Integrating Symmetries and Symbolic Enabling Test for Efficient Simulation of SWNs

Lorenzo Capra
Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20131, Milano (Italy)
capra@dico.unimi.it

Abstract— **(Stochastic) Colored Petri Nets are a formalism widely used to specify and analyze distributed discrete-event systems. Determining the set of transition color instances enabled in a given marking is a basic task affecting analysis techniques based on state-space exploration, model-checking, and especially discrete-event simulation (the latter is an interesting alternative when analytical solutions are unfeasible due to state-space explosion). An algebraic approach to enabling test has been recently proposed as kernel for efficient state-space exploration of SWN, a Stochastic CPN flavor retaining expressive power. The approach is symbolic because it directly manipulates arc functions by means of rewriting rules. What makes interesting the SWN formalism is the ability of exploiting behavioral system's symmetries, thanks to structured color annotations. This paper illustrates in a semi-formal way how the symmetry-based technique typical of SWN can be profitably combined with the SWN symbolic enabling test, giving rise to a fully symbolic simulation kernel. An application example to a workflow model is presented.**

Keywords— **Colored Petri Nets, Enabling Test, Symbolic Techniques**

I. Introduction

Colored Petri Nets (CPN) [1] are a high-level Petri Net formalism widely used to specify and analyze several kinds of discrete-event systems, e.g. network protocols, workflow models, flexible manufacturing systems, distributed algorithms, etc. In particular stochastic CPN extensions are attractive due to the possibility they offer of studying the performances of systems, either solving the associated stochastic process or via discrete-event simulation.

Stochastic Well-formed Nets (SWN) [2] are a CPN subclass retaining expressive power, that may be conveniently used to model discrete-event systems exhibiting behavioral symmetries, that can be naturally encoded on the SWN structured syntax. In addition to the effective state-space based analysis techniques exploiting symmetries, typical of SWN [3], an algebraic calculus have been recently proposed [4] which allows some common tasks of SWN's analysis algorithms to be developed in an efficient way taking advantage of structural considerations. In [5] in particular the SWN calculus has been used to define an algorithm for finding enabled instances of a colored transition in a given marking without any unfolding. The enabling test has been recognized to be a critical task in reducing the complexity of performance-oriented analysis by means of discrete-event simulation [6], but is also a relevant task during state-space generation, as well as in model checking. The approach proposed in [5] relies upon moving the task complexity from run-time to structural analysis: it essentially consists in rewriting SWN arc functions according to algebraic rules, with the purpose of finding a formal expression for local enabling functions.

This paper illustrates in a semi-formal way how the symmetry-based techniques typical of SWN can be profitably merged to the SWN symbolic enabling test, giving rise to a fully symbolic simulation kernel for SWN. An application example of the SWN symbolic enabling test approach to a SWN workflow model is also presented. The example points out the benefits coming from the integration of the symbolic enabling test approach to the SWN symbolic firing rule, on the perspective of providing an efficient, algebraic, discrete-event simulation engine for SWN.

The discussion is organized as follows. Section II recalls in pragmatic way the SWN formalism and introduces the enabling test computation problem in SWN, providing the state-of-the-art with regard to several techniques which may be utilized to approach it. Section III overviews the theoretical framework that the symbolic approach here proposed is based upon. Section IV describes the algorithm recently proposed with the aim to solve in efficient and general way the enabling test task in SWN. Section V illustrates how this approach can be naturally merged to the symbolic firing mechanism of SWN that exploits symmetries. Section VI shows an application to a SWN workflow model, pointing out the possible combination of the symbolic enabling test with the symbolic marking notion, that SWN algorithms for effective state-space exploration rely upon. An outline of the presented work and a discussion about possible evolutions conclude the paper.

II. The SWN Formalism

Colored PN ([1]) are a major extension of PN belonging to the High-Level PN category ([7]). Several classes of CPN have been developed worldwide. This work focuses on Stochastic Well-formed Nets (SWN) [2], a CPN flavor retaining expressive power, characterized by a structured syntax used for effective performance analysis [3]. This paper just introduces SWN informally using a practical example: Fig. 1 shows the workflow of the document reviewing process that is op-

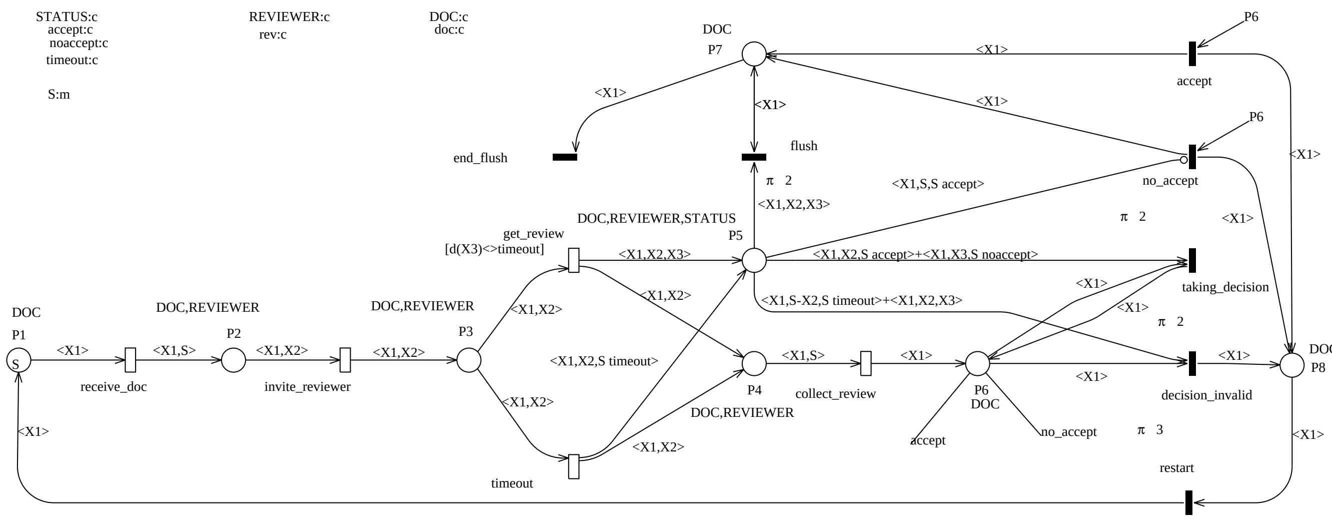

Fig. 1. SWN Workflow of Reviewing Process.

erated by a given organization. The model will be explained in Section VI.

