newly reached marking m_i would have no more tokens in T-buffer, and the same set of tokens initially in T-buffer would now be in R-buffer. In marking m_i one instance of TxAck is enabled, with binding $X_1^1 = s_1, X_1^2 = s_3$: observe that the arc function $\langle X_1^1, X_1^2, S_{C_{2,1}}\rangle$ appearing on the transition input arc implements a synchronization of all the messages initially sent out by transition instance (TxMsg,$\langle X_1^1 = s_1, X_1^2 = s_3\rangle$) to reply with an acknowledge message to the site who initiated the communication. Observe that the instance of RxMsg moving the last token (message) from T-buffer to R-Buffer is in (effective) *causal connection* with TxAck (i.e. it causes one instance of TxAck to become newly enabled): also for this relation there exists a structural necessary condition. The a-priori knowledge of the structural conflict and causal connection relations may be fruitfully exploited to speed up the RG generation or the simulation of the SN model, or to support the modeler in the verification of model consistency (e.g. by checking certain marking invariants), or in the complete specification of the some model parameters (e.g. transition rates or weights and priorities, when the model is used to generate a stochastic process: this extension is known under the name of Stochastic Symmetric Nets SSN). The calculus proposed in the rest of this paper has several possible applications, among which the computation of the above mentioned structural relations in a concise and parametric form.

Table 1: **SN NOTATION**

$\mathcal{N} = \{P, T, \Sigma, \mathcal{C}, \Phi, W^-, W^+, W^h, m_0\}$	A symmetric net
T	A finite set of *transitions*;
P	A finite set of *places*;
$T \cup P$ $(P \cap T = \emptyset)$	The set of the nodes of the net;
$C_1, C_2, \ldots, C_n$	Finite sets of *colors* called *basic color classes*;
$C_{i,j}$, $i \in \{1 \ldots \|\|C_i\|\|\}$	When a basic color class C_i is partitioned into static subclasses, $C_{i,j}$ is its j-th subclass;
$\|\|C_i\|\|$	Number of static subclasses in which C_i is partitioned
$\Sigma = \{C_1, C_2, \ldots, C_n\}$	The set of the basic color classes;
$C_1^{e_1} \times C_2^{e_2} \times \ldots C_n^{e_n}$	A color domain: e_i is the multiplicity of color class C_i in the domain.
$\mathcal{C}(s)$ $\forall s \in T \cup P$	*Color domain* of node s. Shortly it will be referred as "the domain of s"
Φ	Transition predicates (constraints on a transition color domain)
$Bag[D]$	Set of all multisets defined on set D
$m \in Bag[\mathcal{C}(p)]$	A multisets on the color domain of place p denoted with m stands for a place marking.
$W^-(t,p), W^+(t,p), W^h(t,p)$	Functions labeling respectively the input, output and inhibitor arcs of a given transition $t \in T$; they have domain $\mathcal{C}(t)$ and codomain $Bag[\mathcal{C}(p)]$, hence for any given binding c of t they return a multiset in $Bag[\mathcal{C}(p)]$.
$X_i^j, S_{C_i}, S_{C_{i,j}}, !X_i^j$	SN basic functions: it is a limited set of functions defined on a color domain of a transition t with values into a basic color class, $\mathcal{C}(t) \rightarrow Bag[C_i]$
$X_i^j = X_i^k, X_i^j \neq X_i^k$ $d(X_i^j) = d(X_i^k)$ $d(X_i^j) \neq d(X_i^k)$ $d(X_i^j) = C_{i,k}, d(X_i^j) \neq C_{i,k}$	Basic predicates: simple assertions built on SN basic functions. They test the (dis)equality of transition variable binding or the inclusion of the value bound to a given variable in a static subclasses($d(X_i^j)$ denotes the static subclass of the value bound to variable X_i^j).
$\Phi(t)$	a transition predicate is a boolean expression whose terms are basic predicates.
$[p]$	A guard: p is a boolean expression whose terms are basic predicates. A guard $[p]$ is function that maps each color $c \in \mathcal{C}(t)$ in $\{c\}$ if $p(c)$ evaluates to true, in $\emptyset$ otherwise.

2.2. SN arc functions formal definition

The SN formalism defines a precise syntax to express arc functions W^-, W^+ and W^h. A SN function W labeling an (input, output or inhibitor) arc connecting transition t and place p, is a mapping $W : \mathcal{C}(t) \rightarrow Bag[\mathcal{C}(p)]$ whose form is:

$$W = \sum_i \lambda_i.T_i[p_i], \quad \lambda_i \in \mathbb{N}$$

In words, we say that W is a weighted sum of guarded *function tuples*. Function tuple domain and codomain (consistently with the definition of W) are $T_i : \mathcal{C}(t) \rightarrow Bag[\mathcal{C}(p)]$. The sum is a multiset sum and λ_i are scalars. Guard $[p_i]$ associated with T_i has the following interpretation: $T_i[p_i](c)$ is equal to $T_i(c)$ if $p_i(c) = true$, otherwise it is equal to $\emptyset$. Both the predicates and the elements of the function tuples have a precise syntax which is discussed next.

Syntax of function-tuples. A function-tuple T, denoted by $\langle f_1, \ldots, f_k \rangle$, corresponds to the Cartesian product of functions f_i called class-functions. Each class-function f_i maps the elements of $\mathcal{C}(t)$ into the elements of $Bag[C_j]$ for some $j \in \{1, \ldots, n\}$. The angular brackets indicate the Cartesian product of the elements of the tuple thus the application of tuple T to color $c \in \mathcal{C}(t)$ results in $T(c) = \otimes_{i=1}^{k} f_i(c)$ where $\otimes$ is the multiset Cartesian product operator. Thus $T_i(c) \in Bag[\mathcal{C}(p)]$

The components f_i of a tuple T are linear combinations with coefficients in $\mathbb{Z}$ of *basic* functions, namely:

$$f_i = \sum_{k=1}^{e_j} \alpha_k.X_j^k + \sum_{q=1}^{||C_j||} \beta_q.S_{C_{j,q}} + \sum_{k=1}^{e_j} \gamma_k.!X_j^k, \quad \alpha_k, \beta_k, \gamma_k \in \mathbb{Z} \tag{1}$$

where $X_j^k, !X_j^k, S_{C_j}, S_{C_{j,k}}$ are the basic functions and represent the limited set of symbols upon which class-functions are defined. The basic functions have the following semantics:

X_j^k	Is called *projection*: it selects the value associated with the k-th variable of type C_j in the transition binding;
$!X_j^k$	Is called *successor*: if class C_j is circularly ordered it selects the successor of the value associated with the k-th variable of type C_j in the transition binding;
$S_j, S_{j,k}$	Are constant functions called *diffusion/synchronization*: they maps respectively into the colorset C_j and $C_{j,k}$. Observe that the following equivalence holds: $S_{C_j} = \sum_k S_{C_{j,k}}$.

In (1) scalars must be such that no negative coefficient result from the evaluation of f_i for any color (consistent with the guard possibly associated with the function-tuple or transition).

In general, if class C_j is ordered, then it cannot be partitioned in subclasses. This restriction could be relaxed for ordered color classes only when each element in the class belongs to a different static subclass, i.e. $\forall k \, |C_{j,k}| = 1$.

Guards. Guards in SSN models are boolean expressions whose terms are *basic predicates*: the set of basic predicates is summarized in Tab. 1. The basic predicates allow to check whether two variables are associated with the same value in a given binding, or in case of ordered classes if they are associated with values that can be related through the successor function; it is also possible to check if the value associated with a variable belongs to a given static subclass or if two variables have values in the same or different static subclass. There is no possibility to explicitly refer to a specific color (unless it is the only element of a static subclass).

Guards can be applied to transitions or to function tuples within arc functions. When applied to transitions the effect of a guard is to restrict color domain $\mathcal{C}(t)$. When applied to a function tuple, the effect of a guard is to make it evalutate to the empty set when the guard is false. If for a given transition binding all function tuples composing a given arc expression are annulled by their guards, the effect is equivalent to deleting the arc.

Table 1 summarizes the SN notation used througout the paper.

2.3. The language for the symbolic calculus

This section formally introduces the syntax of the expressions on which the calculus is based.

They are very similar to the SN arc functions introduced in the previous section but with some variations and extensions, such as the introduction of the intersection operator and of the filters. Before presenting the calculus expressions let us introduce the linear extension of SN functions and a different interpretation of guards associated with function tuples.

Linear extension of SN functions. Let $F : \mathcal{C}(t) \to Bag[D]$ be a SN function tuple, class function or elementary function, then $F^* : Bag[\mathcal{C}(t)] \to Bag[D]$ is defined in the following way: $F^*(a + b) = F(a) + F(b),\ F^*(\lambda.a) = \lambda.F(a),\ F^*(0) = 0$ for each $a, b \in \mathcal{C}(t)$, $\lambda \in \mathbb{N}$ and where 0 is the empty multiset. Abusing notation in the remainder of the paper we shall use the same symbols to denote SN functions and their linear extensions. When needed we use the term linear function to make it clear that we are using the linear extension of the same function.

Interpreting SN guards as functions. Let D be a color domain, a guard can be interpreted as a function $[p] : Bag[D] \to Bag[D]$ such that $[p](S) = \{c \in S : p(c) = true\},\ \forall S \in Bag[D]$ and $[p](\emptyset) = \emptyset$. The expression $T[p]$, used in SN to express a guarded function tuple can thus be reinterpreted as composition of two functions: $T \circ [p]$, where $T : Bag[D] \to Bag[D']$.
Extension with filters. Exploiting the above interpretation of guards, it is possible to extend the SN arc functions by allowing the application of a guard after the application of the function tuple (left composition of a guard): $[p] \circ T$. A guard left composed with a function tuple is called *filter*.

Composition operator $\circ$ between a tuple-function and its guard or filter will be hereafter omitted except in those situations where we want to stress the actual computation of such composition under the framework of the calculus.

We are now ready to introduce the language of the calculus expressions: it is closed with respect to a number of operators that will be introduced later on, and upon which the algorithms for the computation of SN structural properties are based.

Let f and g be two class-functions with same domain $Bag[D]$ and with codomain $Bag[C_i]$. Assume $\mathbf{m} \in Bag[D]$ and $\mathbf{m} = \sum_j \lambda_j.c_j$. The function intersection $f \cap g$ is a linear function from $Bag[D]$ to $Bag[C_i]$ so defined $f \cap g(\mathbf{m}) = \sum_j \lambda_j.f{\cap}g(c_j) = \sum_j \lambda_j.f(c_j){\cap}g(c_j),\ \forall \mathbf{m} \in Bag[D]$, where the $\cap$ on the right side is the multiset intersection.

Definition 2.1. (Language $\mathcal{L}$)
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;
- $\mathcal{S} = \left\{X_i^j, S_i, S{-}X_i^j, S_{i,k}, !^m X_i^j, S{-}!^m X_i^j\right\}, i : 1, \ldots, n,\ j : 1, \ldots, e_i,\ k : 1, \ldots, ||C_i||,\ m \in \mathbb{Z};$ is the set of *elementary* (class) functions; these are functions $Bag(D) \rightarrow Bag[C_i]$, and they correspond to the linear extension of the corresponding (class) functions used in SN expressions, with the following semantics for the $m - th$ successor/predecessor functions (reserved to ordered classes): if $m = 0$, $!^m X_i^j = X_i^j$, if $m > 0$, $!^m X_i^j$ is equivalent to the application of the successor function $m\ mod\ |C_i|$ times, finally if $m < 0$, $!^m X_i^j$ is equivalent to the application of the successor function $(m + K|C_i|)$ times, where K is the smallest natural number such that $(m + K|C_i|) > 0$.
- $T_j = \langle f_1, \ldots f_l \rangle\ Bag[D] \rightarrow Bag[D']$, and $\forall r \in \{1, \ldots, l\}$, $f_r = \sum_j \Gamma_{i,j}$ where $i \in \{1, \ldots, n\}$ and $\Gamma_{i,j}$ is any intersection of symbols in $\mathcal{S}$ with the same class subscript i;
- $[g'_j]$ and $[g_j]$ be SN guards on D' and D, respectively.

The set of expressions:

$$\mathcal{L} = \left\{E : E = \sum_j [g'_j]\lambda_j T_j[g_j]\right\},\ \lambda_j \in \mathbb{N}$$

is the language used to express SN structural relations.

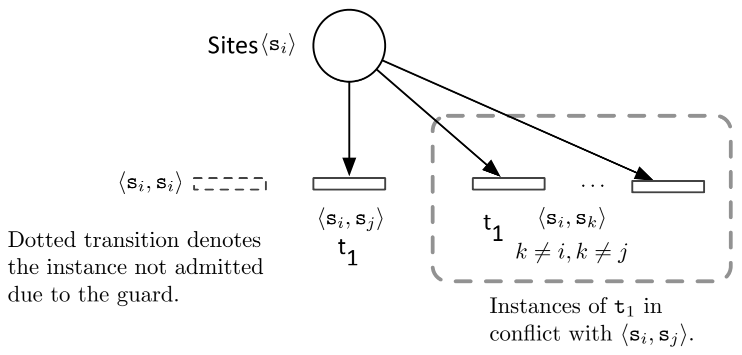
It can be proven that any SN arc-function can be re-written using the syntax of the expressions provided by the above definition: let us illustrate the correspondence on a few examples (the expressions on the left can be used in SN arc functions but are not valid expressions in $\mathcal{L}$):
$\langle 2.S_1 - X_1^1 - X_1^2 \rangle \equiv \langle (S{-}X_1^1) + (S{-}X_1^2) \rangle$;
$\langle S_{i,q} - X_i^j \rangle [d(X_i^j) = C_{i,q}] = \langle (S_i - X_i^j) \cap S_{i,q} \rangle [d(X_i^j) = C_{i,q}]$;
$\langle S_1 - X_1^1 - X_1^2 \rangle [X_1^1 \neq X_1^2] \equiv \langle (S{-}X_1^1) \cap (S{-}X_1^2) \rangle [X_1^1 \neq X_1^2]$.

Language $\mathcal{L}$ without filters is equivalent to that used to express SN arc functions, and the equivalence depends on the restriction introduced in the formal definition of SN functions that requires no negative coefficients appearing in a SN function evaluation for any color: indeed in the second and third example above the guard was needed to satisfy such restriction. The possibility of using filters instead makes language $\mathcal{L}$ strictly more powerful than that of SN arc functions: for instance the expression $[X_1^1 \neq X_1^2] < S, S >$ cannot be expressed in this compact form through the SN syntax.

Hereafter we shall use the term *complement* to denote elementary function $S - X_i^j$. In some cases, to keep notation simpler, we shall use the *extended complement* expression $S_1 - X_1^1 - X_1^2$ as a shorthand notation for an intersection of complements $(S{-}X_1^1) \cap (S{-}X_1^2)$.

2.4. Expressing structural properties in SN

Some properties of PN models can be derived directly from the net structure. The corresponding analysis techniques are indicated with the term *structural analysis* to stress the fact that they are purely based on the net structure, and do not depend on the dynamic evolution of the model through the reachable markings. Examples of structural analysis techniques are: P and T-invariants derivation or verification


Sites$\langle s_i \rangle$
$\langle s_i, s_i \rangle$
$\langle s_i, s_j \rangle$
t1
t1
$\langle s_i, s_k \rangle$
$k \neq i, k \neq j$
...
Dotted transition denotes the instance not admitted due to the guard.
Instances of t_1 in conflict with $\langle s_i, s_j \rangle$.

DA RIFARE - PRESA DA esempi - rinominare t1 con TxMsg

Figure 2: unfolding of transition TxMsg with respect to input place Sites

and computation of relations between nodes of the net (e.g. structural causal connection, structural conflict or structural mutual exclusion between transition pairs).

In this section the general definition of structural relations in High Level Petri Net models, extending the corresponding (basic) relations in PN models, is provided. They correspond to a *folded* representation of the *basic* structural relations on the underlying unfolded PN model.

Definition 1. [Symbolic relation] Given a binary relation $\mathcal{R}$ between the color instances of nodes s and s' of a CPN model defined as $\mathcal{R} \subseteq (s \times \mathcal{C}(s)) \times (s' \times \mathcal{C}(s'))$, its symbolic representation denoted $\mathcal{R}(s, s')$ is a mapping from $\mathcal{C}(s')$ to $2^{\mathcal{C}(s)}$ such that $\mathcal{R}(s, s')(c') = \{c : (s, c)\mathcal{R}(s', c')\}$ for each $c' \in \mathcal{C}(s')$.

It is interesting to observe that the structural relations of any SN model can be expressed in a symbolic form similar to that used to specify the SN arc expressions, presented in Sec. 2.1. To clarify the above statement let us consider an example of symbolic structural conflict between a pair of colored transitions of a SN model, and explain how it relates to the *basic* structural relation between instances of the same two transitions in the underlying unfolded model.

In the SN model in Fig. 1 let us consider the structural conflict between different instances of the colored transition TxMsg with respect to place Sites. Fig. 2 shows the unfolded subnet corresponding to transition TxMsg and place Sites. If we pick any instance (TxMsg, $\langle X_1^1 = t, X_1^2 = r \rangle$), $\langle t, r \rangle \in C_1 \times C_1$ it is clear that it might be disabled by any (TxMsg, $\langle X_1^1 = t, X_1^2 = r' \rangle$) with $r' \neq t$ and $r' \neq r$, assuming that $|C_1| \geq 3$ (there are at least three sites in the system). We could thus say that for any $t, r \in C_1, t \neq r$ it holds true

$$(\mathsf{TxMsg}, \langle X_1^1 = t, X_1^2 = r' \rangle)\ SC\ (\mathsf{TxMsg}, \langle X_1^1 = t, X_1^2 = r \rangle),\ \ \forall r' : r' \neq t \wedge r' \neq r.$$

In the actual model this can be interpreted as follows: in each cycle a sender transmits the message to only one among the possible receivers, hence all instances are in mutual conflict. It is possible to express such statement in a compact and symbolic form using the SN arc functions syntax. The symbolic structural relation is denoted $SC($TxMsg, TxMsg and it is possible to show that it can be expressed as a function with the following syntax:

$$SC(\mathsf{TxMsg}, \mathsf{TxMsg}) = \begin{cases} \langle X_1^1, S - X_1^1 - X_1^2 \rangle [X_1^1 \neq X_1^2] & |C_1| \geq 3 \\ \langle 0, 0 \rangle & |C_1| = 2 \end{cases}$$

Indeed, given any instance of TxMsg: $\langle t, r \rangle \in C_1 \times C_1$, the function evaluation gives $SC(\mathsf{TxMsg}, \mathsf{TxMsg})(\langle t, r \rangle = \langle t, C_1 - t - r \rangle [t \neq r]$ which is equivalent to the expression given on the unfolded model (it is equivalent in the sense that it identifies the same set of conflicting instances).