Formally a SWN is a tuple

$$(P, T, \{C_1, ..., C_n\}, \mathcal{C}, W^-, W^+, H, \mathbf{m}_0)$$

P is the finite set of the *places* of the net, T is the finite set of the *transitions* of the net. With respect to ordinary PN, places may contain tokens of different identity, these are called colors. A marking maps each place to a multiset of colors (denoted by a formal sum) belonging to the place color domain. $\mathbf{m}_0$ defines the initial marking of the net. In the example place P_1 initially contains all the colors representing the documents to be reviewed ($\sum_{d_k \in DOC} 1.d_k$). Color domains are defined as Cartesian products of basic classes of colors, $C_1 : DOC$, $C_2 : REVIEWER$ and $C_3 : STATUS$ in the example (look e.g. at places P_2, P_4). Basic color classes C_i may be partitioned into sub-classes denoted $C_i = \bigcup_j C_{i,j}$ (e.g. class $STATUS$ is partitioned into singletons representing the status of a single revision). Transitions are colored too, and their color domains are implicitly derived from functions labeling surrounding arcs. A colored transition actually folds together many elementary ones: in CPN one talks about instances of a colored transition to denote its own single colored entities. Transition *invite_reviewer* in the example represents many elementary instances (tuples) (d_k, r_j). Function $\mathcal{C}$ assigns to each $s \in P \cup T$ a color domain.

W^-,W^+ and H (called input, output and inhibitor arc functions, respectively) are mappings assigning to each pair (t, p) a function $F : \mathcal{C}(t) \rightarrow Bag(\mathcal{C}(p))$, in turn F assigns to each instance of t a multiset on the color domain of place p. SWN impose constraints on the arc function syntax. A color class-i function, hereafter simply class-function, is a mapping $\mathcal{C}(t) \rightarrow C_i$ and may be an integer linear combination of elementary functions denoted by: X_k, S, $S_{C_{i,j}}$, $S - X_k$. A function tuple is a tuple of class functions: $\langle X_1, X_2\rangle$ surrounding *invite_reviewer* is a function tuple where the first component is a class-*1* function and the second one is a class-*2* function.

A guard may be associated to transition t to restrict its set of admissible color instances. A guard $[guard]$ is a function $\mathcal{C}(t) \rightarrow \mathcal{C}(t)$ useful to filter out from the domain of a transition t instances which do not satisfy a given Boolean predicate. In the example, transition *get_review* is guarded by $[d(X_3) <> timeout]$. The Boolean predicate is built upon a limited set of basic predicates: $X_i = X_j$, $d(X_i) = d(X_j)$, $X_i \neq X_j$, $d(X_i) \neq d(X_j)$, $d(X_i)$ denoting the co-domain of X_i. Finally, $W^-(t,p)$,W^+ and H are integer linear combinations of function tuples possibly left-composed to a guard (e.g., $\langle X_1, X_2\rangle \circ [X_1 <> X_2]$).

Now, let us introduce the semantics of the elementary symbols: X_k projects the arguments on its *k-th* components, possibly a successor operator $!^d$ is applied to X_k when C_i ($C_i = d(X_k)$) elements are circularly ordered; S is a constant mapping each element of its domain to the (multi)set C_i, assumed it is a class-i function; $S_{i,j}$ is a constant mapping to the (multi)set $C_{i,j}$, assumed it is a class-i function and C_i is partitioned in sub-classes; $S - X_k$ is the complement of the k-th component of the argument. Function tuple applications result in Cartesian products. Referring to the example, the application of $\langle X_1, X_2\rangle$ to the color instance (d_k, r_j) of *invite_reviewer* results in $\langle X_1, X_2\rangle(d_k, r_j) = X_1(d_k, r_j) \times X_2(d_k, r_j) = (d_k, r_j)$.

A priority level is associated to each transition: level 0 is reserved for *timed* transitions (graphically represented as white boxes), while all other levels are for *immediate* transitions (graphically represented as black bars), which fire in zero time. An exponential *firing rate* is associated to each timed transition, while a *weight* is associated to each immediate transition to probabilistically solve conflicts between immediate transitions with equal priority.

A reachability-graph is built starting from $\mathbf{m}_0$ and the enabled transition set. In stochastic PN extensions such graph is associated to a Markovian stochastic process for performance analysis purposes. When no analytical methods are feasible then discrete-event simulation can be carried out. The structured syntax of SWN arc functions allows system symmetries (implicitly encoded into SWN models) to be exploited for the building of a compact Symbolic Reachability Graph (SRG) and a corresponding stochastic process (a *lumped* CTMC), or to perform symbolic discrete-event simulation runs [3].

A. Enabling Test in SWN

The determination of the enabled instances of a transition in a given marking is a common task for many analysis techniques which need to explore the state-graph of a CPN model. Focusing on SWN, this task might be efficiently achieved exploiting the structured syntax of the paradigm. The problem may be stated in terms of looking for a function $EN(\mathbf{m}, \mathbf{t})$ which for a given marking and transition gives the enabled instances.

The enabling test might be trivially done passing through the unfolding of a SWN model. This solution however is highly inefficient. The enabling expression for SWN is: $W^-(t,p)(c) \leq \mathbf{m}(p) < H(t,p)(c)$, $\forall p$, in which comparison operators are extended to multisets. $EN(\mathbf{m}, \mathbf{t})$ is obtained by evaluating this expression for each color c of t. This kind of method however does not exploit the symmetries encoded in SWN arc functions. In literature several works have been presented trying to exploit such feature in order to enhance the computation of $EN(\mathbf{m})$.

The optimization proposed in [6] was integrated in the *GreatSPN* tool to improve the efficiency of the SWN discrete-event simulation engine. This optimization implements some heuristics that may be applied when SWN arc functions match enough simple patterns and reveals effective on a restricted set of practical cases. For instance function tuples cannot include the complement function, and guards are only partially treated.

The approach proposed in [8] is more related to the contents of this paper and presents some similarities. It provides a basic calculus for SWN enabling test, however is only partially symbolic, does not explicitly deal with guards, and considers only a subset of class-functions.

An enhancement of [8] has been recently presented [5]. Exploiting a calculus recently presented in [4] the technique is generalized to comprise all SWN features and this is done symbolically, i.e. without unfolding SWN nodes. The hypothesis is that W^-,W^+ and H may be manipulated by means of operators working on function tuples in order to obtain a formal expression for a given colored transition t which, applied to a marking $\mathbf{m}$, returns its enabled colored instances. In this paper we show an application of the technique to symbolic discrete-event simulation of SWN workflow models.

III. Overviewing the SWN Calculus

[4] introduces an algebraic framework allowing efficient implementation of several SWN analysis algorithms. The framework is characterized by a high-level language and a core of functional operators which are applicable to language's expressions. The language syntax is a simple extension of SWN arc function syntax.The operators are the ones required by most algorithms based on structural check of Colored PN. In particular the transpose ^{t}, the intersection $\cap$ and the difference $\ominus$ are needed to compute $EN(\mathbf{m})$. Each top-level operator has a low-level version acting on elementary function symbols: X_k, S, $S - X_k$, $S_{C_{i,j}}$ and $!^h X_k$.

The formal description of the language may be found in [4], here it is just overviewed. The language $\mathcal{L}$ is a set of expressions E_i closed with regard to the core $\mathcal{O}$ of (functional) operators. An expression $E \in \mathcal{L}$ is a formal sum of terms $[f_i] \circ T_i \circ [g_i]$ where $[g_i]$ and $[f_i]$ are SWN's guards and T_i is a SWN's function tuple with some more extension (the composition symbol $\circ$ is usually omitted). In order to satisfy the closure requirement under set $\mathcal{O}$, another extension has been necessary: class-functions may be linear combinations of *intersections* of elementary functions, so $[X_1 \neq X_2]\langle X_1 \cap !^2 X_2 + 3 \cdot (S - X_2), X_1\rangle$ belongs to $\mathcal{L}$. The extensions above make the SWN syntax richer from a descriptive point of view.

The subset composed by $E_i \in \mathcal{L}$ such that class functions are intersections of elementary symbols, is regarded as the kernel set and is denoted as $\mathcal{K}$. Equivalence between $\mathcal{K}$ and $\mathcal{L}$ is stated by:

Prop. 1: For each $E \in \mathcal{L}$ there exists $\{E'_k\} \subset \mathcal{K}$ such that $\sum_k E'_k \equiv E$

[4] introduces the rewriting rules translating E to E'. Without losing generality, we shall assume that $W^-(p,t)$ and $H(p,t)$ belong to $\mathcal{K}$.

A. Language Operators

Let $F, F' : A \rightarrow Bag(B)$. **Transpose** $(\cdot)^t$ The operator ^{t} applied to F results in $F^t : B \rightarrow Bag(A)$ such that $F^t(b)(a) = F(a)(b)\ \forall a \in A, \forall b \in B$ ($m(c)$ denotes the multiplicity of element c on multi-set m). Focusing on $W^-(t,p)$, given a p's color instance c, $W^-(t,p)^t(c)$ maps to the t's instances which need c on place p in order for being enabled, along with its quantity. For example a function $W^-(t,p)^t = \langle S - X_1, 2X_1\rangle$: $C_1 \rightarrow Bag(C_1 \times C_1)$, when evaluated in $c_1 \in C_1$ maps to multiset $\langle C_1 - c_1, 2.c_1\rangle = 2.\langle C_1 - c_1, c_1\rangle$. It means that t instances in $\langle C_1 - c_1, c_1\rangle$ require two tokens c_1 on place p to be enabled.

Intersection and Difference ($\cap$, $\ominus$) $(F \cap F')(a) = F(a) \cap F'(a)$ (recall that is m, m' belong to $Bag(C)$, then $\forall c \in C$,$(m \cap m')(c) = \min(m(c), m'(c))$). Functions F, F' are said disjoint if $F \cap F'$ is equal to the null (multi)set constant $\emptyset$. Analogously, $(F \ominus F')(a) = F(a) \ominus F'(a)$. When restricting to functions mapping on sets the $\ominus$ operator corresponds to the set-difference (it will be exclusively used in this context).

Linear extension (F^*) $F^* : Bag(C) \rightarrow Bag(D)$ is defined as $F^*(c+c') = F(c)+F(c')$. Abusing notation, we shall use symbol F to denote a function as well as its linear extension.

The symbolic framework algorithms operate rewritings on expressions in $\mathcal{K}$ that rely upon the language's algebraic properties [4]. Hereafter symbol $\longrightarrow$ will denote a rewriting application.

Prop. 2: Let $E \in \mathcal{K}$, $E = \sum_j \lambda_j \cdot f_j$, $\lambda_j \in \mathbb{N}$. Then: $E \equiv E'$, $E' = \sum_k \gamma_k \cdot g_k$ where $\forall k_1, k_2$: $g_{k_1} \cap g_{k_2} = \emptyset$

In other words any formal sum belonging to $\mathcal{K}$ (and consequently $\mathcal{L}$), can be rewritten into an equivalent formal sum of pairwise-disjoint terms belonging itself to $\mathcal{K}$. As an example, tuple $\langle 2S + X_1, X_2\rangle$ may be rewritten as $2\langle S - X_1, X_2\rangle + 3\langle X_1, X_2\rangle$.