Main structural relations. The following table collects several of the most common SSN symbolic structural relations with their meanings.

$SbT(p, t)$	$SubtractedbyTransition$: provides the set of tokens an instance of t withdraws from p
$SfP(t, p)$	$SubtractedfromPlace$: given a color of p it provides the color instances of t that withdraw it
$AbT(p, t)$	$AddedbyTransition$: provides the set of tokens an instance of t produces in p
$AtP(t, p)$	$AddedtoPlace$: given a color of p it provides the color instances of t that produce it in p
$SC(t, t')$	Structural Conflict between transition instances
$SCC(t, t')$	Structural Causal Connections between instances of t and t'
$simpleSME(t, t')$	Structural Mutual Exclusion

In the next sections it is shown how functional expressions for structural symbolic relations, as of the type illustrated in the example and those in the table, can be obtained operating a given calculus directly to the SSN arc functions. Before this, however several extensions and algebraic characterizations must be done to the SSN definition of functions.

Computing structural relations: the operators of the calculus. Table 2 shows how to compute the structural properties introduced in previous subsection "**Main structural relations**". As the formulas show, structural symbolic relations are obtained from the arc functions by doing several calculations. The following table summarizes the involved operators. In the list, symbols f and g are functions on multisets.

$\overline{f}$	*Support* operator: $\overline{f}$ is such that $\overline{f}(c) = \{d \in f(c)\}_{set},\ \forall c$
$f - g$	*Difference* operator: $f - g$ is such that $f - g(c) = f(c) - g(c)\ \forall c$, where $-$ on the right side is the difference between multisets
f^t	*Transpose* operator: $f^t(c)(d) = f(d)(c)$
$\overline{f} \circ \overline{g}$	*Composition* operator: it is the classical composition operator between functions.
$f \cup g$	*Union* operator :where $\cup$ on the right side is the union between multisets
$f \cap g$	*Intersection* operator: where $\cap$ on the right side is the intersection between multisets
$\neg g$	*Complement* operator
$!^k f$	*Successor* operator

Several of the listed operators, namely the support, the difference, and the transpose, have been discussed in [3]. In this report we extensilvely discuss the composition operator which is fundamental for the effective calculation of all structural relations.

Table 2: **Computing structural properties in SSN.**

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

Table 3: **Main Notations of the Calculus**

$\mathcal{L}$	The language used to represents structural relations in SN (see Def. 2.1);
l, l_i	elements of language $\mathcal{L}$;
$\Omega = \{+, -, .^t, \cap, \bar{.}, \circ\}$	The operators on language $\mathcal{L}$;
$\{\mathcal{L}, \Omega\}$	The set of the expressions writable on language $\mathcal{L}$ applying the operators in Ω;
$\emptyset_{D \to D'}$	the empty function on domain D and codomain D';
$S_{D \to D'}$	the universe function on domain D and codomain D';
e, e', e_i	elements of $\{\mathcal{L}, \Omega\}$;
$T, T_i, T', \ldots$	Function tuples;
f, f_i, g, g_i	Class-functions, if not otherwise stated;
p, p_i	Predicates in filters and guards, if not otherwise stated;
$f \cap \ldots$	Intersection form involving class-function f;
$p \ldots$	Conjunctive form involving predicate p;
$\langle \ldots, f, \ldots \rangle$	Tuple having class-function f as component;
$\langle \ldots, f_i, \ldots \rangle$	Tuple whose ith component is class-function f_i;
$\langle f_i \rangle_{\otimes_{i:1}^m}, \langle f_i \rangle_{\otimes_i} \;\; i = 1, \ldots, m$	m-tuple of class-functions that are indexed on their position in the tuple, index i can be omitted in some circumstances ;
$\langle f \rangle_{\otimes^m}$	m-tuple formed exclusively by class-function f;
$Symb(f)$	$\{!^h X^i\}$, $!^h X^i$ occurs on f ($X^i \equiv !^0 X^i$);
$Var(g)$	$\{X^i\}$, $!^h X^i \in Symb(g)$;
$Idx(p)$	$\{i\}$, $X^i \in Var(p)$

3. The calculus as a parametric rewriting system

The computation of SN's structural relations as illustrated in Tab. 2 has as consequence the necessity to solve expressions $e \in \{\mathcal{L}, \Omega\}$ that is expressions whose terms are in $\mathcal{L}$ and that may involve operators in Ω. The calculus aims at transforming expressions e in order to remove all the operators and obtain in the end an element $l \in \mathcal{L}$. It is step by step driven by the application of given rewriting rules depending on the form of the expression e_i at step i^{th} of the transformation

$$e \overset{(0)}{\rightarrow} e_1 \overset{(1)}{\rightarrow} \ldots \rightarrow e_i \overset{(i)}{\rightarrow} \ldots \rightarrow l$$

The algorithm running the calculus doesn't lead to a *normal form* for l. This last aspect is clarified in the following definitions.

Definition 3.1. Let us assume $l \in \mathcal{L}$ and $l' \in \mathcal{L}$, then we say that l and l' are equivalent, writing

$$l \equiv l'$$

iff l and l' are defined on the same domain D and codomain $D'$and $l(c) = l'(c),\ \forall c \in D$.

$\equiv$ is an *equivalence relation* on $\mathcal{L}$ and $(\mathcal{L}, \equiv)$ is the partitioning it induces.

If $l \in \mathcal{L}$, then $[l] \in (\mathcal{L}, \equiv)$ is the equivalence class of l, that is $[l] = \{l' \in \mathcal{L} : l \equiv l'\}$. We will use $r_i \in \mathcal{L}$ as representative to denote the generic element $[r_i] \in (\mathcal{L}, \equiv)$. We introduce a notation to to denote representatives for the *empty* and *universe* functions for given domain D and codomain D', namely $[\emptyset_{D\to D'}]$ will be used to denote the class of elements of $\mathcal{L}$ mapping into the empty set while $[S_{D\to D'}]$ will be used to denote the class of elements of $\mathcal{L}$ mapping each element of D into D'.

Relation $\equiv$ can be extended (observe that $\mathcal{L} \subset \{\mathcal{L}, \Omega\}$) on set $\{\mathcal{L}, \Omega\}$ of expressions inducing similarly the partitioning $(\{\mathcal{L}, \Omega\}, \equiv)$. Let $\omega \in \Omega$ be an operator defined on $\mathcal{L}$, then given two elements of the language $l_1 \in \mathcal{L}$ and $l_2 \in \mathcal{L}$ **resolving the operator** in $e : l_1\ \omega\ l_2$ means finding $l \in \mathcal{L}$ such that $l \equiv l_1\ \omega\ l_2$, that is $[l] = [l_1\ \omega\ l_2]$ where both $[l]$ and $[l_1\ \omega\ l_2]$ are elements of $(\{\mathcal{L}, \Omega\}, \equiv)$. The solution l may not be unique and it depends on the sequence of rewriting rules used in the intermediate transformations.

The processing actuated to **resolve** an operator proceeds by steps, at each step an expression containing an operator is syntactically rewritten according to its algebraic properties in order to arrive in a finite time at a final expression in which no operators appear (except $+$). Each basic algebraic transformation is denoted as a *rewriting rule* and is expressed in the following terms $l_1\ \omega\ l_2 \rightarrow e$ where e is either an element of $\mathcal{L}$ or a calculus *expression* that still contains **unresolved** operators. A necessary condition to guarantee the algorithm termination is that if an expression e encountered in the processing is such that if $[e] = [\emptyset]$ or $[e] = [S]$ then eventually, after some rewriting step, they become respectively $e \rightarrow \emptyset_{D\to D'}$ or $e \rightarrow S_{D\to D'}$ (observe that due to filters and guards $e \neq \emptyset_{D\to D'}$ and $[e] = [\emptyset_{D\to D'}]$ may hold).

Each rewriting rule definition will be univocally identified by a number on the right side of the definition. The application of a rule in a calculus will be denoted by using its identifier between square brackets in the rewriting step:

$$e \xrightarrow{[n]} e'$$

Rewriting rules that are related will be collected in rule-sets. Use of rule-sets in a calculus skips over the details of the intermediate expressions. Applications of rule-sets will be denoted using their name in the rewriting step:

$$e \xrightarrow{[RS]} e' \qquad .$$

Since the calculus is parametric in the color class cardinalities the right side of a rewriting rule definition can be composed by several alternatives according to the values of the parameters. In this case each alternative will be associated with a constraint on the range of the parameters representing the cardinality of basic classes. Hereafter when the semantics of an expression depends on the basic classes size we shall explicitly indicate the constraint.

Hereafter two particular forms for tuples and predicates are defined, required to apply some rewriting rules defined in the following section.

A characterization of the tuples

In this section a specific form of expression equivalence class representative is introduced: this leads to a fundamental characterization of tuples form that will be used later on to define some expression rewriting rules to be used in the solution process.

Let us consider the following example of function $l \in \mathcal{L}$: $\langle S - X_1^1 - X_1^2 \rangle$ This tuple can be rewritten as: $\langle S - X_1^1 - X_1^2 \rangle[X_1^1 \neq X_1^2] + \langle S - X_1^1 \rangle[X_1^1 = X_1^2]$. The two guarded terms in the last expression are such that for any value of their arguments satisfying the associated guard they map on a constant size set. In the example the the first function $S - X_1^1 - X_1^2$ maps on a multiset of size $|C_1 - 2|$ while the second function $S - X_1^1$ maps on a multiset of size $|C_1 - 1|$, for any value of X_1^1 and X_1^2 satisfying their guard.

For any equivalence class $[l]$ it is always possible to find a representative with constant size terms.

Definition 3.2. (constant size function)
Function f with values in Bag[D] is constant size iff $f(d) \neq \emptyset$ implies $|f(d)| = n$, where n is a natural number indicating the size (or cardinality) of the function.

Let us consider a class function f_i composing a tuple $T[g] \in \mathcal{L}$: it is constant size if $f_i(c') = n, \forall c' \in \{c : g(c)\}$. Let us state a sufficient condition on the form of a class function ensuring that it is constant size.

Property 1. Let us consider a class function f_i appearing in a guarded tuple $T[g]$: A syntactical sufficient condition for f_i being constant size is: either f_i belongs to $\mathcal{S}$ (Def. 2.1) or it has one of the following forms

$$a) \quad \bigcap_{q,j} S - !^q X_i^j \qquad b) \quad S_{i,k} \bigcap_j S - X_i^j$$

where in a) and b) for each pair $!^{q_1} X_i^{j_1}, !^{q_2} X_i^{j_2}$ s.t. $j_1 \neq j_2$: $g \Rightarrow !^{q_1} X_i^{j_1} \neq !^{q_2} X_i^{j_2}$, additionally in b) for each X_i^j: $g \Rightarrow d(X_i^j) = C_{i,k}$.

Of course a tuple composed of constant size class functions is also constant size. Proving the property is a just a technical matter. Rewriting rules exist allowing to transform any tuple into a constant size tuples: they are formalized in the Appendix. Let us just give the general idea through an example. Consider the tuple $\langle S - X_1^1 - X_1^2, S_{1,2} - X_1^1 \rangle \equiv \langle S - X_1^1 \cap S - X_1^2, X_1^1 \cap S_{1,2} \rangle$. In order to obtain an equivalent constant size form we just have to discriminate whether X_1^1, X_1^2 are assigned the same element or not, and represent the second intersection through a membership clause:

$$\langle S - X_1^1 \cap S - X_1^2, X_1^1 \rangle [X_1^1 \neq X_1^2 \wedge d(X_1^1) = C_{1,2}] + \langle S - X_1^1, X_1^1 \rangle [X_1^1 = X_1^2 \wedge d(X_1^1) = C_{1,2}]$$

Our calculus is parametric in the basic classes cardinaly, e.g. in the first example $|C_1| = n_1$: in this case n in Def. 3.2 is parametric. The form of the representative expression may change depending on the parameter value. In the first example, assuming $|C_1| = n_1 \geq 2$ there are two constant size forms, one for $n_1 = 2$ and another one for $n_1 \geq 2$:

$$F = \begin{cases} \langle S - X_1^1 \rangle [X_1^1 = X_1^2] & n_1 = 2 \\ \langle S - X_1^1 - X_1^2 \rangle [X_1^1 \neq X_1^2] + \langle S - X_1^1 \rangle [X_1^1 = X_1^2] & n_1 > 2 \end{cases}$$

A characterization of predicates

In this section a specific form for the predicates used as guards or filters in the language expression: this is a canonical form. Unless otherwise indicated, we focus on non trivial filters (i.e, filters not equivalent to $true$ or $false$) meeting the form below, hereinafter referred to as *canonical*

- filters are conjunctions of SN elementary guards
- (in)equalities take the form $X_i^j = (\neq)!^k X_i^h$, with $j < h$
- if $X_i^j =!^r X_i^h$ then X_i^h is not involved in any other equality and $\nexists w$, $X_i^w =!^k X_i^j$
- if $X_i^j \neq!^r X_i^h$ then $\nexists w$, $X_i^w =!^k X_i^h$ or $X_i^w =!^k X_i^j$

A filter's canonical form results from replacing X_i^h with $!^{-k} X_i^j$, for any equality $X_i^j =!^k X_i^h$, recursively. Here is an example of canonical rewriting of a filter.

$$[X^1 \neq X^2 \wedge !^{-1} X^1 = X^4 \wedge X^4 \neq X^3] \quad \longrightarrow [X^1 \neq X^2 \wedge X^1 =! X^4 \wedge X^1 \neq! X^3]$$

The rewriting rules to be applied for transforming any predicate to its canonical form are summarized in the Appendix, ruleset$[E]$.

4. Top-down rules to solve the composition of language expressions

This section describes the application of composition between elements of language $\mathcal{L}$, introduced in Def. 2.1. Some rewriting rules based on the algebraic properties of functions shall be introduced, allowing to solve the following problem: $l_1 \circ l_2 \overset{\cdots}{\longrightarrow} l$, where $l_1, l_2, l \in \mathcal{L}$.

4.1. Starting rules

The first rule is based on distribution of composition over union. In formulae:

$$l \circ l' = \textstyle\sum_i [.] \lambda_i T_i [.] \circ \sum_j [.] \lambda_j T'_j [.] \to \sum_{i,j} \left([.] \lambda_i T_i [.] \circ [.] \lambda_j T_j [.] \right) \quad [1]$$

the correctness of the transformation comes directly from the basic elements of Algebra.

Rule [1] is not final, in that, expression e at the right term of the rule is not an element $l \in \mathcal{L}$. There are two main operators requiring to be resolved, namely the **sum** of **compositions**. The problem requires next to solve in order the following two subproblems:

A. composition between tuples, possibly with guards and/or filters;

B. investigate a suitable and simplified form of union of terms: although the result provided to problem A could be already an element of $\mathcal{L}$, in this step one tries to achieve a convenient form for the result, that in general will be a simplified union of terms satisfying a given *form*.

The rest of this section focuses on the first subproblem (A), that is on the composition of tuples as expressed in the summation at the right side of [1].

Each term in [1] can be rewritten joining the guard of the left-tuple with the filter of the right-tuple by posing the two predicates in *and*. In other words, the following rule (that can be easily verified) holds true:

$$[.]T_i[p_i] \circ [p_j]T_j[.] \rightarrow [.]T_i[p_i \wedge p_j]T_j[.] \qquad [2]$$

The calculus can thus proceed in one of two different directions depending on the syntax of tuple T_i, namely:

1. looking for an equivalent syntactical form of the function $T_i[p_i \wedge p_j]$ of rule [2] in which the inner predicate $[p_i \wedge p_j]$ has been canceled. This problem has already been tackled in the solution of other operators, where the possibility to represent a predicate of a guard within the tuple it has proven useful. Precisely, it was possible in several cases, exploiting the class-function intersection operator. We do not discuss further the validity of such rules here and the reader can find them listed in Appendix under the rule-set $[H]$. In general each rule in set $[H]$ has the form $T[p] \rightarrow T'$. Applying rules from rule-set $[H]$ allows to rewrite the expression on the right of [2] as:

$$[a]T_i[p]T_j[b] \xrightarrow{r} [a]T_i' \circ T_j[b] \ , \ r \in [H]$$

after such rewriting the problem is conducted to that of composing simple tuples. The rules developed to deal with this case are discussed in Sec. 4.2;

2. If rules in set $[H]$ are not applicable because of the form of T_i, then the filter/guard can not be syntactically canceled and different rewriting steps than those in 1. have to be applied to obtain a solution for the problem of composition. These cases are discussed in Sec. 4.3.

4.2. Composition between tuples $T_1 \circ T_2$

If f is a function with domain D, its *extension* to domain 2^D, denoted f^{ext}, is defined in terms of f in the following way:

$$f^{ext} : f^{ext}(\mathbf{d}) \overset{def}{=} \bigcup_{d \in \mathbf{d}} f(d), \ \ \forall \mathbf{d} \in 2^D \qquad \text{(D1)}$$

An operator $*$ on functions with domain D so defined $f * g(d) = f(d) * g(d)$ and $d \in D$, is naturally extended to the functions with domain 2^D in the following way:

$$f^{ext} * g^{ext}(\mathbf{d}) \overset{def}{=} (f * g)^{ext}(\mathbf{d}) \overset{D1}{=} \bigcup_{d \in \mathbf{d}} f * g(d) \overset{def}{=} \bigcup_{d \in \mathbf{d}} f(d) * g(d), \ \ \forall \mathbf{d} \in 2^D \qquad \text{(D2)}$$

The next property states a condition on f^{ext} and g^{ext} that if satisfied allows to solve the above expression at the level of the extended functions:

$$f^{ext} * g^{ext}(\mathbf{d}) = f^{ext}(\mathbf{d}) * g^{ext}(\mathbf{d}) \qquad (2)$$

if $\mathbf{d}$ is a set of one element d then the statement is directly verifiable following Definition D2. In general (2) is not valid, however when the elements $d \in \mathbf{d}$ are structured (as it happens in our context where d is an element of a Cartesian product of basic sets) it can be proved that (2) is valid if f and g depend on disjoint pieces of the element's structure.