IV. Efficiently Computing Enabled Instances

The computation of the color instances of transition t that are enabled in marking $\mathbf{m}$ ($EN(\mathbf{m}, t)$) is accomplished in modular way considering *locally* enabled sets. Given t and place p, a function $EN_{(t,p)} : Bag(\mathcal{C}(p)) \to \mathcal{P}(\mathcal{C}(t))$ is defined that, when applied to a given marking of place p, results in the exact set of color instances of t that are locally enabled in $\mathbf{m}(p)$, thereby denoted $EN_{(t,p)}(\mathbf{m}(p))$. Function $EN_{(t,p)}$ is formally expressed in terms of language $\mathcal{K}$. The set of color instances of transition t that are enabled in $\mathbf{m}$ is obtained by intersecting locally enabled sets: $EN(\mathbf{m}, t) = \bigcap_p EN_{(t,p)}(\mathbf{m}(p))$.

The technique will be illustrated by means of an example (Figure 2) in which the local enabling of a SWN transition depends on evaluating both an input and a inhibitor arc-function. Despite its simplicity the example is enough complete.

It is necessary to consider colored tokens occurring on $\mathbf{m}(p)$ and evaluate which instances of transition t they enable. Such information are contained in the transpose of functions $W^-(t,p)$ and $H(t,p)$. It is possible to build an enabling table, which enumerates for each $c'_i \in \mathcal{C}(t)$, and for each $c \in \mathcal{C}(p)$, an interval of positive integer values representing the number of occurrences of color c in p that ensure the local enabling of c'_i. The first step consists in finding compact representations for local enabling tables using transposition and difference operators on language $\mathcal{K}$.

.0.a The Tabular Symbolic Form. The symbolic framework provides efficient algorithms to compute the transposes of $W^-(t,p)$ and $H(t,p)$. The whole approach is based on rewriting both $W^-(t,p)^t$ and $H(t,p)^t$ in (also reciprocally) disjoint sums of terms (property 2). Let us consider the example on Fig. 2: $W^-(t,p_1)^t$ and $H(t,p_1)^t$ are functions $C_1 \to C_1 \times C_1$, operating the framework's transpose algorithm we obtain:

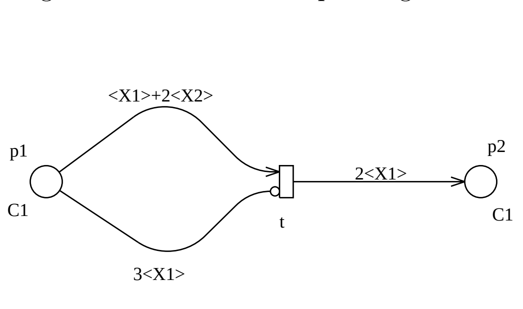

Fig. 2. A Simple SWN Model.

$\langle X_1\rangle^t + 2.\langle X_2\rangle^t \longrightarrow \langle X_1, S\rangle + 2.\langle S, X_1\rangle$
$3.\langle X_1\rangle^t \longrightarrow 3.\langle X_1, S\rangle$

Applying the disjoining steps [5] to $W^-(t,p_1)^t$ and using the rewriting rules for operator $\cap$, results in the following expression for $W^-(t,p_1)^t$:
$\langle X_1, S\rangle + 2.\langle S, X_1\rangle \longrightarrow$
$3.\big(\langle X_1, S\rangle \cap \langle S, X_1\rangle\big) + 1.\big(\langle X_1, S\rangle \cap \langle S, S - X_1\rangle\big) +$
$2.\big(\langle S, X_1\rangle \cap \langle S - X_1, S\rangle\big) \longrightarrow$
$3.\langle X_1, X_1\rangle + 1.\langle X_1, S - X_1\rangle + 2.\langle S - X_1, X_1\rangle$

As concerns $H(p_1,t)^t$, it is rewritten to:
$3.\langle X_1, S\rangle \to 3.\langle X_1, X_1\rangle + 3.\langle X_1, S - X_1\rangle$

At the end $W^-(t,p_1)^t$ and $H(t,p_1)^t$ take the expected forms, respectively:

$3.\langle X_1, X_1\rangle + 1.\langle X_1, S - X_1\rangle + 2.\langle S - X_1, X_1\rangle$
$3.\langle X_1, X_1\rangle + 3.\langle X_1, S - X_1\rangle$

These expressions may be folded into a single one $\mathcal{EN}_{(t,p_1)} : Bag(C) \to \mathcal{P}(\mathcal{C} \times \mathcal{C})$, which defines the form of locally and potentially enabled instances of t in a generic marking:

$$\lfloor 3, 2\rfloor\, \langle X_1, X_1\rangle \oplus \lfloor 1, 2\rfloor\, \langle X_1, S - X_1\rangle \oplus \lfloor 2, \infty\rfloor\, \langle S - X_1, X_1\rangle \qquad (1)$$

(*locally* means that we are only referring to place p_1). Each term T_k ($\in \mathcal{K}$) of $\mathcal{EN}_{(t,p_1)}$ is associated to an interval $[\alpha_k, \beta_k]$, $\alpha_k \in \mathbb{N}$, $\beta_k \in \mathbb{N} \cup \{\infty\}$, defining enabling bounds for t on p_1. $\mathcal{EN}_{(t,p_1)}$ provides a pattern: the first term says no instances $\langle c_1, c_1\rangle$ of t will be ever enabled. The second and third terms instead say that for an instance $\langle c_1, c_2\rangle$, $c_2 \neq c_1$, to be (locally) enabled, it is necessary and sufficient that color c_1 occurs at least once, but no more than twice, in p_1, while c_2 must occur at least twice. Symbols $\oplus$ and $\lfloor, \rfloor$ are used to avoid any confusion with the (multi-)set sum, and guards.

The computation of $\mathcal{EN}_{(t,p)}$ is required for each place p and transition t connected via an input/inhibitor arc. Even if potentially expensive, it has to be performed only once, then may be saved as structural information associated to a given SWN model.

Def. 1: Let $\mathcal{EN}_{(t,p)} : Bag(\mathcal{C}\ (p)) \to \mathcal{P}(\mathcal{C}\ (t))$ be the formal sum 1. Then $\mathcal{EN}_{(t,p)}(\mathbf{m}(p)) = \mathcal{EN}^+ / \mathcal{EN}^-$, where
$\mathcal{EN}^+ = \bigcup_{c \in \mathcal{C}(p), k:1\ldots n, \alpha_k \le \mathbf{m}(p)(c) \le \beta_k} T_k(c)$
$\mathcal{EN}^- = \bigcup_{c \in \mathcal{C}(p), k:1\ldots n, \alpha_k > \mathbf{m}(p)(c) \vee \beta_k < \mathbf{m}(p)(c)} T_k(c)$

Under the hypothesis (that we can always reconduce to via function rewriting) that $W^-(p,t)$ and $H(p,t)$ cannot be simultaneously annulled by any argument, the locally enabled set $EN_{(t,p)}(\mathbf{m}(p))$ is formally expressed by ($[g_t]$ is the transition guard): $[g_t]\mathcal{EN}_{(t,p)}(\mathbf{m}(p))$.

Intuitively speaking, if a color instance c' of t matches one term T_k on the formal sum, and the multiplicity of a colored token c in $p1$ such that $c' \in T_k(c)$ ranges over $[\alpha_k, \beta_k]$, then c' is potentially enabled. If instead the multiplicity of c in p_1 is out of the interval, then c' cannot be enabled.

Assuming $C = \{a, b, c, d, e\}$, let us consider the marking $\mathbf{m}(p_1)$: $3a + 2b + 1c$. According to Definition 1 $EN_{(t,p_1)}(\mathbf{m}(p_1))$ is:

$$(\langle b, S-b\rangle + \langle S-a, a\rangle + \langle S-b, b\rangle + \langle c, S-c\rangle) - (\langle a, S-a\rangle + \langle S-c, c\rangle + \textstyle\sum_{\forall x \notin \overline{\mathbf{m}(p_1)}}(\langle x, S-x\rangle + \langle S-x, x\rangle)) = \{\langle b,a\rangle, \langle c,a\rangle, \langle c,b\rangle\}$$

A. *Efficient Calculation of $EN_{(t,p)}$*

Definition 1 apparently forces one to consider all colors in the domain of p for computing the local enabled set of transition t at a given marking. The computation of $EN_{(t,p)}(\mathbf{m}(p))$ may be actually accomplished referring only at colors in $\overline{\mathbf{m}(p)}$ (the $^-$ operator applied to a multi-set results in its support-set).

The optimization proposed in [5] relies upon splitting symbolic computation in two steps, to better exploit information on the shape of the input arc function.

For simplicity we are making the hypothesis that $W^-(t,p)$, when evaluating different from ϕ, returns a constant number of tokens: $\forall c \in \mathcal{C}(t), W^-(t,p)(c) \neq \phi : |W^-(t,p)(c)| = n,\ n \in \mathbb{N}$. This assumption is not restrictive, as we can always fall into the hypothesis above (once again) via SWN arc function rewriting [5]. Then, to compute the local enabled instances of t in $\mathbf{m}(p)$ let us write a sum so defined: for each color $c' \in \overline{\mathbf{m}(p)}$ the expression $\mathcal{EN}'(c')$ is derived, that results from applying function (1) as it were a formal sum mapping on multisets, taking as coefficients lower-bounds α_k, and treating $\oplus$ as multiset sum; terms whose bounds are not satisfied by the multiplicity of c' in $\mathbf{m}(p)$ are deleted. The multiset $\mathcal{EN} = \sum_{c' \in \mathbf{m}(p)} \mathcal{EN}'(c')$ provides an expression for t's instances *potentially enabled* in $\mathbf{m}$ due to the marking of p: the instances of t requiring n tokens from p are the multiset elements having multiplicity n. They can be computed by intersecting such terms in $\mathcal{EN}$ whose multiplicity sum is n. In the running example $\mathcal{EN}$ is:

$$\underbrace{2\cdot\langle S-a, a\rangle}_{\mathcal{EN}'(a)} + \overbrace{(\langle b, S-b\rangle + 2\cdot\langle S-b, b\rangle)}^{\mathcal{EN}'(b)} + \underbrace{\langle c, S-c\rangle}_{\mathcal{EN}'(c)}$$

This expression highlights the contributions due to the different colors $\{a,b,c\}$ occurring on place p_1. Because $|\langle X_1 + 2.X_2\rangle| = 3$ the local enabled color instances of t are obtained intersecting:

(1) $\langle S-a, a\rangle \cap \langle b, S-b\rangle \to \langle b,a\rangle$;
(2) $\langle S-a, a\rangle \cap \langle c, S-c\rangle \to \langle c,a\rangle$;
(3) $\langle S-b, b\rangle \cap \langle c, S-c\rangle \to \langle c,b\rangle$.

Step 2 consists of taking into account residual terms in (1) such that $\alpha_k = 0$. The exact set of enabled color instances is obtained subtracting from color instances computed during step 1 the set:

$$\bigcup_{c \in \overline{\mathbf{m}(p)},k:\,\alpha_k=0 \wedge \mathbf{m}(p)(c) > \beta_k} T_k(c)$$

V. Extension to Symbolic Markings

The peculiar and most interesting feature of the SWN formalism is the ability of capturing system's behavioral symmetries thanks to the structured syntax of color annotations (Section II). In some sense such symmetries are encoded into the SWN color syntax. Efficient algorithms can be applied that exploit such symmetries to build an aggregated state space (called Symbolic Reachability Graph, or SRG [3]) and its corresponding stochastic process (a *lumped* CTMC), or to perform symbolic discrete-event simulation runs. These methods rely upon the notion of Symbolic Marking (SM) (and symbolic firing rule).