Property 2. (Distributivity Condition)
Let D be the Cartesian product $D = 2^{S_1} \times \ldots \times 2^{S_n}$. Let f and g be linear functions defined on domain D and with values in 2^C. Let $*$ an operator defined on the set of linear functions $L[D, C]$ and such that it is distributive with regard to the sum in 2^C. If it exists i such that, denoted with $S = 2^{S_1} \times \ldots \times 2^{S_i}$ and $S' = 2^{S_{i+1}} \times \ldots \times 2^{S_n}$, it holds

i. $f(\mathbf{d} \times \mathbf{d}') = f(\mathbf{d} \times \mathbf{d}'')$ $\forall \mathbf{d} \in S$ and $\forall \mathbf{d}', \mathbf{d}'' \in S'$

ii. $g(\mathbf{d}' \times \mathbf{d}) = g(\mathbf{d}'' \times \mathbf{d})$ $\forall \mathbf{d} \in S'$ and $\forall \mathbf{d}', \mathbf{d}'' \in S$

then

$$f{*}g\ (\mathbf{d}) = f(\mathbf{d}) * g(\mathbf{d}), \ \ \forall \mathbf{d} \in D$$

Proof: Let $\mathbf{d}$ be a Cartesian product of multiset such that $\mathbf{d} = (\mathbf{d_1}, \ldots, \mathbf{d_n}) \in D$ and $\mathbf{d}_i \in 2^{S_i}$, and let it be considered the following computation

$$f{*}g(\mathbf{d})$$

By the linearity of $f * g$ function it holds:

$$f{*}g(\mathbf{d}) = \bigcup_{d \in \mathbf{d}} f{*}g(d) \overset{def}{=} \bigcup_{d \in \mathbf{d}} f(d) * g(d)$$

where $d \in S_1 \times S_2 \times \ldots \times S_n$.

Due to partition of D according to index i, the elements $d \in \mathbf{d}$ are denoted (d', d''). Hence:

$$\bigcup_{d \in \mathbf{d}} f(d) * g(d) = \bigcup_{(d', d'') \in \mathbf{d}} f(d', d'') * g(d', d'')$$

By hypothesis i. given any $\mathbf{d} \in D$ it exist i such that for each pair $(d', d''), (d', d''')$ both belonging to $\mathbf{d}$ it holds $f(d', d'') = f(d', d''')$. Moreover, because $*$ is distributive with regard to $+$, for any given d' it is possible to factorise value $f(d', \cdot) = f(d', d'') \forall d''$. Hence the previous expression becomes:

$$\bigcup_{(d', d'') \in \mathbf{d}} f(d', d'') * g(d', d'') = \bigcup_{d' : (d', d'') \in \mathbf{d}} f(d', \cdot) * \bigcup_{d'' : (d', d'') \in \mathbf{d}} g(d', d'')$$

by hypothesis ii. it holds

$$\bigcup_{d' : (d', d'') \in \mathbf{d}} f(d', \cdot) * \bigcup_{d'' : (d', d'') \in \mathbf{d}} g(d', d'') = \bigcup_{d' : (d', d'') \in \mathbf{d}} f(d', \cdot) * \bigcup_{d'' : (d', d'') \in \mathbf{d}} g(\cdot, d'') = f(\mathbf{d}) * g(\mathbf{d})$$

Observation: Prop. 2 does not apply when the argument $\mathbf{d}$ cannot be expressed as a Cartesian product of sets, for instance the expression $[X_1^1 \neq X_1^2]\langle S - X_1^1, S - X_1^2 \rangle$ is not rewritable using $\mathcal{L}$ as a Cartesian product of class functions mapping on sets, so is any set $\mathbf{d}$ resulting form the aplication of the expression to an element of the domain.

All the functions we will use are actually extensions of underlaying counterparts by Definition D1. Hereafter the superscript *ext* will be omitted.

Tuple composition The following property characterises the simple case of tuple composition.

Property 3. Let F, F' be two tuples, where $F = \langle f_1, f_2, \ldots, f_n \rangle$; if $Var(f_i) \cap Var(f_j) = \emptyset$ for all i, j such that $i \neq j$ then

$$F \circ F' = \langle f_1 \circ F', f_2 \circ F', \ldots, f_n \circ F' \rangle$$

The truth of Prop. 3 follows from Prop. 2, when it is considered $\mathbf{d}$ as $F'(c)$, $*$ as the Cartesian product between functions $\{f_i\}$ of F. Since Cartesian product is ditributive with regard the sum and because the hypothesis $Var(f_i) \cap Var(f_j) = \emptyset$ for all i, j such that $i \neq j$ assures the independency of $\{f_i\}$, the thesis follows.

Property 3 allows to solve the composition between two tuples in terms of more elementary compositions, namely between class-functions and tuples. Property 3 may be generalised:

Property 4. let F be such that:

$$F = \langle \underbrace{f_1, \ldots, f_{n_1},}_{F_1} \ \underbrace{f_{n_1+1}, \ldots, f_{n_2},}_{F_2} \ \ldots, \ \underbrace{f_{n_k+1}, \ldots, f_{n_{k+1}}}_{F_k} \rangle$$

and let us assume $Var(F_i) \cap Var(F_j) = \emptyset$ for each $i, j = 1 \ldots k$ such that $i \neq j$, moreover for each F_i does not exist a subset of function $\{f_{i_j}\}$ belonging to it and such that $Var(\{f_{i_j}\}) \cap Var(F_i \backslash \{f_{i_j}\}) = \emptyset$, then $F \circ F'$ is equal to:

$$F \circ F' = \Big\langle \langle F_1 \rangle \circ F', \ \langle F_2 \rangle \circ F', \ \ldots, \ \langle F_k \rangle \circ F' \Big\rangle$$

It is noticeable that the codomain of $\langle F_i \rangle$ is at most equal to $C_i^{m_i}$.

Concluding the section, to deal with composition two relevant and related aspects should be tackled, in order they are:

1) the ability to symbolically solve $f_i \circ F'$ where f_i is a class-function and F' is a function tuple;

2) the ability to symbolically solve $\langle F_1 \rangle \circ F'$ where tuple $\langle F_1 \rangle$ contains cross variable repetitions.

Next both issues are addressed.

4.2.1. Elementary composition rules $f \circ F$

The calculus solves the composition of class-function f by tuple F utilising a collection of elementary rules. These rules are terminal rules and are based on the algebraic properties of elementary functions. The assumption are:

i- f is an intersection form class-function, different from the empty class-function;

ii- $D = 2^{C_1^{m_1}} \times \ldots \times 2^{C_n^{m_n}}$ is the codomain of F;

iii- $F = \langle f_1^1, \ldots, f_1^{m_1}, f_2^1, \ldots, f_2^{m_2}, \cdots, f_n^1, \ldots, f_n^{m_n} \rangle$;

iv- f_i^k is an elementary class-function for all i, k.

In general, the composition is neither right nor left distributive with respect to the intersection, thus the following rewriting $(f \cap g) \circ F \equiv f \circ F \cap g \circ F$, that would allow us to go one step ahead in solving the operator, can not be applied. However, in expression $f \circ F$ if f is an *intersection of a given subset of SN elementary functions* then F can be distributed on single terms of f.

Indeed in several cases an intersection of SN elementary functions, if simplified, is composed by functions that according to the meaning of Prop. 2 are independent one each other. The following table summarizes the possible forms of f after simplification:

1. $\bigcap_{j\in I} !^{k_j} X_i^j \cap \bigcap_{j\in J} S - !^{k_j} X_i^j$, for all $I, J \subseteq \{1, \ldots, m_i\}$ and $I \cap J = \emptyset$ and $k_j \in \{0, \ldots, |C_i| - 1\}$

2. $\bigcap_{j\in I} !^{k_j} X_i^j \ \cap \ \bigcap_{j\in J} \bigcap_{k\in K_j} S - !^{k} X_i^j$, for all $I \subseteq \{1, \ldots, m_i\}$, $K_j \subseteq \{0, \ldots, |C_i| - 1\}$, $J \subseteq \{1, \ldots, m_i\}$ and $I \cap J = \emptyset$

In case 1. it is possible to distribute F on each term of the intersection because of Prop. 2 since each term maps onto a different component of the tuple argument.

In case 2. it is possible to distribute F on the terms of the outer intersections but not on the terms of the innermost, indeed terms of the innermost intersection are dependent because map on a same component j of the tuple argument. Such forms $\bigcap_{k\in K_j} S - !^k X_i^j$ are discusssed at the end of this subsection.

Table in Rule Set $[A]$ shows the rules utilised to solve $e \circ F$ where e is an elementary function.

Rule Set $[A]$. (Elementary compositions)

[1] $X_i^j \circ F$	[2] $!^k X_i^j \circ F$	[3] $S - X_i^j \circ F$	[4] $S - !^k X_i^j \circ F$	[5] $S_i \circ F$	[6] $S_{i,j} \circ F$
f_i^j	$!^k f_i^j$	S if $\|f_i^j\| > 1$ $S - f_i^j$ if $\|f_i^j\| = 1$	S if $\|f_i^j\| > 1$ $S - !^k f_i^j$ if $\|f_i^j\| = 1$	S_i	$S_{i,j}$

Rules $[A]$.1,$[A]$.5, and $[A]$.6 are terminal rules. Rule $[A]$.2 is not a terminal rule and the rules to compute successor of a SN class-function have to be applied. Rules $[A]$.3 and $[A]$.4 require a further analysis because the related result is dependent on the size of the f_i^j function of F.

Rule set $[A]$ shows that generally the calculus is functional to the size of the class-functions. Calculus keeps track of class-function sizes on a rewriting rules and on the involved expressions basis. Sizes of elementary functions are provided in Tab. 4. When class-function is an intersection-form its size may

Table 4: Size of elementary functions.

X_i^j	$S - X_i^j$	S_i	$S_{i,j}$
1	$\|C_i\| - 1$	$\|C_i\|$	$\|C_{i,j}\|$

range over an interval of values: for instance an expression involving at least two projections on different variables such as $X_i^j \cap X_i^k$ can have size 1 or 0, so $|f_i^j| \leq 1$. The choice to rewrite the expression as $\langle X_i^j \rangle [X_i^j = X_i^k]$, where the class-function has size exactly 1, rather than keep it in intersection form is merely context dependent. Table 5 shows the sizes of intersection forms as computed by the intersection rules.

Table 5: Size of class-functions resulting from intersection rules.

$\bigcap_j X_i^j$	$\bigcap_j S - X_i^j$	$\bigcap_j S{-}!^{k_j} X_i^j$
≤ 1	$\lvert C_i\rvert - 1 \leq \lvert f\rvert \leq \lvert C_i\rvert - v$	$\lvert C_i\rvert - v$

v: number of variables.

Observation: Simplification rules implicitly map class-functions whose size is 0 into the empty class-function, by this reason it is not possible to have an expression representing the empty function. This is a critical requirement which assures that algorithms used to implement the calculus will terminate.

Example 4.1. Let us assume we need to compute the following composition:

$$\langle S - X_1^1 \rangle \circ \langle S - X_1^1 \,\cap\, S - X_1^2 \rangle$$

We observe that depending on the size of class C_1 the result may differ since the size of the intersection-form may be 1 or greater than 1. For instance, if we are assuming $|C_1| = 3$ then the right term of composition may be rewritten as:

$$S - X_1^1 \,\cap\, S - X_1^2 = \underbrace{(S - X_1^1)[X_1^1 = X_1^2]}_{\text{size is 2}} + \underbrace{(S - X_1^1 - X_1^2)[X_1^1 \neq X_1^2]}_{\text{size is 1}}$$

where the size of the leftmost term is greater than 1, namely 2, whereas the size of the rightmost term is 1. Composition with term $S - X_1^1$ may be now done using the rules discussed in the previous sub-section, in particular the composition with the leftmost term results in $S_{C_1}[X_1^1 = X_1^2]$.

Whereas, if we are assuming $|C_1| > 3$ then surely $|S - X_1^1 \cap S - X_1^2| > 1$ and the initial composition $S - X_1^1 \circ (S - X_1^1 \,\cap\, S - X_1^2)$ becomes simply S_{C_1}.

The example shows that the result of compositions of this type must take into account the number of complements and the size of the class on which these are defined in order to determine the size of the function itself. The approach making use of guards to split the intersection-form into terms whose size is 1 or greater than 1, shown in the example, may be generalised to expressions with any number of complements. Let n be the number of complement terms appearing in the intersection-form (which includes complements only) namely $f_i = \bigcap_{j=1}^{n}(S - X_i^j)$ where for simplicity we assumed consecutive indices for projection complements, then the following cases are possible:

1. if $n < |C_i| - 1$ then it holds that $|f_i| > 1$, and the composition with the left term may be solved according to the rules introduced in the previous sub-section;
2. if $n = |C_i| - 1$ then using appropriate guards we may re-write the intersection-form in two disjoint tuples:

   $$\bigcap_{j=1}^{n}(S - X_i^j) \circ [g_1] + \bigcap_{j=1}^{n}(S - X_i^j) \circ [g_2]$$

   where the two guards express the following conditions: $g_1 =$"all projections are different from each other" and $g_2 =$"at least two projections are not different". The first term has size 1 whereas the second term has size greater than 1. Also in this case we may proceed with the composition with the left term according to the rules given in the previous sub-section;

3. if $n > |C_1| - 1$, i.e. $n \geq |C_1|$, then the intersection-form may be re-written using guards in three disjoints terms:

$$\bigcap_{j=1}^{n}(S - X_i^j) \circ [g_1] + \bigcap_{j=1}^{n}(S - X_i^j) \circ [g_2] + \bigcap_{j=1}^{n}(S - X_i^j) \circ [g_3]$$

where the guards are so defined: g_1 ="there are $|C_i|$ projections that differ"; g_2 ="there are exactly $|C_i| - 1$ projections that differ"; g_3 ="there are less than $|C_i| - 1$ projections that differ". Observe that the first term results in the the empty set, since there can not exist a colour of the domain common to all complements. The second term instead is such that its size is equal to 1, and the third one is such that its size is greater than 1. Once this partition has been made the rules given in the previous sub-section may be applied in order to compute the composition.

Intersection forms with dependent terms When, in expression $f \circ F$, function f is an intersection-form class function containing repetitions of successors of a given variable X_i^j then Prop. 2 is not applicable due to the dependence between the terms of the intersection. In this case the composition can not be distributed in order to solve it.

The only not empty intersection-form showing repetitions of variable X_i^j is the intersection of complements, whose form is $f = \bigcap_{k \in I} S - !^k X_i^j$, with $|I| > 1$. The composition of this form with a function $F = \langle \ldots, f_i^j, \ldots \rangle$ can be still syntactically solved according to the next rule. Set I contains whole numbers representing the successor indexes appearing in f. We can assume that (possibly after having split the rewriting process) $max(\{h\}, h \in I) - min(\{h\}, h \in I) < n = |C_i|$ (remember that n may be parametric). If n is fixed then the successor indexes are Natural values $mod|C_i|$: thus if $|C_1| = 7$ then $!^9 X_1^1$ is rewritten as $!^2 X_1^1$.

Let $|f_i^j| = m$ (again, m may be parametric if n is). The formula below expresses the composition result. The first case may include the parametric case, the other case refers to a fixed color class cardinality:

$$f \circ F = \begin{cases} S & m > |I| \\ \sum_{s \in \{0,\ldots,n-1\}/I} !^s f_i^j & elsewhere \end{cases}$$

The first result directly follows from the assumption above, that ensures f (whose cardinality is $n - |I|$) is injective: its linear extension to a set of size at least $|I| + 1$ results in the whole color class. The second result descends from the rewriting of f as $\sum_{s \in \{0,\ldots,n-1\}/I} !^s X_i^j$.

4.2.2. Repetition of a given variable

The previous section discussed cases of composition $F \circ F'$ between tuples where the class-functions of F are pairwise independent. The independence can be stated directly utilising the WN notion of $Var(f_i)$, that is, looking at the set of variables appearing in the class-function f_i. However, repetitions of symbol X_i^j for given values i, j in different class-functions of F introduces a dependence relationships. The more general situation of composition shown in Prop. 4 must thus be addressed. Here the calculus focus on computation $\langle F \rangle \circ F'$ pointed out in such property, precisely F is such that $F = \langle f_1, \ldots, f_{m'_i} \rangle$, F's co-domain is $2^{C_i^{m'_i}}$ for a given i, where $m'_i \leq m_i$ and whose exact value depends by Prop. 2. The problem complexity is faced in two subsequent steps:

1. Tuple F is re-written in a way that composition by F' can be solved with regard to variable X_i^k only, for each value of k;
2. Rewriting rules are provided to solve compositions $F \circ F'$ where F only depends by a single variable X_i^k for a given k.

Step 1. is supported by the following rules which allow to re-write a tuple function F containing intersection-form class-functions into an equivalent intersection of tuple functions each one being dependent by a single variable.

Step-1 rewritings: Simplification of the left-tuple's form.

In order to reduce the left operand of the composition to few basic forms, the left tuple is rewritten.

Consider as example F, in $F \circ F'$, to be $\langle S - X_3^1, X_3^1 + X_3^2\rangle$, then it is equivalent to the following expression $\langle S - X_3^1, X_3^1\rangle + \langle S - X_3^1, X_3^2\rangle$, where '$+$' is the multiset sum. Composition can thus be simplified distributing F' to the addends of the sum (rule [1]): $\langle S - X_3^1, X_3^1\rangle \circ F' + \langle S - X_3^1, X_3^2\rangle \circ F'$. This allows to reduce the form of the left tuple in the composition to those cases in which its class functions are elementary symbols or intersection of them.

The tuples resulting from the previous rule are further rewritten. The target is to syntactically separate those parts of the expression that are independent according to the meaning of Prop. 2. As example consider F to be $\langle S - X_3^1, S - X_3^1 \cap S - X_3^2\rangle$ it can be re-written as $\langle S - X_3^1, S - X_3^1\rangle \cap \langle S, S - X_3^2\rangle$. The nice property of the re-writing just introduced is that the obtained tuples depend on at most one variable each: in the example the first tuple is function of X_3^1 while the second tuple is function of X_3^2. The expression coming out from the application of such property can be composed separately with F' by Prop. 2. The example would become $F \circ F' = \langle S - X_3^1, S - X_3^1\rangle \circ F' \cap \langle S, S - X_3^2\rangle \circ F'$.