A SM provides a syntactical equivalence relation on ordinary markings: two markings belong to the same SM iff they can be obtained from one another by means of a *permutation* on unordered color classes that preserves static subclasses, and a *rotation* on ordered color classes.

.0.b Symbolic Marking. The definition of a SM $\widehat{\mathbf{m}}$ comprises two parts: one specifying the so called *dynamic subclasses* and another representing the distribution of colored symbolic tokens over the places.

Dynamic subclasses define a *parametric partition* of color classes preserving static subclasses: $\widehat{C_i}$ denotes the set of dynamic subclass of C_i (in $\widehat{\mathbf{m}}$). Let s_i be the number of static subclasses of color class C_i (if C_i is not split then $s_i = 1$). The j-th dynamic subclass $Z^i_j \in \widehat{C_i}$ refers to a static subclass, denoted $d(Z^i_j)$, $1 \leq d(Z^i_j) \leq s_i$, and has an associated cardinality $|Z^i_j|$ (i.e., it represents a parametric set of colors). It must hold, $\forall k : 1 \ldots s_i$

$$\textstyle\sum_{j:d(Z^i_j)=k} |Z^i_j| = |Ci_k|.$$

If Ci is an ordered class (we assume in that case $s_i = 1$), then its dynamic subclasses must be ordered accordingly, formally there is a homomorphism $h : \widehat{C_i} \to \widehat{C_i}$ such that $h^{|\widehat{C_i}|} = h$. Intuitively, contiguous dynamic subclasses represent contiguous (sets of) colors. If all dynamic subclass of an ordered class are of cardinality one then we prefer to use the notation ! instead of h.

The formal representation of $\widehat{\mathbf{m}}$ is a mapping $P \to Bag(\widehat{\mathcal{C}(p)})$, $\widehat{\mathcal{C}(p)}$ being the symbolic color domain of p obtained replacing each C_i in $\mathcal{C}(p)$ by $\widehat{C_i}$.

Among several possible (equivalent) definitions for a SM, two are of particular relevance. The *canonical representation* [2], that minimizes the number of dynamic subclasses and provides a unique formal representation. On the opposite there is the *split representation* of $\widehat{\mathbf{m}}$, where all dynamic subclasses are of cardinality one. Given a SM $\widehat{\mathbf{m}}$ its split representation is obtained replacing each Z^i_j, $|Z^i_j| > 1$, by $\sum_{k:1\ldots|Z^i_j|} Z^i_{j,k}$ (double subscripts denote dynamic subclasses deriving from splitting). Cardinality, static subclasses, and (possibly) ordering must be preserved by splitting operation.

Let us consider, once again, the net on Fig. 2: an example of SM is (place $p2$ is assumed empty, superscripts are omitted as there is only one color class, $\forall j,\ d(Z_j) = 1$):

$$\widehat{\mathbf{m}}(p1) :\ 3\langle Z_1\rangle + 2\langle Z_2\rangle + 1\langle Z_3\rangle$$
$$\widehat{C} = \{Z_j\}_{j:1\ldots4},\ |Z_1| = 2,\ |Z_2| = |Z_3| = |Z_4| = 1$$

An example of ordinary marking belonging to $\widehat{\mathbf{m}}$ is

$$\mathbf{m}'(p_1) :\ 3\langle a\rangle + 3\langle b\rangle + \langle 2c\rangle + 1\langle d\rangle.$$

The split representation of $\widehat{\mathbf{m}}$ is:

$$\mathit{split}(\widehat{\mathbf{m}})(p1):\; 3\langle Z_{1,1}\rangle + 3\langle Z_{1,2}\rangle + 2\langle Z_2\rangle + 1\langle Z_3\rangle$$
$$|Z_{1,1}| = |Z_{1,2}| = |Z_2| = |Z_3| = |Z_4| = 1$$

The SM notion is accompanied by that of symbolic color instance of a transition t. Let $\mathcal{C}(t)$ be the color domain of t and $\widehat{\mathbf{m}}$ be a split symbolic marking. The symbolic color domain of t respective to $\widehat{\mathbf{m}}$ is obtained replacing each C_i in $\mathcal{C}(t)$ by $\widehat{C_i}$ and is denoted $\widehat{\mathcal{C}(t)}$. A symbolic color instance of t on $\widehat{\mathbf{m}}$ is a tuple $\widehat{c} \in \widehat{\mathcal{C}(t)}$. If we consider symbolic color domains (related to a split SM) functions labeling an arc connecting p and t have arity $\widehat{\mathcal{C}(t)} \rightarrow Bag(\widehat{\mathcal{C}(p)})$ and are defined as in the ordinary case (see also Section II). For instance the application of function $\langle X_1, S-!X_2\rangle[X_1 \neq X_2]$ to the tuples $\langle Z_2, Z_1\rangle$ and $\langle Z_2, Z_2\rangle$ result in $\langle Z_2, S-Z_2\rangle$ and $\emptyset$, respectively (assuming $!Z_1 = Z_2$). The symbol $S-Z_2$ is a compact notation for "sum of all dynamic subclasses but Z_2", thereby $\langle Z_2, S-Z_2\rangle$ stands for $\sum_{j\neq 2}\langle Z_2, Z_j\rangle$.

Examples of symbolic color instances of t respective to the split SM above are $\langle Z_1, Z_1\rangle$, $\langle Z_2, Z_1\rangle$, $\langle Z_2, Z_{3,1}\rangle$. Note that some symbolic instances may be equivalent (i.e., they lead to the same SM), e.g. $\langle Z_2, Z_{3,1}\rangle \equiv \langle Z_2, Z_{3,2}\rangle$.

The notion of enabling of a color instance on a given marking (Section II) is thus straightforwardly extended to SM and symbolic transition instances. Considering the example above we can simply check that $\langle Z_2, Z_{3,1}\rangle$ is enabled. Testing the enabling of symbolic instances is an expensive operation for two main reasons: it forces the splitting of the source SM and does not take into account possible equivalences between symbolic instances.

.0.c Testing symbolic color instances. The basic formula (1), that the whole approach to symbolic enabling test relies upon (section VI), can be used to efficiently compute the set of enabled symbolic color instances of a transition in a given SM. The only difference is that it applies to (tuples of) dynamic subclasses instead of (tuples of) basic colors. Two conditions have to be meet in order for the optimization described in Section IV-A to be efficiently applied: 1) only (symbolic) colors occurring on a given SM representation should be considered, 2) the function-tuples composing the local enabling function should be evaluated as if they were mapping on multi-sets rather than on power-sets. That may be done without any difficulty as long as we consider a split representation of SM. The calculus manipulates such compact symbols as S,$S-Z^i_j$,.. that should be expanded at the end in order to obtain the enabled symbolic color instances.

Let us consider once again the running example. Applying the local enabling function to $\mathit{split}(\widehat{\mathbf{m}})$ (as explained in Section IV-A) results in the multiset

$$\underbrace{2\langle S-Z_{1,1}, Z_{1,1}\rangle + 2\langle S-Z_{1,2}, Z_{1,2}\rangle}_{Z_1} +$$
$$\underbrace{1\langle Z_2, S-Z_2\rangle + 2\langle S-Z_2, Z_2\rangle}_{Z_2} +$$
$$\underbrace{1\langle Z_3, S\rangle}_{Z_3}$$

The expression above denotes a multiset on $\widehat{C_1} \times \widehat{C_1}$, the symbolic color domain of transition t. Terms are on the form of tuples, each representing a set of symbolic color instances. Terms referring to the same color (of the symbolic color domain $\widehat{C_1}$ of place p) are pairwise-disjoint by construction. Locally enabled symbolic color instances of t correspond to multiset elements of multiplicity 3, that are obtained by intersecting (non disjoint) terms whose coefficient sum is 3.

$\langle S-Z_{1,1}, Z_{1,1}\rangle \cap \langle Z_2, S-Z_2\rangle$,
$\langle S-Z_{1,2}, Z_{1,2}\rangle \cap \langle Z_2, S-Z_2\rangle$,
$\langle S-Z_{1,1}, Z_{1,1}\rangle \cap \langle Z_3, S\rangle$,
$\langle S-Z_{1,2}, Z_{1,2}\rangle \cap \langle Z_3, S\rangle$,
$\langle S-Z_2, Z_2\rangle \cap \langle Z_3, S\rangle$

Using the intersection rules we obtain (no expansion is required in this case):

$$\{\underbrace{\langle Z_2, Z_{1,1}\rangle, \langle Z_2, Z_{1,2}\rangle}_{\equiv}, \underbrace{\langle Z_3, Z_{1,1}\rangle\ , \langle Z_3, Z_{1,2}\rangle}_{\equiv}, \langle Z_3, Z_2\rangle$$

The 1st and 2nd, and the 3rd and 4th instances are syntactically recognizable as equivalent to one another, so only one instance of each pair has to be processed for firing.

The split SM representation allows the evaluation of local enabling functions (expressed in terms of language $\mathcal{K}$) on symbolic color domains to be accomplished as in the ordinary case. In the example above splitting dynamic subclass Z_1 has been required to correctly evaluates function-tuples on which projection symbol X_1 is repeated. The other possible splitting causes are 2) repetition of a dynamic subclass on a tuple argument of a function-tuple, 3) occurrence of projection successor symbol on a function-tuple.

Very often however only a partial (possibly null) splitting of a SM $\widehat{\mathbf{m}}$, assumed in canonical form, is necessary. Avoiding splitting as long as possible makes faster computation of enabled symbolic instances as well as recognition of equivalent instances. How that can be achieved is informally described through an application example.

VI. An Application Example

In this section we present an example of application of the symbolic enabling test using as case-study the workflow SWN model of document reviewing process depicted in Figure 1. A number of documents are being reviewed by a team of reviewers at a given organization. The number of documents simultaneously reviewed and the size of the reviewer team are the model's parameters. After sending a copy of each document to be reviewed to the team the revision process begins. Each reviewer can simultaneously work on many different documents, and (for simplicity) that only two judgments are possible, accept or not accept. It the reviewing process by a single reviewer takes too much time that particular revision should not be considered for the final decision. The decision process about a given document starts only once all the reviewers have completed their single revision on that document in the due time, or they have exceeded the timeout fixed for revision. The decision algorithm is quite simple: the decision about a given document may be considered valid if and only if at most two reviewers did not complete their revisions in the due time. In case of valid decision, a document is approved (accepted) if the "accept" are more than the "not-accept".

The main activities of the reviewers and the timeout are represented by timed transitions, while the decision algorithm is modeled by a subnet of immediate transitions. Examples of performance figures that could be computed with steady-state analysis/simulation are the percentage of acceptance/rejection per document, the percentage of valid decisions, the throughput of the revision process (i.e. the number of documents on which a valid judgment is expressed per time unit), and so on.

The focus here is on the complexity of both exact solution and (discrete/event) simulation, and the potential computational gain provided by the technique described in Section IV to simulation efficiency. Despite the relative complexity of model's color annotations we were able to analyze/simulate a reduced number of configurations using the algorithms implemented in the *GreatSPN* package. Table I shows the state-space growing for increasingly complex configurations (all documents are initially on place P_1). Looking at the data on the table we can argue that despite the significant reduction due to symmetries only configurations with a small number of documents ($n_d \leq 3$, $n_d{\cdot}n_r \leq 12$, $n_d = |DOC|$, $n_r = |REVIEWER|$) can be analytically solved. The 3×4 configuration represents a threshold for the (symbolic) discrete-event simulation engine of *GreatSPN* (running on a Pentium 4 630 with 2 GByte of RAM), what makes highly unreliable trying to infer tendency curves on realistic working scenarios.