The rewriting is formally defined in the following rule [3]:

If f_k, with $k = 1, \dots, n$, maps on sets then:

$$\langle f_1, f_2, \dots, f_n\rangle \rightarrow \langle g_1, g_2, \dots, g_n\rangle \cap \langle h_1, h_2, \dots, h_n\rangle \text{ w.r.t. } X_i^j \qquad [3]$$

where $\forall k = 1 \dots n$:

if $f_k : Var(f_k) = \{X_i^j\}$ then $g_k = f_k, h_k = S_i$;

if $f_k : X_i^j \notin Var(f_k)$ then $g_k = S_i, h_k = f_k$;

if $f_k = e_k \cap l_k$ where $Var(e_k) = \{X_i^j\}$ and $X_i^j \notin Var(l_k)$ then $g_k = e_k, h_k = l_k$.

After the application of rule [3] $Var(\langle g_1, g_2, \dots, g_k\rangle) = \{X_i^j\}$ and $Var(\langle h_1, h_2, \dots, h_k\rangle) = Var(\langle f_1, f_2, \dots, f_k\rangle) - \{X_i^j\}$.

Iteratively applying [3] for each variable X_i^j allows to re-write a tuple function F containing intersection-form class-functions into an equivalent intersection of tuple functions where each tuple depends on an unique and different variable X_i^j. If $v = |Var(\langle f_1, f_2, \dots, f_k\rangle)|$ then the time complexity of [3] is an $O(n)$ and that of the complete algorithm is an $O(vn)$.

The validity of the rewriting follows directly from the properties of intersection and Cartesian product between sets, and by the fact that if variables X_i^j,X_i^k with $j \neq k$ appear in a same class-function of F then they must be in an intersection form.

Due to the application of Rewriting [3], tuples appearing at the final step of composition may show a very few kind of forms limited by the only two possible symbols X_i^j and $S - X_i^j$ for a given j.Table 6 summarizes such basic forms.

Table 6: Basic tuple forms in composition.

A.	$\langle S - !^{s_k} X_i^j \rangle_{\otimes_{k:1}^m}$	tuple full of complements of a given variable
B.	$\langle \ldots, !^s X_i^j, \ldots \rangle$	tuple in which at least one element is a projection
C.	$\langle S - \sum_{s \in I_k} !^s X_i^j \rangle_{\otimes_{k:1}^m}$	tuple full of generalized-complements of a given variable

Step-2 rewritings: The basic forms of the left-tuple.

A. Tuple of complements. The first basic form of tuple F we consider is the one composed by complements:

$$\langle S - !^{s_k} X_i^j \rangle_{\otimes_{k:1}^m} \circ F', \quad \text{where } s_k \in \{0, \ldots, |C_i| - 1\}$$

in this expression the k^{th} component of the left tuple is the complement of the s_k-successor of X_i^j (j being fixed). The operator is solved applying first the following rewriting:

$$\langle S - !^{s_k} X_i^j \rangle_{\otimes_{k:1}^m} \to \langle !^{s_k} X_i^k \rangle_{\otimes_{k:1}^m} \circ \langle S - X_i^j \rangle_{\otimes^m} \qquad [4]$$

we have thus reduced to solve a simpler case of composition where the left tuple consists of the repetition of $S - X_i^j$:

$$\langle !^{s_k} X_i^k \rangle_{\otimes_{k:1}^m} \circ \left(\langle S - X_i^j \rangle_{\otimes^m} \circ F' \right)$$

in the above expressions the brackets show in which order the composition operator will apply. Let $G = [p] \langle g_k \rangle_{\otimes_{k:1}^m}$ be the outcome of the inner composition. The outer composition's result is:

$$\langle !^{s_k} X_i^k \rangle_{\otimes_{k:1}^m} \circ G \to [p'] \langle !^{s_k} g_k \rangle_{\otimes_{k:1}^m} \qquad [5]$$

where p' is obtained from p by replacing each X_i^k occurrence with $!^{s_k} X_i^k$.
The inner composition is next illustrated.

Basic case of complement repetition.
Let us analyse function $S - X_i$ applied to a set of elements $\{a, b, c\} \subseteq C$. By definition the result is: $S - X(\{a, b, c\}) = (S - a) + (S - b) + (S - c)$. Observing that $S - a = b + c + (S - a - b - c)$, where association by brackets is relevant, and repeating a similar trick for b and c also, it is possible to express $S - X(\{a, b, c\})$ by $b + c + (S - a - b - c) + a + c + (S - a - b - c) + a + b + (S - a - b - c) = (b + c) + (a + c) + (a + b) + (S - a - b - c)$. Given $F \subseteq C$, let $(S - X)_F$ be a function from F to $Bag(F)$ defined as the complement restricted to set F, that is, $(S - X)_F(c) = F - c \;\; \forall c \in F$, and let S_F be the constant function on set $C \backslash F$, that is $S_F(c) = C \backslash F \;\; \forall c \in F$, then it is possible to formally write $S - X$ evaluated in a set $F \subseteq C$ as $S - X(F) = \sum_{c \in F} (S - X)_F + S_F$. It is worth of notice that S_F is a function independent form the application because it is a constant.

In the context of composition $\langle S - X_i^j \rangle \circ F'$, set C above mentioned is C_i and F is the set of colours of the application of class-function f_i^j of tuple F' to a colour of its domain. Thus utilising $\sum_{c \in f_i^j} (S - X)_{f_i^j} + S_{f_i^j}$ to compute $S - X_i^j(f_i^j)$ two computation are required: a) $S_{f_i^j} = S - f_i^j$; b)

$\sum_{c\in f_i^j}(S-X)_{f_i^j}$. While the former is computable by use of re-writing rules for class-functions, the latter seems to require unfolding on colours in f_i^j. Actually, unfolding is not necessary if the semantics of the function is utilised, in fact just two results are possible: it is verifiable that if $|f_i^j| = 1$ the result of $\sum_{c\in f_i^j}(S-X)_{f_i^j}$ is the empty set; if $|f_i^j| > 1$ the result of $\sum_{c\in f_i^j}(S-X)_{f_i^j}$ is f_i^j. Class-functions admissible by the language assume a few basic forms and for these forms it is possible to trace the size.

Let us now analyse the result of composition $\langle S-X_i^j, S-X_i^j\rangle \circ F'$, where $F' = \langle f_i^1, \ldots, f_i^{m_i}\rangle$ By the above dissertation, it can be written $\langle (S-X)_{f_i^j} + S_{f_i^j}, (S-X)_{f_i^j} + S_{f_i^j}\rangle(f_i^j)$. Distributing over the sum the expression becomes:

$$\langle (S-X)_{f_i^j}, (S-X)_{f_i^j}\rangle + \langle (S-X)_{f_i^j}, S_{f_i^j}\rangle + \langle S_{f_i^j}, (S-X)_{f_i^j}\rangle + \langle S_{f_i^j}, S_{f_i^j}\rangle$$

Let us analyse each term:

- $\langle S_{f_i^j}, S_{f_i^j}\rangle$ when applied to set f_i^j results in the computation of $S - f_i^j$;
- $\langle (S-X)_{f_i^j}, S_{f_i^j}\rangle$ when applied to set f_i^j, due to constant nature of $S_{f_i^j}$ and by the previous discussion, results into $\langle f_i^j, S - f_i^j\rangle$ if $|f_i^j| > 1$, the empty set elsewhere;
- $\langle (S-X)_{f_i^j}, (S-X)_{f_i^j}\rangle$ when applied to f_i^j results into $\langle f_i^j, f_i^j\rangle$ if $|f_i^j| \geq 3$, $[X_i^1 = X_i^2]\langle f_i^j, f_i^j\rangle$ if $|f_i^j| = 2$, and the empty set if $|f_i^j| = 1$.

B. Projection and complement repetition. Let us analyse function $\langle X_i^j, S - X_i^j\rangle$ applied to a set of elements $\{a,b,c\} \subseteq C_i$ whose size is greater than 1. By definition the result is: $\langle X_i^j, S - X_i^j\rangle(\{a,b,c\}) = \langle a, S-a\rangle + \langle b, S-b\rangle + \langle c, S-c\rangle$. Observing that $\langle a, S-a\rangle = \langle a, S-a-b-c\rangle + \langle a,b\rangle + \langle a,c\rangle$, and repeating a similar trick for b and c also, it is possible to express $S - X(\{a,b,c\})$ as:

$$\begin{aligned}\langle X_i^j, S - X_i^j\rangle(\{a,b,c\}) = \; & \langle a, S-a-b-c\rangle + \langle a,b\rangle + \langle a,c\rangle + \\ & \langle b, S-b-a-c\rangle + \langle b,a\rangle + \langle b,c\rangle + \\ & \langle c, S-c-a-b\rangle + \langle c,a\rangle + \langle c,b\rangle\end{aligned}$$

By the Cartesian product property the above can be re-written as:

$$\langle X_i, S - X_i\rangle(\{a,b,c\}) = \langle a+b+c, S-a-b-c\rangle + [X_i^1 \neq X_i^2]\langle a+b+c, a+b+c\rangle$$

In the contex of composition $F \circ F'$, set $\{a,b,c\}$ is class function f_i^j of F', thus:

$$\langle X_i^j, S - X_i^j\rangle \circ F' = \langle f_i^j, S - f_i^j\rangle + [X_i^1 \neq X_i^2]\langle f_i^j, f_i^j\rangle$$

This can be further re-written as:

$$\langle X_i^j, S - X_i^j\rangle \circ F' = [X_i^1 \neq X_i^2]\{\underbrace{\langle f_i^j, S - f_i^j\rangle + \langle f_i^j, f_i^j\rangle}_{\langle f_i^j, S\rangle}\} + \underbrace{[X_i^1 = X_i^2]\langle f_i^j, S - f_i^j\rangle}_{\phi}$$

The example analysed the case where $|f_i^j| > 1$. When $|f_i^j| = 1$ then (see Definition D2) the computation is trivial:

$$\langle X_i^j, S - X_i^j\rangle \circ F' = \langle f_i^j, S - f_i^j\rangle$$

It is possible to write a single rule that summarizes both the situations:

$$\langle X_i^j, S - X_i^j\rangle \circ F' \to [X_i^1 \neq X_i^2]\langle X_i \circ F', S - X_i \circ F'\rangle \qquad [6]$$

The splitting of the result in two branches according to the size of f_i^j ($|f_i^j| = 1$ or $|f_i^j| > 1$) is done when the inner class-function composition $S - X_i \circ F'$ is solved.

The above discussion can be generalised to any number of projections X_i^j. and a unique complement. With the notation that $!^0 X_i^j = X_i^j$, let $F = \langle f_1, f_2, \ldots, f_n, f_{n+1}\rangle$ be such that $f_s = !^{k_s} X_i^j$ for each $s = 1 \ldots n$ and $f_{n+1} = S - !^{k_{n+1}} X_i^j$ where $k \in \{0, |C_i|\}$, then

$$F \circ F' = [g]\langle f_1 \circ F', f_2 \circ F', \ldots, f_{n+1} \circ F'\rangle$$

where $[g]$ is a guard which is the conjunction of the following predicates:

- for each $h, t \in \{1, \ldots, n\}$, $h \neq t$ the predicate $!^{k_h} X_i^h = !^{k_t} X_i^t$ is in $[g]$;
- for each $h \in \{1, \ldots, n\}$ the predicate $!^{k_h} X_i^h \neq !^{k_{n+1}} X_i^{n+1}$ is in $[g]$;

Proof.
To prove the above statement consider the function $F : \langle X, S - X - !X\rangle$. We need to evaluate $F(f)$. We proceed to a rewriting of F under this assumption. Adding and subtracting $f + !f$ to the first component the function F it can be rewritten as it follows, where all operations are on multisets:

$$\begin{array}{lll} \langle X, S - X - !X\rangle & \to & \langle X, S - f - !f + f + !f - X - !X\rangle & \to \\ & \to & \langle X, S - f - !f\rangle + \langle X, f + !f - X - !X\rangle & \to \\ & \to & \langle X, S - f - !f\rangle + \langle X, f + !f\rangle - \langle X, X\rangle - \langle X, !X\rangle & (1) \end{array}$$

Assuming first that $f \cap !f = \emptyset$ the support of the last expression is equal to the support of the following expression:

$$\langle X, S - f - !f\rangle + [X_1 \neq X_2, !X_1 \neq X_2]\langle X, f + !f\rangle, \quad \text{iff } x \in f$$

Applying now $F(f)$ we obtain:

$$\sum_{x \in f}\langle X, S - f - !f\rangle + \sum_{x \in f}[X_1 \neq X_2, !X_1 \neq X_2]\langle X, f + !f\rangle$$

and since $S - f - !f$ and $f + !f$ are constants we obtain:

$$\langle \sum_{x \in f} X, S - f - !f\rangle + [X_1 \neq X_2, !X_1 \neq X_2]\langle \sum_{x \in f} X, f + !f\rangle$$

which is equal to:

$$\langle f, S - f - !f\rangle + [X_1 \neq X_2, !X_1 \neq X_2]\langle f, f + !f\rangle$$

Assume now that $f \cap !f \neq \emptyset$, for instance assume $f = \{a, b, e\}$. Expression (1) is

$$\langle X, S \underbrace{-a-b}_{-(a+!a)}\underbrace{-b-c}_{-(b+!b)}\underbrace{-e-f}_{-(e+!e)}\rangle + \langle X, a+b+b+c+e+f, -X-!X\rangle$$

Applying to f it becomes:

$$\langle f, S-a-b-b-c-e-f\rangle + \langle f, a+b+b+c+e+f\rangle - \langle a,a\rangle - \langle b,b\rangle - \langle e,e\rangle - \langle a,b\rangle - \langle b,c\rangle - \langle e,f\rangle$$

we can take $\langle f, \rangle$ out of the second tuple and elide it in the first tuple:

$$\langle f, S-a-b-c-e-f\rangle + \langle f, a+b+c+e+f\rangle - \langle a,a\rangle - \langle b,b\rangle - \langle e,e\rangle - \langle a,b\rangle - \langle b,c\rangle - \langle e,f\rangle$$

folding the result we finally obtain that it is equivalent to the support of:

$$\langle f, S-f-!f\rangle + [X_1 \neq X_2, !X_1 \neq X_2]\langle f, f+!f\rangle$$

C. Generalised-Complement repetition. A generalised-complement on variable X_i^j is a class-function $f(X_i^j)$ expressed as the complement of the sum of several instances of X_i^j, that is

$$f(X_i^j) = S - \sum_{k \in I} !^k X_i^j$$

where $k \in \{0, \ldots, |C_i|\}$ and I is a set of successor indexes. Syntactically it can be rewritten as:

$$S - \sum_{k \in I} !^k X_i^j \equiv \bigcap_{k \in I} S - !^k X_i^j$$

This paragraph illustrates the rewriting rule to solve composition $T \circ T'$ when tuple T is composed by repetitions of generalised-complements. This is the most general form of complements repetition (the previous section "Basic case of complement repetition" discussed a particular case in which each generalised complement in tuple T is actually the complement of a single instance of variable X_i^j).

Generalised-Complement repetition is solved applying a more general algorithm based on the k-projection operator $\mathbf{\Pi}_k$. $\mathbf{\Pi}_k$ is a unary operator on tuples and it maps a m-tuple T into a k-tuple T', with $k < m$, projecting it on the first k components. This operator is discussed in detail in next Sec. 4.3 where it is used to solve critical cases due to the presence of filters in compositions.

The problem of the Generalised-Complement repetition can be formalized as follow:

$$\langle f_1(X_i^j), f_2(X_i^j), \ldots, f_l(X_i^j)\rangle \circ T'$$

where f_i are Generalised-Complement repetitions of variable X_i^j. In order to solve the composition we need to add several variables to set Var; to do this without incur in a global renaming, we add a second dimension in the index denoting the variable's instance, that allows us to name variables related to X_i^j: assuming there are l generalised-complement class-functions in T dependent on X_i^j, set Var is modified replacing X_i^j with $X_i^{j,1}, X_i^{j,2}, \ldots X_i^{j,l}$.

Making a substitution of variable, T is rewritten as T_r:

$$T \to T_r : T_r = \langle f_1(X_i^j \hookleftarrow X_i^{j,1}), f_2(X_i^j \hookleftarrow X_i^{j,2}), \ldots, f_l(X_i^j \hookleftarrow X_i^{j,l})\rangle$$

Tuple T' is expanded duplicating l-times component g at position j (those related to the value that function X_i^j maps to), denoted as g^j, in the following way:

$$T' \to T'_e : T'_e = \langle \ldots, \underbrace{g^{j,1}, g^{j,2}, \ldots, g^{j,l}}_{\text{component } g^j}, \ldots, \rangle$$

The solving of $T \circ T'$ is thus brought to the solving of the following expression:

$$T_r \circ [X_i^{j,1} = X_i^{j,2} \wedge X_i^{j,2} = X_i^{j,3} \wedge \ldots \wedge X_i^{j,l-1} = X_i^{j,l}] \circ T'_e$$

The above expression contains a predicate which can not be solved with the rules till now described. The following rewritings will be justified later on. Basically they extend tuple T_r with a component in order to represents the predicate so that the composition rules can be applied, $\mathbf{\Pi}_l$ is then used to obtain the final result:

$$\mathbf{\Pi}_l\{\langle \underbrace{f_1(X_i^{j,1}), f_2(X_i^{j,2}), \ldots, f_l(X_i^{j,l})}_{T_r}, \underbrace{X_i^{j_1} \cap X_i^{j_2} \cap \ldots \cap X_i^{j_l}}_{\text{predicate in tuple-form}}\rangle \circ T'_e\}$$

Table of elementary symbol compositions of Rule $[A]$ is completed in order to solve elementary compostion involving generalized-complement:

Ruleset $[A]$ continued [7]

$S - \sum_{k\in I} !^k X_i^j \circ F$

$\bigcap_{k\in I}(S - !^k X_i^j \ \circ \ f_i^j)$	if $\lvert f_i^j\rvert = 1, \forall n$
$\sum_{r:0\ldots n-1, !^r X \notin Symb(f)} !^r X \circ g$	if $\lvert f_i^j\rvert > 1, m+1 \le n \le m+k$
S	if $\lvert f_i^j\rvert > 1, n > m+k$

where, $|I| = m$, $1 < m < n$, $|f| = n - m$, $|f_i^j| = n - k$, $0 \le k < n$.

Example 4.2. Consider the following expression:

$$\langle S - X^1, S - X^1 - !X^1\rangle[X^1 \neq X^2] \circ \langle S - X^2, S - X^1\rangle, n \ge 3$$

which is equivalent to:

$\mathbf{\Pi}_2(\langle S - X^1, S - X^1 - !X^1, X^1 \cap S - X^2\rangle \circ \langle S - X^2, S - X^1\rangle) \xrightarrow{[3]}$
$\mathbf{\Pi}_2(\langle S - X^1, S - X^1 - !X^1, X^1\rangle \circ \langle S - X^2, S - X^1\rangle \cap \underbrace{\langle S, S, S - X^2\rangle \circ \langle S - X^2, S - X^1\rangle}_{\langle S,S,S\rangle}) \to$
$\mathbf{\Pi}_2(\langle S - X^1, S - X^1 - !X^1, X^1\rangle \circ \langle S - X^2, S - X^1\rangle)$.
Using the method described in this section, the expression above is rewritten as:
$\mathbf{\Pi}_2(\langle S - X^{1,1}, S - X^{1,2} - !X^{1,2}, X^{1,3}\rangle[X^{1,1} = X^2 \wedge X^{1,2} = X^{1,3}] \circ \langle \underbrace{S - X^2, S - X^2, S - X^2}_{\text{duplicated component}}, S - X^1\rangle) \to$

$\mathbf{\Pi}_2(\langle S - X^{1,1}, S - X^{1,2} - !X^{1,2}, X^{1,3} \cap X^{1,2} \cap X^{1,1}\rangle \circ \langle S - X^2, S - X^2, S - X^2, S - X^1\rangle)$

The tuple's composition can be separately solved for the different unary components of the left-most tuple, according to the intersection distribution property, obtaining:

$\mathbf{\Pi}_2([X^1 \neq X^3 \wedge X^2 \neq X^3 \wedge X^2 \neq !X^3]\langle S, (S - X^2 - !X^2) \circ (S - X^2), S - X^2\rangle)$

where:

$$(S - X^2 - !X^2) \circ (S - X^2) \begin{cases} S & n > 3 \\ S - !^2 X^2 & n = 3 \end{cases}$$

Thus, after applying the projection we obtain:

$$\rightarrow \begin{cases} \langle X^2, S - !^2 X^2\rangle + [X^1 \neq !X^2]\langle S - X^2, S - !^2 X^2\rangle & n = 3 \\ \langle X^2, S\rangle + [X^1 \neq !^2 X^2]\langle S - X^2, S - !^2 X^2\rangle + [X^1 \neq !X^2]\langle S - X^2, S - !^3 X^2\rangle & n = 4 \\ \langle S, S\rangle & n > 4 \end{cases}$$

4.3. Generalizing the composition: tuples with filters

Rules [1] and [2], illustrated as the starting rules in the processing of solving $l \circ l'$, simplify the problem into a subordinate operation, that is the solving of $T_1 \circ [p] \circ T_2$. In several circumstaces $[p]$ does not represent an issue, in these cases in fact the predicate p can be represented in one or both of the tuples applying **filter/guard reductions** rules and the process of solving can continue applying the *tuple composition rules* previously illustrated (Sec. 4.2). Filter reductions are summarized in Appendix under rule-sets $[J]$ and $[H]$ and the process can be overall summarized as[2]:

$$T_1\ [p]\ T_2 \xrightarrow{[J],[H]} T_1' \circ T_2' \xrightarrow{[\text{S4.2}]} l \qquad (3)$$

In detail, the predicate p can be represented into either or both tuples T_1 and T_2, that is one of the following intermediate transformations can hold:

$$T_1[p] \xrightarrow{[H]} T_1' \quad \text{or} \quad [p]T_2 \xrightarrow{[J]} T_2' \quad \text{or} \quad T_1[p'] \xrightarrow{[H]} T_1' \quad \text{and} \quad [p'']T_2 \xrightarrow{[J]} T_2'$$

where p' and p'' are parts forming p (i.e. $p' \circ p'' \equiv p$). Hence the expression respectively becomes:

$$T_1' \circ T_2 \quad \text{or} \quad T_1 \circ T_2' \quad \text{or} \quad T_1' \circ T_2'$$

where *tuple composition rules* can be applied.

An alternative set of rules that can be used by the process of solving in place of those in $[J]$ and $[H]$ or that can be complementary, consists in rewriting the predicate p as a tuple (see rule-set $[K]$), that is $[p] \xrightarrow{[K]} T_p$ and then continuing by using *tuple composition rules* to solve the expression:

$$T_1 \circ T_p \circ T_2$$

There are some cases where neither of the above approaches is effective in the processing of solving due to the lack of rules to go ahead: this happens when a complete filter reduction is not feasible using rule-set $[J]$ or $[H]$ or, in the alternative approach, the solution of the intermediate expressions $T_1 \circ T_p$ or

[2]Notation $\xrightarrow{[\text{S4.2}]}$ represents the application of the rewriting rules explained in Sect.4.2

$T_p \circ T_2$ by means of rules [S4.2] brings back to $[p]$. The next section formally provides the whole set of rewriting rules that allows to solve the composition between tuple in a such case, here we introduce by means of an example the problem illustrating the process and an overview of the method.

Example 4.3. Consider the following cases of compositions where $n = |C_1|$ and where the subscript denoting the color class of the function has been omitted:

1. $\langle S - X^1 \rangle \circ [X^1 = X^2] \circ \langle S - X^2, S - X^2 \rangle, n > 1$
2. $\underbrace{\langle S - X^1, S - X^2 \rangle}_{T_1} \circ [\underbrace{X^1 = X^2 \wedge X^2 \neq X^3}_{p}] \circ \underbrace{\langle S - X^1, S - X^1, S - X^1 \rangle}_{T_2}, n > 1$

In neither of the above expressions the filter can be represented in some component of the tuples using rules in $[J]$,$[H]$. The process of solving thus applies the alternative approach. Rewriting the filters as tuples the two expressions above become respectively:

1. Since $[X^1 = X^2] \overset{[K]}{\rightarrow} \langle X^1 \cap X^2 \rangle$, and $\langle X^1 \cap X^2 \rangle \circ \langle S - X^2, S - X^2 \rangle \overset{[4.2]}{\rightarrow} \langle S - X^2 \rangle$, the whole expression is rewritten into $\langle S - X^1 \rangle \circ \langle S - X^2 \rangle$, which results in:

$$\langle S - X^1 \rangle \circ \langle S - X^2 \rangle \xrightarrow{[4.2]} \begin{cases} \langle S \rangle, & n > 2 \\ \langle X^2 \rangle, & n = 2 \end{cases}$$

2. $[\underbrace{X^1 = X^2 \wedge X^2 \neq X^3}_{p}] \overset{[K]}{\rightarrow} \underbrace{\langle X^1 \cap X^2, X^2 \cap \ S - X^3, X^3 \rangle}_{T_p}$.

   Using tuple composition rules the rightmost composition is solved as:

$$\underbrace{\langle X^1 \cap X^2, X^2 \cap \ S - X^3, X^3 \rangle}_{T_p} \circ \underbrace{\langle S - X^1, S - X^1, S - X^1 \rangle}_{T_2} \xrightarrow{[4.2]} [\underbrace{X^1 = X^2 \wedge X^2 \neq X^3}_{p}] \circ \underbrace{\langle S - X^1, S - X^1, S -}_{T_2}$$

   making the rewriting process fall into a a loop.

Case 2 is the representative of a class of instances requiring an apart treatment. Before formally presenting the rewriting steps let us outline the approach.

Considering that it is possible to project the elements belonging to the image of a tuple function T on the first k components by composing the k-tuple $\langle X^i \rangle_{i=1...k}$ with T, then the preliminary step of the approach builds on a basic result: it is possible to construct from a guarded tuple $T[p] : \mathcal{D} \rightarrow C_i^k$ a tuple $T_e : \mathcal{D} \rightarrow C_i^{k+m}$, $m > 0$, such that T_e's projection on the first k component is equivalent to $T[p]$:

$$\langle X^i \rangle_{i=1...k} \circ T_e \equiv T[p]$$

This is accomplished by adding m components to T, forming T_e, such that whenever p evaluates to $false$ some of these m components maps to $\emptyset$ (and vice versa).

According to this consideration, case 2 of Example 4.3, after constructing T_e for $T_1[p]$, would become:

$$\langle X^i \rangle_{i=1,2} \circ \underbrace{\langle \overbrace{S - X^1, S - X^2}^{T_1}, \overbrace{X^1 \cap X^2 \cap \ S - X^3}^{p} \rangle}_{T_e} \circ \underbrace{\langle S - X^1, S - X^1, S - X^1 \rangle}_{T_2}$$

Applying the rules of Sec. 4.2 for tuple composition the process of solving $T_e \circ T_2$ ends up with:

$$T_e \circ T_2 \xrightarrow{[4.2]} \begin{cases} \left[X^1 \neq X^3 \wedge X^2 \neq X^3\right] \langle S, S, \underbrace{S - X^1}_{extra}\rangle, & n > 2 \\ \langle X^1, X^1, \underbrace{X^1 \cap S - X^1}_{extra}\rangle \to \langle \emptyset, \emptyset, \emptyset\rangle, & n = 2 \end{cases}$$

The non parametric term, $n = 2$, is equivalent to the empty tuple, which corresponds to the composition's final result $\langle \emptyset, \emptyset\rangle$ after projectin on the first two component ($\langle X^i\rangle_{i=1,2}$). Instead the parametric expression, $n > 2$, represents the sets:

$$Y(c) = \{\langle c_1, c_2, c_3\rangle \in C_1^3 \mid c_3 \neq c \wedge c_1 \neq c_3 \wedge c_2 \neq c_3\}, \quad \forall c \in C_1$$

The composition's final result for the case $n > 2$ is the projection of the first two components of Y, that is:

$$\langle X^i\rangle_{i=1,2} \circ Y(c) = \{\langle c_1, c_2\rangle \in C_1^2 \mid \exists c_3, \langle c_1, c_2, c_3\rangle \in Y(c)\}, \quad \forall c \in C_1$$

In this case the problem has been reformulated in terms of a composition of a filtered m-tuple by the k-projection function, with $k < m$:

$$\langle X^i\rangle_{i=1,2}[X^1 \neq X^3 \wedge X^2 \neq X^3]\langle S, S, S - X^1\rangle, \quad n > 2$$

The next section will illustrate the rewriting rule that the process of solving use in order to solve an expression of the following form:

$$\langle X^i\rangle_{i=1,\ldots,k} \circ [p]T$$

It can be verified that the composition in the above expression will lead to the following result:

$$= \begin{cases} \langle S, S\rangle, & |n| > 3, \\ \langle S - X^1, X^1\rangle + [X^1 = X^2]\langle S - X^1, S - X^1\rangle + \langle X^1, S\rangle, & |n| = 3 \end{cases}$$

Summarizing, $T \circ [p] \circ T'$ is rewritten to $\langle X^i\rangle_{i=1,\ldots,k} \circ (T_e \circ T')$. Because $T_e \circ T' \to l \in \mathcal{L}$, it comes down to solve $\langle X^i\rangle_{i=1,\ldots,k} \circ l$. This kind of composition, called *k-projection* of l, must be treated with ad-hoc rules, a particular notation will be adopted for it, that is $\mathbf{\Pi}_k(l)$.

An algorithm parametric with respect to color class cardinality is presented in the following sections. A few extra notations are to be introduced.

4.3.1. Notation

Table 7 summarizes some basic notation used in this section.

Table 7: **NOTATIONS**

T	A m-tuple $\langle f_1, f_2, \ldots, f_m \rangle$
F	A tuple-function $[p]T[p']$ where T is *constant size* and predicates p, p' are in *canonical form*
$\mathcal{C}(F) = C_i^{e_i}$	tuple co-domains are built of one possibly repeated color class C_i, and for this reason subscripts will be omitted in notations: e.g., X_1^2 will be denoted as X^2, S_i as S, $S_{i,k}$ as S_k, and $C_{i,k}$ as C_k
$\mathbf{\Pi}_k(F)$	Projection of tuple-function F on the first k components: $\langle X^i \rangle_{i=1,\ldots,k} \circ F$
$\langle T, f \rangle$	tuple obtained from T appending class function f;
$T \mid_k$	$\langle f_1, f_2, \ldots, f_k \rangle$ restriction of tuple T to the first k component
$T \mid_{\neg k}$	$\langle f_{k+1}, f_{k+2}, \ldots, f_m \rangle$ restriction of the tuple T to the last $m - k$ components
$A \subseteq Var(p)$	A subset of variables of predicate p
$p \mid_A$	restriction of p to basic predicates involving only symbols in A; if $A = \emptyset$ then $p \mid_A \overset{def}{=} true$; $p \mid_A \overset{def}{=} true$ also in case p does not contain any term involving only symbols in A.
$p \mid_{\neg A}$	$p \mid_{Var(p) \setminus A}$
$p \mid_k$	$p \mid_{\{X_1, \ldots, X_k\}}$
$p \mid_{\neg k}$	$p \mid_{\{X_{k+1}, \ldots, X_{k+m}\}}$
$p \mid_=, p \mid_{\neq}$	restrictions of p respectively to its equalities and its inequalities*
$\ominus$	$f \ominus f' \equiv f \cap (S - f')$
$F \subseteq G$	$F \ominus G \equiv \emptyset$

* Observe that $p \equiv p_= \wedge p_{\neq}$, but $p \not\equiv p \mid_k \wedge p \mid_{\overline{k}}$. Let $p = X^1 \neq X^2 \wedge X^1 \neq !X^2 \wedge X^1 \neq X^3$. Then $p \mid_2 = X^1 \neq X^2 \wedge X^1 \neq !X^2$, $p \mid_{\overline{2}} = true$, $p_{\neq} = p$, and $p_= = true$.

4.3.2. Basic steps

As a whole, the process used to solve the problem $T[p]T' \to l$ can be summarized in the following transformations performed by the solution process by applying one or more rewriting steps:

$$T[p]T' \overset{(1)}{\to} \mathbf{\Pi}(T_e \circ T') \overset{(2)}{\to} \mathbf{\Pi}(\sum_i [.]T_i[.]) \overset{(3)}{\to} \sum_i \mathbf{\Pi}([.]T_i[.]) \overset{(4)}{\to} \sum_i \mathbf{\Pi}([.]T_i)[.] \overset{(5)}{\to} l \in \mathcal{L} \qquad (4)$$

Each step can require the application of multiple intermediate rules. In step (1) the problem is transformed, as described in the preface to this section, in a composition between tuples but modifying the codomain of the tuple T which requires subsequently applying a projection on the original codomain, the rules of the step (1) are discussed below in the rest of the paragraph; in step (2) the rules of composition described in Sec. 4.2 are applied; steps (3) and (4) reduce the result of the composition between tuples to a simpler form by operating immediate rules deriving from the properties of the composition and the sum; step (5) requires the solution $\mathbf{\Pi}([.]T_i)$, the respective rules are discussed in the next section.

Step (1) rewritings. The rules applied at this step are based on the following Lemma:

Lemma 4.1. Let $T[p] \in \mathcal{L}$. Then there exists T_e such that $T[p] \equiv \mathbf{\Pi}_k(T_e)$.

Proof:
Let T_p be the tuple equivalent to $[p]$ and T_e be so constructed $T_e = T \times T_p$, then it holds $\mathbf{\Pi}_k(T \times T_p) \equiv T[p]$, because $(T \times T_p)(c) = T(c) \times T_p(c)$, and $T_p(c) = \emptyset$ iff $p(c) = false$. □

The proof of Lemma 4.1 is constructive and is based on an algorithm to compute T_e in the form $T \times T_p$ in which T is extended with an extra tuple T_p that represents the predicate p, however tuple T_e is not unique. In order to reduce the size of T_p actually rule-set $[H]$ is first applied so to represent in T those basic predicates of $[p]$ that can be absorbed. Rule-set $[B]$ given below is successively applied iteratively to each basic predicate in p:

Rule Set $[B]$. (tuple expansion)

$$\begin{array}{lll} T[d(X^i) = C_j \ldots] & \to & \mathbf{\Pi}_k(\langle T, X^i \cap S_j \rangle[\ldots]) \\ T[d(X^i) \neq C_j \ldots] & \to & \mathbf{\Pi}_k(\langle T, X^i \ominus S_j \rangle[\ldots]) \\ T[X^i = !^k X^j \ldots] & \to & \mathbf{\Pi}_k(\langle T, X^i \cap !^k X^j \rangle[\ldots]) \\ T[X^i \neq !^k X^j \ldots] & \to & \mathbf{\Pi}_k(\langle T, X^i \cap S - !^k X^j \rangle[\ldots]) \end{array}$$

where k is the size of T.

Let us consider the following example:

$\langle X^1, S - X^2 \rangle [X^1 \neq X^2 \wedge X^1 \neq !X^2 \wedge X^2 \neq X^3 \wedge X^1 = X^4] \xrightarrow{[H][B]*} \mathbf{\Pi}_2(\langle X^1 \cap S - X^2 \cap S - !X^2 \cap X^4, S - X^2, X^2 \cap S - X^3 \rangle)$

4.3.3. Solving $\mathbf{\Pi}([p]T)$

An initial set of rules, collected under the set $[J]$ provided for convenience in the appendix, are iteratively applied to the expression $[p]T$ in order to return both the T tuple and the filter p to a form suitable for subsequent transformations. Such rules could simplify and eliminate the filter but this is not always the case.

In general, after the application of rules in $[J]$ a filter is formed *solely* by equalities and inequalities between projection symbols referring to tuple-components with cardinality greater than one: in particular the equalities refer to equivalent (syntactically identical) components, whereas inequalities refer to partially overlapping components[3].

Example 4.4. Consider the term $G =: [X^1 \neq X^2 \wedge X^1 \neq !X^2]\langle X^1, S - X^1\rangle$, with $n > 2$:

$$G \xrightarrow{[J].iv} [X^1 \neq !X^2]\langle X^1, (S - X^1) \ominus X^1\rangle \to [X^1 \neq !X^2]\langle X^1, S - X^1\rangle$$
$$\xrightarrow{[J].v} \langle X^1, (S - X^1)\ominus !^{-1}X^1\rangle \to \langle X^1, S - X^1 \cap S {-} !^{-1}X^1\rangle \ .$$

In conjunction with the rules in $[J]$ the following additional rule for filters reveals helpful in some circumstances:

$$[X^i \neq !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots\rangle \to [X^i \neq !^k X^j \ldots]\langle \ldots, f_i \cap !^k f_j, \ldots, f_j \cap !^{-k} f_i, \ldots\rangle +$$
$$[\ldots]\langle \ldots, f_i \ominus !^k f_j, \ldots, f_j, \ldots\rangle + [\ldots]\langle \ldots, f_i \cap !^k f_j, \ldots, f_j \ominus !^{-k} f_i, \ldots\rangle \quad \textbf{if}$$
$$|f_i| > 1 \wedge |f_j| > 1 \wedge f_i \not\equiv !^k f_j \qquad [7]$$

Rule [7] expands $[p]T$ into a sum of terms $[p']T'$ such that either T' has a smaller cardinality than T or $[p']$ is a restriction of $[p]$. For efficiency reasons this rule should be used when strictly necessary, since it increases the number of terms. It is required to apply this rule when the filter does not contain repetitions of the same variable, it contains an inequality such that the compared tuple elements are different and they have both size greater than one.

Example 4.5. Consider the term $F =: [X^1 \neq X^2]\langle S, S - X^1\rangle$, with $n > 2$: observe that in this case Ruleset $[J]$ cannot be applied.

$$F \xrightarrow{[7]} [X^1 \neq X^2]\langle S - X^1, S - X^1\rangle + \langle S \ominus (S - X^1), S - X^1\rangle + \langle S - X^1, (S - X^1) \ominus S\rangle$$
$$\to [X^1 \neq X^2]\langle S - X^1, S - X^1\rangle + \langle X^1, S - X^1\rangle$$

The first rewritings implemented in order to simplify the problem of k-projection of a constant size tuple with filter operate in bringing to common factor those basic predicates of p on which the k-projection does not affect. To simplify the discussion we will always assume that p is the result of a canonic reduction of the predicates.

[Lemma 4.2] Let $p \wedge p'$ be a predicate where p only contains variables in $\{X^1, X^2, \ldots, X^k\}$, then

$$\mathbf{\Pi}_k([p \wedge p']T) \to [p]\mathbf{\Pi}_k([p']T) \qquad [8]$$

Proof:
$\mathbf{\Pi}_k([p \wedge p']T)$ can be rewritten to $\mathbf{\Pi}_k([p] \circ [p']T)$. Because symbols in p refer to the first k elements of T, we can swap symbols $\mathbf{\Pi}_k$ and $[p]$ once $[p]$'s domain has been suitably restricted. □

[3]In the following example notation $\xrightarrow{[J].iv(v)}$ refers to the application of the fourth (fifth) rule in Ruleset$[J]$

In the case $[p']$ in [8] is reduced to being empty, the following transformation is valid, of immediate derivation given the hypothesis:

[Lemma 4.3] Let T be a constant size tuple, then

$$\mathbf{\Pi}_k(T) \to T|_k \qquad [9]$$

Finally, the following rule further simplifies the form of the p' filter in [8]:

[Lemma 4.4]

$$\mathbf{\Pi}_k([p']T) \to [p' \mid_{=,k}]\mathbf{\Pi}_k([p' \mid_{\neq}]T) \qquad [10]$$

Proof:
Due to Lemma 4.2 we just have to show that $\mathbf{\Pi}_k([p]T) \supseteq \mathbf{\Pi}_k([p']T)$, where $p' = p \mid_{=,k} \wedge p_{\neq}$ ($\subseteq$ is immediate, given that $p \Rightarrow p'$). That is, for each c, $s' \in [p']T(c)$ there exists $s \in [p]T(c)$ s.t. $\mathbf{\Pi}_k(s) = \mathbf{\Pi}_k(s')$. Let $s' = \langle c_1, \dots, c_n \rangle$. We need to consider equalities $X^i =!^h X^j$, with $i > k \vee j > k$, which occur in p but not in p'. Under the assumption that p is in canonical form, and equality reduction rules have been applied (Rule-set $[J]$), the following holds: $i < j$, symbol X^j in p occurs just on the above equality, finally $f_i =!^h f_j$ in T (i.e., the i-th and j-th components in T coincide, modulo the successor). The color-tuple s, built in such a way that it initially coincides with s', and whose j-th component (note that $j > k$) c'_j is set equal to $!^{-h}c_i$ (considering all equalities of the form above), belongs to $[p]T(c)$: in fact, $c'_j(=!^{-h}c_i) \in f_j(c)$ because we know that $c_i \in !^h f_j(c)(= f_i(c))$ (remind that $!^h f(c) = \bigcup_{a \in f(c)} !^h a$). Moreover, by construction, $p(s) = true$ and the first k elements of s are the same as in s'. □

Consider the following example, to which lemmas above apply ($n > 2$):

$$\mathbf{\Pi}_2([X^1 \neq X^2 \wedge X^1 \neq !X^2 \wedge X^2 = X^3]\langle S - X^1, S - X^2, S - X^2 \rangle \xrightarrow{[8]} [X^1 \neq X^2 \wedge X^1 \neq !X^2]\mathbf{\Pi}_2([X^2 = X^3]\langle S - X^1, S - X^2, S - X^2 \rangle \xrightarrow{[10],[9]} [X^1 \neq X^2 \wedge X^1 \neq !X^2]\langle S - X^1, S - X^2 \rangle$$

Summarizing, we can hereinafter focus on terms $\mathbf{\Pi}_k([p]T)$ where T is constant-size, and p is a canonical predicate composed from basic predicates of inequality and does not only contain variables in $\{X^1, \dots, X^k\}$.

4.3.4. Solving $\mathbf{\Pi}_k([p]T)$ in a general parametric way

$|C|$ can be a parameter n of the calculus, in this case the result of the solving process may be parametric on the values that n can assume. Usually we can obtain an expression for a fixed value of n or for an interval of values, that is:

$$e \to \begin{cases} e_1, & n = k,\ k \in \mathbb{N} \\ e_2, & l \leq n \leq u,\ l \in \mathbb{N}, u \in \mathbb{N} + \{\infty\} \end{cases}$$

Accordingly, the class-function's cardinalities are parametric in n. For instance, if $f_i = S - X^1 - X^2$ and $|C| = n$, with $n \geq 3$ then:

$$|f_i| = \begin{cases} n-1, & n \geq 3, \quad \text{if } X^1 = X^2 \\ n-2, & n \geq 3, \quad \text{if } X^1 \neq X^2 \end{cases}$$

Lower and an upper bounds for class functions are straightforwardly derived: If $n = K$ then $|f_i| = d_i$, where the actual value of d_i may depend on K and on the form of f_i: for example if $f_i = S - X^j$ then $d_i = K - 1$, then $lb(f_i) = ub(f_i) = d_i$; If $l \leq n \leq u$, then $l - k_i \leq d_i \leq u - k_i$ where k_i depends on f_i. The rules and lemmas so far presented are not enough to always solve the expression $\mathbf{\Pi}_k([p]T)$. In fact, there are some terms $\mathbf{\Pi}_k([p]T)$ to which none of the rules till now presented applies in order to solve the expression, even when n (the color-class cardinality) is fixed. Here are three such terms:

$$\begin{array}{lll} e_1: & \mathbf{\Pi}_2([X^1 \neq X^2 \wedge X^1 \neq X^3 \wedge X^2 \neq X^3]\langle S - X^1, S - X^1, S - X^1\rangle), & n > 2 \\ e_2: & \mathbf{\Pi}_2([X^1 \neq X^2 \wedge X^1 \neq !X^2 \wedge X^1 \neq X^3]\langle S, S, S\rangle), & n > 2 \\ e_3: & \mathbf{\Pi}_1[X^1 \neq X^2 \wedge X^1 \neq !^2 X^2]\langle S - X^1 - !^2 X^1, S - X^1 - !^2 X^1\rangle, & n = 4 \end{array}$$

$$e_1 \xrightarrow{4.2} [X^1 \neq X^2]\mathbf{\Pi}_2([X^1 \neq X^3 \wedge X^2 \neq X^3]\langle S - X^1, S - X^1, S - X^1\rangle) \overset{?}{\rightarrow}$$
$$e_1 \xrightarrow{4.2} [X^1 \neq X^2 \wedge X^1 \neq !X^2]\mathbf{\Pi}_2(X^1 \neq X^3]\langle S, S, S\rangle) \overset{?}{\rightarrow}$$
$$e_3 \overset{?}{\rightarrow}$$

An even more important concern is the ability to figure out a parametric result in the event of terms with an associated constraint of kind $n \geq l$. The rewriting process of such terms raises termination and efficiency issues.

A termination criterion useful for the solving of parametric terms is given by the next claim, whose validity is a direct consequence of k-projection definition of a tuple and because $p(c) \subseteq p(c) \mid_k$, $\forall c \in cd(p)$:

Claim 4.1. $\mathbf{\Pi}_k([p]T) \subseteq [p \mid_k]\mathbf{\Pi}_k(T)$

Assuming we are able to solve $\mathbf{\Pi}_k([p]T)$ fixed n, a brute-force approach would consist in finding out the k-projection result for increasing values of n starting from $n = l$. The following property simplifies the analysis.

Property 5. If for $n = val$ it holds:

$$\mathbf{\Pi}_k([p]T) \equiv [p \mid_k]\mathbf{\Pi}_k(T)$$

then the same is valid $\forall n > val$

The validity of the above property is guaranteed by filter monotonicity: if $p(c) = true$ for $n = val$ then $p(c) = true$ for $n > val$. The stop criterion given by Prop. 5 is eventually met, as n grows up (as it will be formally shown later). The problem is that it is not know a priori when, so we should check for the equivalence $\mathbf{\Pi}_k([p]T) \equiv [p \mid_k]\mathbf{\Pi}_k(T)$ between possibly complex terms at the end of each rewriting branch. Even if theoretically feasible, such an approach is highly inefficient, inelegant, and complex to implement.

4.3.5. Parametric bounds in solving k-projection

In this section it is provided a parametric analysis on n, the size of color-class C, giving intervals of values for it in which the solving of k-projection can be simplified and solved.

In first place it is provided an algorithm calculating an upper value for val as defined in Prop. 5. We call this a *monotonicity-bound* for n.

k-Projection monotonicity-bound

Definition 4.1. A value $u_{\pi_k} \in \mathbb{N}$ such that:

$$\mathbf{\Pi}_k([p]T) \equiv [p\ |_k]\mathbf{\Pi}_k(T),\ \ \forall n > u_{\pi_k}$$

is called a *k-projection monotonicity-bound*

In the treatment it is convenient to study $[p]T(c)$ as a system made up of a set of inequations (modulo-n) between variables $\{X^i\}$, with the implicit constraints $X^i \in f_i(c)(\subseteq C_1)$, $\forall i : 1 \ldots k+m$. Set $[p]T(c)$ coincides with the set of solutions of such a system. We shall use the notation $\{[p]T(c)\}_{\mathcal{S}}$ when we'll refer to this interpretation. In particular: $\mathbf{\Pi}_A\{[p]T(c)\}_{\mathcal{S}}$, with $A \subseteq \{X^1, \ldots X^{k+m}\}$, denotes the projection of system's solutions to subset A of variables.

Example 4.6. For instance consider the following expression:

$$[p]T = [X^1 \neq X^2 \wedge X^1 \neq X^3]\langle S, S, S - X\rangle$$

given a color c in the domain C of the tuple T, its image $[p]T(c)$ is the set of 3-tuple (X^1, X^2, X^3) with $X^1 \in C$, $X^2 \in C$, $X^3 \in C/\{c\}$ and that are solutions of the following system of inequalities:

$$\mathcal{S}_c = \begin{cases} X^1 \neq X^2 \\ X^1 \neq X^3 \end{cases}$$

The image of $c \in C$ of the 2-projection $\mathbf{\Pi}_2([p]T(c)$, if $|C| \geq 2$, is the set of 2-tuple (X^1, X^2) with $X^1 \in C$, $X^2 \in C$ and that solutions of the following system of inequalities:

$$\mathcal{S}_c = \left\{\ X^1 \neq X^2 \right.$$

□

Since T has been assumed constant size, we can think of variables X^i as taking values on discrete domains of cardinality $d_i = |f_i|$.We also know that d_i can be expressed as $n - k_i$ (parametric) or simply k_i (constant), where $k_i \in \mathbb{N}$ is known.

A criterion to find out a value for u_{π_k} builds on the following Lemma 4.5.

[Lemma 4.5] Let $q_{i,p}$ be the number of inequations involving X^i in p and $d_i = |f_i|$, if $d_i > q_{i,p}$ then $\forall c$:

$$\mathbf{\Pi}_{\neg\{X^i\}}\{[p]T(c)\}_{\mathcal{S}} = \{[p\ |_{\neg\{X^i\}}]T(c)\}_{\mathcal{S}}$$

Proof:
$\subseteq$ directly follows from $p \Rightarrow p\ |_{\neg\{X^i\}}$. For convenience, let $i = 1$: given any solution $\langle c_2, \ldots, c_{m+k}\rangle$ of $\{[p\ |_{\neg\{X^1\}}]T(c)\}_{\mathcal{S}}$, because $f_1(c) > q_{1,p}$ there exists $c_1 \in f_1(c)$ such that $\langle c_1, c_2, \ldots, c_{m+k}\rangle$ is a solution of $\{[p]T(c)\}_{\mathcal{S}}$. □

A generalization of Lemma 4.5 is the following (we hereinafter omit the argument c and the subscript $\mathcal{S}$):

[Lemma 4.6] Let $Q_X = \{X^i \in Var(g) : d_i > q_{i,p}\}$. Then

$$\mathbf{\Pi}_{\neg Q_X}([p]T) = [p\ |_{\neg Q_X}]T$$

Proof:
Let $Q_X = A \cup \{X^j\}$: due to Lemma 4.5, $\mathbf{\Pi}_{\neg Q_X}([p]T) = \mathbf{\Pi}_{\neg A}(\mathbf{\Pi}_{\neg\{X^j\}}([p]T) \equiv [p\,|_{\neg\{X^j\}}]T)\dots$ □

Lemma 4.5 and 4.6 give sufficient conditions to relax a k-projection instance by eliminating symbols X^i from p, without affecting the k-projection's outcome. They depend on the cardinality of tuple's components and the number of inequations involving X^i. An immediate application is the following:

Corollary 4.1. Given $\mathbf{\Pi}_k([p]T)$, $\forall i > k : d_i = n - k_i$ then $max\{q_{i,p} + k_i\}_{i>k}$ is a *k-projection upper bound.*

Example 4.7. $\mathbf{\Pi}_1([p]T)$, where

$$p = [X^1 \neq X^4 \wedge X^1 \neq X^2 \wedge X^1 \neq !X^2 \wedge X^2 \neq X^3 \wedge X^2 \neq !^{-1}X^3 \wedge X^2 \neq X^4 \wedge X^2 \neq !X^4 \wedge X^3 \neq X^4]$$
$$T = \langle S - X^1, S, S, S\rangle$$

According to Corollary 4.1, $u_{\pi_1} = max\{6, 3, 4\} = 6$, then $\forall n > 6$ $\mathbf{\Pi}_1([p]T) = \langle S - X^1\rangle$.
□

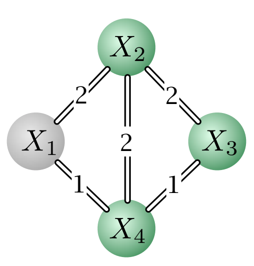
Corollary 4.1 does not assure that u_{π_k} is minimal. Algorithm 1 calculates if it exists a minimal u_{π_k} value. It builds on Lemma 4.5.

Algorithm 1 Tuple k-projection's Upper Bound

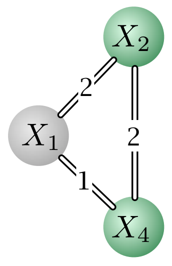
```
Require: a simple tuple $T = \langle f_i\rangle_{i=1,\dots,k+m}$, a canonical filter $p$ formed by inequations
Ensure: a minimal upper-bound for $\mathbf{\Pi}_k([p]T)$
function ProjUb($k, p, T$)
    $u_\pi := 0$
    while ($V_{extra} := \{X^j \in Var(p) : j > k\}) \neq \emptyset$ do
        if ($X_{lb} := \{X^i \in V_{extra} : lb(f_i) > q_{i,p}\}) \neq \emptyset$ then
            $p := p\,|_{\neg X_{lb}}$
        else if ($X_{ub} := \{X^j \in V_{extra} : ub(f_j) > q_{j,p}\}) \neq \emptyset$ then
            $qm := min(\{k_i + q_{i,p},\ X^i \in X_{ub}\})$
            $u_\pi := max(u_\pi, qm)$
            $p := p\,|_{\neg\{X^i \in X_{ub} | k_i + q_{i,p} = qm\}}$
        else
            return $\infty$
        end if
    end while
    return $u_\pi$
end function
```

The **ProjUb**'s output if finite is a *minimal* value u_{π_k} s.t., for each $n > u_{\pi_k}$: $\mathbf{\Pi}_k([p]T) \equiv [p\,|_k]\mathbf{\Pi}_k(T)$. The idea on which the algorithm works can be illustrated representing the inequation system p as a multigraph whose set of nodes is $Var(p)$, such that an edge between X^i and X^j with weight $w_{i,j}$ denotes $w_{i,j}$ inequations relating X^i to X^j.

Figure 3 refers to Example 4.7. Two different colors are used to depict nodes relative to variables with indices little than or equal to k and for variables with indices bigger than k. The figure describes



(a) graph representing p



(b) graph representing $p\mid_{\neg\{X^3\}}$

Figure 3: Computation of the upper bound u_{π_k} illustrated

the first iteration, in which the set of nodes $X^i : i > k$ and $q_{i,p} + k_i$ is minimal is identified and then eliminated according to Lemma 4.6: this happens for $i = 3$ because $q_{3,p} + k_3 = 3$ and thus the set is $\{X^3\}$. The results is a reduced graph corresponding to a system $\{[p\mid_{Var(p)/X^3}]T\}_{\mathbb{S}}$ whose k-projection is the same as the original. Analogously the second iteration considers this graph and identifies $\{X^4\}$ as the candidate for elimination beacuse $d_{4,p'} + k_4 = 3$ is minimal, setting as the running u_{π_1} value the maximum between the current u_{π_1} value and 3. The iteration stops when the reduced system contains only symbols with indices $\leq k$ or there are no nodes candidate for elimination (returning ∞ in the latter case).

Algorithm 1 works both in the case the cardinality of tuple components f_i is fixed and in the case it is parametric in n. If all components are "unbounded" it results in a meaningful upper bound.

Claim 4.2. If $\forall f_i, i > k : ub(f_i) = \infty$ then Algorithm 1 results in $u_{\pi_k} \in \mathbb{N}$

It may therefore be convenient to separately consider the case in which the parameter n assumes a given value, and thus all components of T have a fixed cardinality, and the case in which the parameter n assumes values on an interval and thus so the T's components.

The only critical expression is of the following kind, where for simplicity we consider a 2-tuple:

$$[X^1 \neq X^2]\langle S_h, S - X^1\rangle,\ |C| = n$$

$|S_h|$ is constant, $|S - X^1| = n - 1$. The initial expression is rewritten, using rule [7], into a sum whose first term (that retains the filter) is in turn rewritten into a pair of *simple* terms. The resulting tuples' components have all constant cardinality ($|S_h| - 1$ and $|S_h|$).

$$[X^1 \neq X^2]\langle S_h, S - X^1\rangle \xrightarrow{[7]} [X^1 \neq X^2]\overbrace{\langle S_h \cap S - X^1, S_h \cap S - X^1\rangle}^{T_1} + T' + T''$$
$$T_1 \to \langle S_h \cap S - X^1, S_h \cap S - X^1\rangle[d(X^1) = C_{1,h}] + \langle S_h, S_h\rangle[d(X^1) \neq C_{1,h}]$$

Seemingly the computation of u_{π_k} doesn't depend on the form of functions f_i in T. Indeed, since we are supposing that rule set $[J]$ has been preliminarily applied, if $X^i \neq !^h X^j$ then f^i and $!^h f_j$ at least partially overlap. In many cases we can also suppose that all variables X^i in p have the same domain (i.e., refer to the same f_i), modulo a given $!^h$ (possibly $h = 0$). This is true, in particular, if p meets the following:

Summary Let l and u be respectively the lower and upper color class bounds, that is $l \leq n \leq u$, where $n = |C|$. If $u_{\pi_k} \in \mathbb{N}$ then there are two possible actions:

if $u_{\pi_k} < l$:

$$\mathbf{\Pi}_k([p]T) \to [p\ |_k]\mathbf{\Pi}_k(T), \quad n \in [l, u]$$

if $u_{\pi_k} < u$:

$$\mathbf{\Pi}_k([p]T) \to \begin{cases} [p\ |_k]\mathbf{\Pi}_k(T), & n \in [u_{\pi_k}+1, u] \\ \mathbf{\Pi}_k([p]T), & n \in [l, u_{\pi_k}] \end{cases}$$

k-Projection lower bound In this section we provide a second bound to the parameter n under which the k-projection can be solved in the empty function.

Definition 4.2. A value $l_{\pi_k} \in \mathbb{N}^+$ such that:

$$[p]T \equiv \emptyset, \ \ \forall n < l_{\pi_k}$$

is called a *k-projection null-bound*

We can exploit a basic result of graph-theory. Let $G = (V, E)$ be a simple graph, where an edge uv is a pair of vertices $\{u, v\} \subseteq V$, with $v \neq u$. The order of G is $|V|$. The degree of a vertex, $d(v)$, is the number of adjacent vertices. The maximum degree (or simply, the degree of G) is denoted Δ. An independent set of G is a set of nodes among which there are no edges. The size of the largest independent set is denoted α. A completely connected graph is said a *clique*. The order of the maximum clique contained in G is denoted ω.

Let $r \in \mathbb{N}^+$. A r*−coloring* of G is a map $\varphi : V \to \{1, 2, \dots, r\}$ such that $\varphi(u) \neq \varphi(v)$ for each $uv \in E$. A graph G is said *r-colorable* if G admits a r-coloring. The *chromatic number* of G, $\mathcal{X}(G)$, is the minimum r such that G is *r-colorable*. We know that $max(\{\omega, |V|/\alpha\}) \leq \mathcal{X}(G) \leq min(\{\Delta + 1, |V| + 1 - \alpha\})$, moreover if G is planar, $\mathcal{X}(G) \leq 4$. A number of algorithms computing $\mathcal{X}(G)$ have been developed, based on the *reduction theorem*.

A predicate (system of inequations) p can be associated to a simple graph. This way, the problem of finding l_π comes down to computing its chromatic number, hereinafter denotef $\mathcal{X}(p)$.

Definition 4.3. The graph (V, E) associated with p is such that $V = Symb(p)$, and $uv \in E$ if and only if "$u \neq v$" is in p or $u = !^h X^i$ and $v = !^j X^i$.

Lemma 4.7. $\mathcal{X}(p)$ is a *k-rojection lower bound*, that is for $n < \mathcal{X}(p)$ it holds $p \equiv \emptyset$

Consider again Example 4.7. The graph associated with the filter (definition 4.3) is depicted in Fig. 4. We have $\mathcal{X}(p) = 3$, so $\forall n < 3\ \mathbf{\Pi}_1([p]T) = \emptyset$.

A greater l_π value may be sometimes found taking account of tuple components.

Corollary 4.2. Let $D = \bigcup_{h,\,!^h X^i \in Symb(p)} !^h f_i$, and $|D| \leq n - \alpha$, $\alpha \in \mathbb{N}^+$. Then $\mathcal{X}(p) + \alpha$ is a Projection Lower Bound.

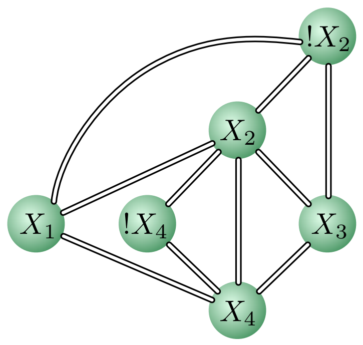


Figure 4: The graph for the computation of the lower bound l_π

Proof:
D represents the union of domains in which *every* symbol occurring in p can take value. If $|D|$ is less than the color-class cardinality (n) of (at least) α, then no $(n-\alpha)$-colouring of the graph associated with p is possible (i.e., no solution of the system $[p]T$ does exist) if $n - \alpha < \mathcal{X}(p)$. □

In particular if $[p]T$ is a pure fixed-point (Definition 6) then $|D| = |f_i|$, for any f_i in T. Consider for example the term:

$$[X^1 \neq X^2 \wedge X^3 \neq X^2 \wedge X^1 \neq X^3]\langle S - X^1 - X^2, S - X^1 - X^2, S - X^1 - X^2\rangle$$

according to Corollary 4.2, $l_\pi = 5$ ($\mathcal{X}(p) = 3$, $\alpha = 2$).

If $[p]T$ is not a pure fixed-point we might obtain an even better result by considering any *subgraph* p' of p: the value we get by applying Corollary 4.2 to p' is a *k-projection lower bound.*

Summary Again, let l and u be respectively the color class bounds, that is $l \leq n \leq u$, where $n = |C|$. After computing l_π there are two possible actions:

if $l_\pi > u$:

$$\mathbf{\Pi}_k([p]T) \rightarrow \emptyset, \quad n \in [l, u]$$

if $l_\pi > l$:

$$\mathbf{\Pi}_k([p]T) \rightarrow \begin{cases} \emptyset, & n \in [l, l_\pi - 1] \\ \mathbf{\Pi}_k([p]T), & n \in [l_\pi, u] \end{cases}$$

4.3.6. Remaining cases

The summaries done in the previous paragraphs on the two bounds (*monotonicity-bound* and *null-bound*) show that the cases to solve are those represented in Fig. 5 in which $l_\pi \leq n \leq u_{\pi_k}$. In order to solve these remaining cases, we consider first the k-projection of expressions $[p]T$ where the predicate p has got a given form called *single-form predicate*.

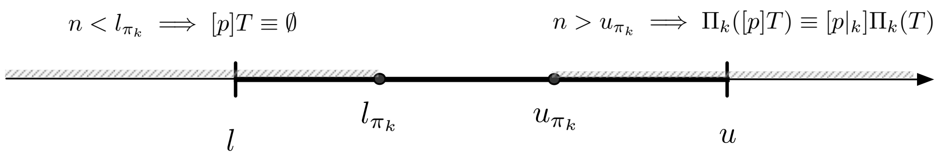

$n < l_{\pi_k} \implies [p]T \equiv \emptyset$
$n > u_{\pi_k} \implies \Pi_k([p]T) \equiv [p|_k]\Pi_k(T)$
l
l_{π_k}
u_{π_k}
u

Figure 5: Remaining cases

Single-form predicates

Definition 4.4. A predicate p is said to have a *single-form* if and only if $|Symb(p)| = |Var(p)|$

In other words p is single-form if each variable X^i belonging to $Var(p)$ appears in just one fashion, say $!^h X^i$. By the way any predicate built on an unordered color class is single-form. The following is an example of a single-form predicate, assuming the color class C ordered:

$$[\ X^1 \neq !X^2 \ \wedge \ X^1 \neq !X^3 \ \wedge \ !X^2 \neq !X^3\ ]$$

where variable X^1 is present with symbol X^1, variable X^2 is present with symbol $!X^2$, and variable X^3 is present with symbol $!X^3$.

When considering single-forms we shall refer to their minimal representative. Generally single-form may be not canonical. Any term $[p]T$ with p single-form is rewritable by rule [7] as a sum:

$$[p]T' + \dots$$

where T' in $[p]T'$ undergoes to the following property:

Property 6. If $X^i \neq !^h X^j \in p$ then $f_i \ = \ !^h f_j$.

Lemma 4.8. Given $[p]T$ such that p is single-form and T satisfies Prop.6, if the graph associated with $p\,|_k$ is a clique and for any f_i in T $|f_i| \geq \mathcal{X}(p)$ then $\mathbf{\Pi}_k([p]T) \equiv [p\,|_k]\mathbf{\Pi}_k(T)$

Proof:
As usual, we have just to prove $\supseteq$. Without losing generality, we suppose $Var(p) = Symb(p)$, $|Var(p)|$ is the tuple's arity, and $|f_i| = \mathcal{X}(p)$. Any solution of the system $[p\,|_k]T\,|_k$ takes the form $\langle c_1, \dots, c_k\rangle$, with $c_i \neq c_j$ for each $i \neq j$. As T is formed only by f_i, and $|f_i| = \mathcal{X}(p)$ ($\geq k$), there must exist a $|f_i|$-colouring $\langle c_1, \dots, c_k, \dots\rangle$ of the graph representing p, i.e., a solution of $[p]T$. □

Lemma 4.8 provides with a simple way to treat unresolved case $\mathbf{\Pi}_k([p]T)$ where p is a *single-form*: if in p there are some pairs $(X^i, !^h X^j)$ of *independent* nodes $X^i, !^h X^j$, with $i, j \leq k$, then we rewrite

$$[p]T \to [X^i = !^h X^j \wedge g]T + [X^i \neq !^h X^j \wedge g]T$$

This way we eventually reduce it to a form where either Lemma 4.8 applies or some of the previous outcomes can be exploited.

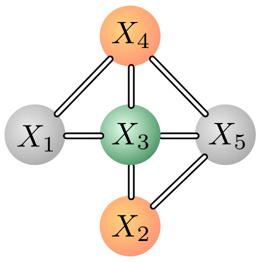


Figure 6: pure fixed-point's projection

Example 4.8. Suppose we have to solve $\mathbf{\Pi}_2([p]\langle S - X^1\rangle_{\otimes^5})$, with $n = 4$ and the graph modelling p is represented in Fig. 6, for which $\mathcal{X}(p) = 3$. The *monotonicity-bound* and the *null-bound* are $u_{\pi_2} = l_{\pi_2} = 4$, so we fall into the category described at the begin of the section. Since X^1, X^2 are independent we carry out the rewriting:

$$[p]\langle S - X^1\rangle_{\otimes^5} \to [X^1 = X^2 \wedge g]\langle S - X^1\rangle_{\otimes^5} + [X^1 \neq X^2 \wedge g]\langle S - X^1\rangle_{\otimes^5}$$

$\mathbf{\Pi}_2([X^1 = X^2 \wedge g]\langle S - X^1\rangle_{\otimes_5}) \overset{[Lm.4.4]}{\longrightarrow} [X^1 = X^2]\mathbf{\Pi}_2([p']\langle S - X^1\rangle_{\otimes_5})$
where p''s graph is obtained from that of p (Fig. 6) by merging X^1 and X^2. Since $\mathcal{X}(p') = 4$, then by Corollary 4.2
$[p']\langle S - X^1\rangle_{\otimes_5} \overset{[Cor.4.2]}{\longrightarrow} \emptyset$
Instead, due to Lemma 4.8:
$\mathbf{\Pi}_2([X^1 \neq X^2 \wedge g]\langle S - X^1\rangle_{\otimes_5}) \overset{Lm.4.8}{\longrightarrow} [X^1 \neq X^2]\langle S - X^1\rangle_{\otimes_2}$.
This is also the final result.

General predicates If p is not a *single-form* then a different set of rules is necessary. If in expression $\mathbf{\Pi}_k([p]T)$ predicate p is not a *single-form*, then the tuple's color class is ordered and p is not rewritable into a form such that for each i there should be at most one symbol $!^h X^i$. In these cases Lemma 4.8 does not generally hold as well as Rule [7]. An example is expression e_3 given at the begin of Sec. 4.3.4:

$$e_3 : \mathbf{\Pi}_1[X^1 \neq X^2 \wedge X^1 \neq !^2 X^2]\langle S - X^1 - !^2 X^1, S - X^1 - !^2 X^1\rangle, n = 4$$

As consequence of Corollary 4.1, there exists some $X^i \in Var(p)$, with $i > k$, involved in at least $u_{\pi_k} + k_i$ inequations. We know that a conjunction of inequalities defined on a fixed size, ordered color-class may be rewritten into a disjunction of equalities. Let $\mathbb{N}_k = \{0, \dots, k\}$:

$$\bigwedge_{h \in H \subseteq \mathbb{N}_{r-1}} X^i \neq !^h X^j \to \bigvee_{h \in \mathbb{N}_{r-1}/H} X^i = !^h X^j \quad \textbf{if } n = r \qquad [11]$$

Starting from $n = l_{\pi_k}$, and supposing H maximal, we repeatedly apply rule [11] until p has been rewritten into a (disjunctive) form for which some of lemma above hold. Because $u \in \mathbb{N}$, the procedure eventually stops. If for a given $n = val$ Lemma 4.3 applies, then the procedure immediately stops because we are in a monotonic setting.

Consider once again Example 4.7 (the corresponding inequation graph is depicted in Figure 4). We know that $l_\pi = u_{\pi_1} = 3$, therefore we just have to treat the case $n = 3$. We can start rewriting

$$X^3 \neq !X^2 \wedge X^3 \neq X^2 \xrightarrow{[11]} X^3 = !^2X^2$$

After having applied Rule Set $[J]$, rule $[11]$, and unified symbols related by any equality (according to the canonical form of filters) a number of times, we finally obtain

$$[X^1 = !^2X^2 \wedge X^1 = X^3 \wedge X^1 = !X^4]\langle S - X^1 - !X^1 - !^2X^1, \ldots\rangle \rightarrow \emptyset$$

5. Conclusions

This report provides a complete set of rules and the algorithms required to solve in general the composition of tuples (that map on sets) in the language defined to represent in a compact way structural properties of SN models. All the presented extensions have been implemented in the SNexpression tool within a modular library that may be easily extended with new features.

An ongoing extension of the work presented in this report concerns the possibility to apply the composition also for tuples that map into multisets (at least when the cardinality of the color classes is fixed (not parametric).

Several applications of the calculus have been found, and completing the rules to treat the composition of tuples has opened the possibility to apply the method also to other applications.

One aspect that deserves a deeper investigation is the analysis of the algorithms complexity.

References

[1] Marco Beccuti, Lorenzo Capra, Massimiliano De Pierro, Giuliana Franceschinis, and Simone Pernice. Deriving symbolic ordinary differential equations from stochastic symmetric nets without unfolding. In *Computer Performance Engineering - 15th European Workshop, EPEW 2018, Paris, France, October 29-30, 2018, Proceedings*, volume 11178 of *LNCS*, pages 30–45. Springer, 2018.

[2] Marco Beccuti, Chiara Fornari, Giuliana Franceschinis, Sami M. Halawani, Omar M. Ba-Rukab, Ab Rahman Ahmad, and Gianfranco Balbo. From symmetric nets to differential equations exploiting model symmetries. *Comput. J.*, 58(1):23–39, 2015.

[3] 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, *ATPN 2005*, volume 3536 of *LNCS*, pages 168–187. Springer, Heidelberg, 2005.

[4] L. Capra, M. De Pierro, and G. Franceschinis. An application example of symbolic calculus for SWN structural relations . In Verronique Carre-Menetrier, Janan Zaytoon, Christos Cassandras, and Xi Ren Cao, editors, *Proc. of 7th IFAC Workshop on Discrete Event Systems*, IFAC, pages 231–236, Reims, France, Sept. 2004. Elsevier.

[5] Lorenzo Capra, Massimiliano De Pierro, and Giuliana Franceschinis. Computing structural properties of symmetric nets. In *Quantitative Evaluation of Systems, 12th International Conference, QEST 2015, Madrid, Spain, September 1-3, 2015, Proceedings*, volume 9259 of *LNCS*, pages 125–140. Springer, 2015.

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

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

[8] Giuliana Franceschinis, Lorenzo Capra, and Massimiliano De Pierro. A tool for symbolic manipulation of arc functions in symmetric net models. In *7th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS'13*. ICST, 1 2014.

A. Rule Sets

All the following rewriting rules assume that any required splitting (leading to different forms and associated constraints) have already been applied. Moreover the tuples in the expressions are already in constant size form, and all predicates are in conjunctive canonical form (i.e. no further simplification is needed).

The dots ... notation is used to indicate any part of tuple or predicate that remains unchanged in the transformed expression.

The parametric cardinality of class C_i is denoted $n_i = |C_i|$; when needed a constrained range of possible values for parameter n_i shall be defined through a lower and an upper bound denoted $[l_i, u_i]$. When not specified it is implicitly assumed that $l_i = 2$ and $u_i = \infty$.

A.1. Rule Sets

The preliminary management of successor is completed by the following intuitive rules. In the following $!^k f(a)$ stands for the linear application of the k^{th} successor to $f(a)$.

Rule Set $[C]$. (successor operator)

$$\begin{array}{rcll}
!^h f & \rightarrow & !^{(h\%k)} f & \textbf{if} \quad n_i = k \wedge (h \geq k \vee h < 0) \\
!^h(f \cap g) & \rightarrow & !^h f \cap !^h g & \\
!^h(f + g) & \rightarrow & !^h f + !^h g & \\
!^h S - f & \rightarrow & S - !^h f & \\
!^r !^h f & \rightarrow & !^{h+r} f & \\
!^0 f & \rightarrow & f &
\end{array}$$

Some additional rules can be defined, provided that the following assumptions hold true.

Let $c^i = [lb_i, ub_i]$ be the constrained range where for parameter n_i. We can assume that after some preliminary rewriting, given any expression E:

- for each $!^h X_i^j : |h| < lb_i$
- if $lb_i = ub_i$, for each $!^h X_i^j : h \geq 0$
- for each n_i verifying $c(E)$, $!^h X_i^j$, $!^m X_i^j$ s.t. $h \neq m$, c: $!^h X_i^j(c) \neq !^m X_i^j(c)$

Given the above assumption, the following rules can be applied.

Rule Set $[D]$. (successor's extra-rules)

$$\begin{array}{rlll}
!^r X_i^j \cap !^q X_i^j & \rightarrow \emptyset & \textbf{if} & r \neq q \\
!^r X_i^j \cap S - !^q X_i^j & \rightarrow !^r X_i^j & \textbf{if} & r \neq q \\
!^r X_i^j = !^q X_i^j & \rightarrow false & \textbf{if} & r \neq q \\
!^r X_i^j \neq !^q X_i^j & \rightarrow true & \textbf{if} & r \neq q
\end{array}$$

Rule Set $[E]$. (predicate basic reductions)

$$\begin{array}{rcll}
X^i \neq !^r X^j & \to & true & \textbf{if} \quad X^i = !^k X^j \wedge r \neq k \\
X^i = !^r X^j \wedge X^i = !^k X^j & \to & false & \textbf{if} \quad r \neq k \\
X^i \neq X^j \wedge X^i \neq !X^j \ldots \wedge X^i \neq !^{k-1} X^j & \to & false & \textbf{if} \quad n = k
\end{array}$$

Rule Set $[F]$. (empty intersection form)

$\bigcap_{!^q X_i^j \in A} S - !^q X_i^j \to \emptyset \quad \textbf{if } ub_i \leq |A| \qquad \bigcap_{X_i^j \in A} S_{i,k} \bigcap_j S - X_i^j \to \emptyset \quad \textbf{if } n_{i,k} \leq |A|$

Rule Set $[G]$. (Intersection between elementary functions)

$$\begin{array}{rll}
!^r X_i^j \cap !^q X_i^j & \to \emptyset & \textbf{if} \quad r \neq q \\
!^r X_i^j \cap S - !^q X_i^j & \to !^r X_i^j & \textbf{if} \quad r \neq q \\
!^r X_i^j \cap S - !^r X_i^j & \to \emptyset & \\
!^r X_i^j = !^q X_i^j & \to false & \textbf{if} \quad r \neq q \\
!^r X_i^j \neq !^q X_i^j & \to true & \textbf{if} \quad r \neq q
\end{array}$$

Rule Set $[H]$. (Reduction of guard predicates in intersection forms)

$$\begin{array}{lcl}
\langle \ldots, X^i \ldots, \ldots \rangle [X^i \in C_{1,k} \wedge \ldots] & \longrightarrow & \langle \ldots, X^i \cap S_{\{k\}} \ldots, \ldots \rangle [\ldots] \\
\langle \ldots, X^i \ldots, \ldots \rangle [X^i \notin C_{1,k} \wedge \ldots] & \longrightarrow & \langle \ldots, X^i \ominus S_{\{k\}} \ldots, \ldots \rangle [\ldots] \\
\langle \ldots, !^h X^i \ldots, \ldots, \rangle [X^i = !^k X^j \wedge \ldots] & \longrightarrow & \langle \ldots, !^h X^i \cap !^{h+k} X^j \ldots, \ldots \rangle [\ldots] \\
\langle \ldots, !^h X^i \ldots, \ldots, \rangle [X^i \neq !^k X^j \wedge \ldots] & \longrightarrow & \langle \ldots, !^h X^i \cap S - !^{h+k} X^j \ldots, \ldots \rangle [\ldots]
\end{array}$$

For example:

$$\langle X^1, S - X^1 \rangle [X^1 \neq !X^2] \xrightarrow{[H]} \langle X^1 \cap S - !X^2, S - X^1 \rangle$$

Rule Set $[I]$. (Reduction of intersection-forms into guard predicates)

$$\begin{array}{lcl}
\langle \ldots, X^i \cap S_{\{k\}} \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, X^i \ldots, \ldots \rangle [X^i \in C_{1,k} \ldots] \\
\langle \ldots, X^i \ominus S_{\{k\}} \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, X^i \ldots, \ldots \rangle [X^i \notin C_{1,k} \ldots] \\
\langle \ldots, S - X^i \cap S_{\{k\}} \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, S - X^i \cap S_{\{k\}} \ldots, \ldots \rangle [X^i \in C_{1,k} \ldots] + \\
 & & \langle \ldots, S_{\{k\}} \ldots, \ldots \rangle [X^i \notin C_{1,k} \ldots] \quad \textbf{if} \neg([\ldots] \Rightarrow X^i \in C_{1,k}) \\
\langle \ldots, !^h X^i \cap !^k X^j \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, !^h X^i \ldots, \ldots, \rangle [X^i = !^{k-h} X^j \ldots] \\
\langle \ldots, !^h X^i \cap S - !^k X^j \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, !^h X^i \ldots, \ldots, \rangle [X^i \neq !^{k-h} X^j \ldots] \\
\langle \ldots, S - !^h X^i - !^k X^j \ldots, \ldots \rangle [\ldots] & \longrightarrow & \langle \ldots, S - !^h X^i - !^k X^j \ldots, \ldots \rangle [X^i \neq !^{k-h} X^j \ldots] + \\
 & & \langle \ldots, S - !^h X^i \ldots, \ldots \rangle [X^i = !^{k-h} X^j \ldots] \quad \textbf{if } i \neq j
\end{array}$$

In the following $f \ominus f' \equiv f \cap (S - f')$. The pattern $!^k f$ matches the form f for $k =: 0$ ($!^0 f \equiv f$). The first rule-set applies to filters.

Rule Set $[J]$. (Reduction of filter predicates into intersection forms)

$$\begin{array}{lcl}
[X^i \in C_k \ldots]\langle \ldots, f_i, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i \cap S_k, \ldots \rangle \\
[X^i \notin C_k \ldots]\langle \ldots, f_i, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i \ominus S_k, \ldots \rangle \\
\\
[X^i = !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i, \ldots, f_j \cap !^{-k} f_i, \ldots \rangle \quad \textbf{if } |f_i| = 1 \\
[X^i \neq !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i, \ldots, f_j \ominus !^{-k} f_i, \ldots \rangle \quad \textbf{if } |f_i| = 1 \\
[X^i = !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i \cap !^k f_j, \ldots, f_j, \ldots \rangle \quad \textbf{if } |f_j| = 1 \\
[X^i \neq !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i \ominus !^k f_j, \ldots, f_j, \ldots \rangle \quad \textbf{if } |f_j| = 1 \\
\\
[X^i = !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [X^i = !^k X^j \ldots]\langle \ldots, f_i \cap !^k f_j, \ldots, f_j \cap !^{-k} f_i, \ldots \rangle \\
 & & \qquad \textbf{if } |f_i| > 1 \wedge |f_j| > 1 \wedge f_i \not\equiv !^k f_j \\
[X^i \neq !^k X^j \ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle & \rightarrow & [\ldots]\langle \ldots, f_i, \ldots, f_j, \ldots \rangle \\
 & & \qquad \textbf{if } |f_i| > 1 \wedge |f_j| > 1 \wedge f_i \cap !^k f_j \equiv \emptyset
\end{array}$$

Rule Set $[K]$. (Transforming a predicate into a tuple)
Rewriting of basic predicates into intersection forms

$$\begin{array}{lcl}
X_i^j = X_i^k & \rightarrow & \langle \ldots, \overbrace{X_i^j \cap X_i^k}^{j^{\text{th}} \text{ component}}, \ldots, \overbrace{X_i^k \cap X_i^j}^{k^{\text{th}} \text{ component}}, \ldots \rangle \\
X_i^j \neq X_i^k & \rightarrow & \langle \ldots, \overbrace{X_i^j \cap S - X_i^k}^{j^{\text{th}} \text{ component}}, \ldots, \overbrace{X_i^k \cap X_i^j}^{k^{\text{th}} \text{ component}}, \ldots \rangle \\
X_i^j \in C_{i,l} & \rightarrow & \langle \ldots, \overbrace{X_i^j \cap S_{i,l}}^{j^{\text{th}} \text{ component}}, \ldots \rangle \\
X_i^j \notin C_{i,l} & \rightarrow & \langle \ldots, \overbrace{X_i^j \cap (S - S_{i,l})}^{j^{\text{th}} \text{ component}}, \ldots \rangle
\end{array}$$

Basic predicate $d(X_i^j) = d(X_i^k)$ is rewritten as

$$\bigvee_{l=1}^{||C_i||} X_i^j \in C_{i,l} \wedge X_i^k \in C_{i,l}$$

thus:

$$d(X_i^j) = d(X_i^k) \rightarrow \sum_{l=1}^{||C_i||} \langle \ldots, \overbrace{X_i^j \cap (S - S_{i,l})}^{j^{\text{th}} \text{ component}}, \ldots, \overbrace{X_i^k \cap (S - S_{i,l})}^{k^{\text{th}} \text{ component}}, \ldots \rangle$$

The above rules can be generalized to basic predicates involving the successors of the variables.

B. Composition $T \circ T'[p]$ with $|Var(T)| = 1$

This section of the Appendix provides with a set of rules that are alternative to those described in Sec. 4.2.2 to solve cases **A**, **B** and **C** and to the rules described in Sec. 4.3.2.

After the application of Rule [3] as described in paragraph **Step-1 rewritings: Simplification of the left-tuple's form** of Sec. 4.2.2 , solving the composition between arbitrary functions is reduced to solve the composition

$$T \circ T'[p]$$

with T being a tuple containing intersection-form such that $Var(T) = \{X_i^j\}$. Moreover we assume that $T'[p]$ is a *constant size* tuple (see Definition 3.2 and Property 1). Under the hypothesis above the following rewriting holds:

$$T \circ T'[p] \rightarrow T_{X_i^j \hookleftarrow X_i^1} \circ \langle g \rangle [p] \qquad [12]$$

where $T_{X_i^j \hookleftarrow X_i^1}$ is obtained by replacing X_i^j by X_i^1 in T, and g is the $j^{\mathtt{th}}$ component of color C_i in T'.

For sake of readabilty subscript i used in the class functions to denote the color-class will be omitted with the exception of rules definition.

Example B.1. The following example illustrates the application of the steps above mentioned:
$\langle S - X^1, S - X^2, S - X^2 \cap X^1 \cap X^3 \rangle \circ \langle S - X^1, S, S - X^2 \rangle$
$\overset{[3]}{\rightarrow} \big(\langle S - X^1, S, X^1 \rangle \cap \langle S, S - X^2, S - X^2 \rangle \cap \langle S, S, X^3 \rangle\big) \circ \langle S - X^1, S, S - X^2 \rangle$
$\overset{\text{Prop.2}}{\rightarrow} \langle S - X^1, S, X^1 \rangle \circ \langle S - X^1, S, S - X^2 \rangle \cap \langle S, S - X^2, S - X^2 \rangle \circ \langle S - X^1, S, S - X^2 \rangle \cap \langle S, S, X^3 \rangle \circ \langle S - X^1, S, S - X^2 \rangle$
$\overset{[12]}{\rightarrow} \langle S - X^1, S, X^1 \rangle \circ \langle S - X^1 \rangle \cap \langle S, S - X^1, S - X^1 \rangle \circ \langle S \rangle \cap \langle S, S, X^1 \rangle \circ \langle S - X^2 \rangle$

In the following we will omit $X^j \hookleftarrow X^1$ in the subscript of tuple T, hence we consider tuple T as *unary*.

Let $T = \langle f_i \rangle_{i=1,\dots,k}$ be unary, there are two cases that can be immediately solved:

- if symbol X^1 occurs just in position l in T the composition outcome is the tuple obtained from T by replacing f_l with $f_l \circ g$;
- If $|g| = 1$ the composition results is $\langle f_i \circ g \rangle_{i=1,\dots,k}$.

However, more generally, disregarding constant functions, other cases whose solving is not straightforward are possible: namely they are when in tuple T the only intersection forms present are projections and generalized complements. We consider separately these situations because different set of rules are applied:

I at least one component of T is a projection;

II no component of T is a projection.

Case I Let $T = \langle f_j \rangle_{j=1}^{k}$, if l is the number of X_i^1 projections, we assume for simplicity the following re-ordering of T's components:

$f_j = !^{h_j} X_i^1$ if $j \leq l \leq k$

$f_j = \bigcap_{s \in H_j} S - !^s X_i^1$, if $j > l$

That is:

$$\langle \underbrace{!^{h_1} X_i^1, \ldots, !^{h_l} X_i^1}_{\text{projections}}, \overbrace{\bigcap_{s \in H_{l+1}} S - !^s X_i^1, \ldots, \bigcap_{s \in H_k} S - !^s X_i^1}^{\text{generalized complements}} \rangle$$

Then the following rule holds:

$$T \circ \langle g \rangle \to [p] \langle !^{h_1} g, S, \ldots, S \rangle \qquad [13]$$

where filter $[p]$ is in conjunctive normal form and it is defined as follows:

- $\forall j = 2, \ldots, l$ there is an equality $X_i^1 = !^{h_1 - h_j} X_i^j$
- $\forall j \in l, \ldots, k$, $s \in H_j$ there is an inequality $X_i^1 \neq !^{h_1 - s} X^j$

Proof:
To prove the validity of the rewriting rule [13] we use the following properties of the transpose operator:

- $(T^{\mathtt{t}})^{\mathtt{t}} \equiv T$
- $(T[p])^{\mathtt{t}} \equiv [p]^{\mathtt{t}} T^{\mathtt{t}}$

The transpose $T^{\mathtt{t}}$ has the form $\langle \bigcap_{j=1}^{k} f'_j \rangle$, where f'_j is obtained from f_j by replacing X^1 with X^j. According to rule-set $[H]$ the expression can be rewritten as $\langle !^{h_1} X^1 \rangle [p']$, where p' is a conjunctive form in which:

- $\forall j : 2 \ldots l$ there is an equality $X^1 = !^{h_j - h_1} X^j$,
- $\forall j : l + 1 \ldots k$, $s \in H_j$ there is an inequality $X^1 \neq !^{s - h_1} X^j$

By transposing again we get $[p] \langle !^{h_1} X^1 \rangle^{\mathtt{t}} \equiv T$. Since $\langle !^{h_1} X^1 \rangle^{\mathtt{t}} \equiv \langle !^{h_1} X^1, S, \ldots, S \rangle$, the composition $T \circ \langle g \rangle$ results in

$$[p] \langle !^{h_1} g, S, \ldots, S \rangle$$

□

Observe that the following equivalence holds:

$$[p] \langle !^{h_1} g, S, \ldots, S \rangle \to [p] \langle f_j \circ g \rangle_{j=1}^{k} \qquad [14]$$

because in general $\langle f_j \rangle_{j=1}^{k} \circ g \subseteq \langle f_j \circ g \rangle_{j=1}^{k}$. The advantage of this representation is that smaller size functions than S may result as tuple components.

The above result straightforwardly extends to any left tuple containing only one variable symbol.

Example B.2. Application of the alternative rules:

$\langle X^2, S - X^2, S - X^2 \cap S - !X^2, !X^2 \rangle \circ \langle X^1, S - X^1 \rangle$ with $|C_1| \geq 4$

$\overset{[12]}{\to} \langle X^1, S - X^1, S - X^1 \cap S - !X^1, !X^1 \rangle \circ \langle S - X^1 \rangle$

$\overset{[13]}{\to} [X^1 = !^{-1} X^4 \wedge X^1 \neq X^2 \wedge X^1 \neq !^{-1} X^3 \wedge X^1 \neq X^3] \, \langle S - X^1, S, S, S \rangle$

$\overset{[14]}{\to} [X^1 = !^{-1} X^4 \wedge X^1 \neq X^2 \wedge X^1 \neq !^{-1} X^3 \wedge X^1 \neq X^3] \, \langle S - X^1, S, S, S - !X^1 \rangle$

Case II All elements of T are generalized-complements, that is:

$$T = \langle f_j \rangle_{j=1}^{k} \text{ with } f_j = \bigcap_{s \in H_j} S{-}!^s X_i^1, |H_j| > 0$$

It is possible to translate the composition in a solvable form by using the tuple k-projection and the outcome of the previous Section B.

Consider in fact the tuple $T_e = \langle T, X_i^1 \rangle$, with the same domain as T, and observe that $\mathbf{\Pi}_k(T_e) = T$. Since $T_e \circ \langle g \rangle \equiv F \in \mathcal{L}$ (Section B), we get $T \circ \langle g \rangle \equiv \mathbf{\Pi}_k(F)$, due to the associativity of the composition.

$$T \circ \langle g \rangle \to \mathbf{\Pi}_k(\langle T, X_i^1 \rangle \circ \langle g \rangle) \qquad [15]$$

Example B.3. k-projection to solve generalized-complement tuple:

$\langle S - X^1, S - X^1 \cap S{-}!X^1 \rangle \circ \langle S - X^1 \rangle \quad |C_1| \geq 4$

$\overset{[15]}{\to} \mathbf{\Pi}_2(\langle S - X^1, S - X_1^1 \cap S{-}!X^1, X^1 \rangle \circ \langle S - X^1 \rangle)$

$\overset{[13]}{\to} \mathbf{\Pi}_2([X_1^1 \neq X_1^3 \wedge X_1^2 \neq X_1^3 \wedge X_1^2 \neq !^{-1} X_1^3] \langle S, S, S - X_1^1 \rangle) \to \ldots$