TABLE I: State-space growing

(n_d,n_r)	$\|SRG\|$	$\|RG\|$
(2,2)	465	1664
(2,3)	3430	34515
(3,2)	5177	56099
(2,4)	20425	731120
(2,5)	103457	15823647
(3,3)	162892	5524164
(2,6)	463513	254974732
(3,4)	$>$ 5000000	-

Unfortunately, the form of model's arc functions is such that neither the technique defined in [6] nor that one defined in [8] can be used to improve simulation efficiency: more precisely the presence of complementary functions, constant functions and guards avoids exploiting any known optimization. On the other side, the *structural* approach for symbolic enabling test defined in section IV, whose complexity does not depend on color class cardinality and that applies to *any* kind of SWN functions, might be conveniently used to study even more complex model's configurations than that ones considered on table I. That approach allows most of complexity of enabling test task to be moved from run-time to syntactical manipulation of arc functions (performed only once for a given model). This technique can be naturally integrated to the symbolic SWN transition firing rule in order to operate an optimized symbolic (discrete/event) simulation of SWN models.

.0.d Symbolic Enabling Test. Let us apply the symbolic enabling test to a selection of transitions (*get_review*,*decision_invalid*,*no_accept*), on a given symbolic marking. Both color classes C_1: *DOC*, and C_2: *REVIEWER* are not split (C_i will be used instead of $\widehat{C_i}$ hereafter).

Assuming $n_d = n_r = 3$, let us first consider the enabling of transition *get_review* ($\mathcal{C}(get\_review)$: $C_1 \times C_2 \times C_3$) on the SM $\widehat{\mathbf{m}}$ below, respective to place P_3 ($\mathcal{C}(P_3)$: $C_1 \times C_2$):

$$\widehat{\mathbf{m}}(P_3): \quad \langle Z_1^1, Z_1^2\rangle \quad |Z_1^1| = 1,\, |Z_1^2| = 2$$

An ordinary marking belonging to $\widehat{\mathbf{m}}$ is

$$\mathbf{m}(P_3): \quad \langle d_1, r_1\rangle + \langle d_1, r_2\rangle$$

According to the transposing rules we obtain ($C_{3,3}$: *timeout*; only the bag support is hereafter indicated in the co-domain of functions):

$$\begin{aligned} W^-(get\_review, P_3)^t \quad &: \quad C_1 \times C_2 \rightarrow C_1 \times C_2 \times C_3 \\ &= \quad [d(X_3) <> C_{3,3}]\langle X_1, X_2, S\rangle \end{aligned}$$

The formal expression of the local enabling function turns out to be, after some rewriting ($C_{3,1}$: *accept*, $C_{3,2}$: *noaccept*):

$$\begin{array}{l} \lfloor 1,\infty\rfloor\, [d(X_3) <> C_{3,3}]\langle X_1, X_2, S\rangle \quad \equiv \\ \lfloor 1,\infty\rfloor\, \langle X_1, X_2, S_{3,1} + S_{3,2}\rangle \end{array}$$

Applying directly the procedure described in section IV-A without any preliminary splitting we obtain the formal multiset:

$$\langle Z_1^1, Z_1^2, S_{3,1}\rangle + \langle Z_1^1, Z_1^2, S_{3,2}\rangle$$

Both terms occur with multiplicity 1 (which is the cardinality of the input function application), moreover they are disjoint, so they represent the locally enabled symbolic color instances of transition *get_review*. Due to the cardinality of dynamic subclass Z_1^2 each of them represents two ordinary color instances.

Let us consider now transition *decision_invalid* ($\mathcal{C}(decision\_invalid)$: $C_1 \times C_2 \times C_3$), focusing on its local enabling respective to place P_5 ($\mathcal{C}(P_5)$: $C_1 \times C_2 \times C_3$):

$$\begin{array}{ll} W^-(decision\_invalid, P_5)^t & : \\ C_1 \times C_2 \times C_3 \rightarrow C_1 \times C_2 \times C_3 & = \\ \langle X_1, S - X_2, S_{3,3}\rangle^t + \langle X_1, X_2, X_3\rangle^t & \equiv \\ \langle X_1, S - X_2, S\rangle[d(X_3) = C_{3,3}] + \langle X_1, X_2, X_3\rangle & \end{array}$$

The formal expression 1 turns out to be in this case:

$$\begin{array}{l} \lfloor 1,\infty\rfloor\, \langle X_1, S - X_2, S\rangle[d(X_3) = C_{3,3}] \quad \oplus \\ \lfloor 1,\infty\rfloor\, \langle X_1, X_2, X_3\rangle \end{array}$$

Consider now symbolic marking $\widehat{\mathbf{m}}'$, that describes a situation where the revision process of a document (Z_1^1) has finished but cannot be considered valid because two (out of three) reviewers did not complete their work in the due time (while the third one approved the document), and the revision of document Z_2^1 is still incomplete (there is only an approval by one reviewer):

$$\begin{aligned}\widehat{\mathbf{m}}'(P_5): &\quad \langle Z_1^1, Z_1^2, S_{3,1}\rangle + \langle Z_1^1, Z_2^2, S_{3,3}\rangle + \\ &\quad +\langle Z_2^1, Z_1^2, S_{3,1}\rangle \\ \widehat{\mathbf{m}}'(P_6): &\quad \langle Z_1^1\rangle + \langle Z_2^1\rangle \\ &\quad |Z_1^1| = |Z_2^1| = |Z_1^2| = 1 \;\; |Z_2^2| = 2\end{aligned}$$

Applying the local enabling function directly to $\widehat{\mathbf{m}}'$, as described in section IV-A, results in the formal multiset:

$$\begin{aligned}&2\cdot\langle Z_1^1, Z_1^2, S\rangle + \langle Z_1^1, Z_2^2, S\rangle + \\ &\langle Z_1^1, Z_1^2, S_{3,1}\rangle + \langle Z_1^1, Z_2^2, S_{3,3}\rangle + \langle Z_2^1, Z_1^2, S_{3,1}\rangle\end{aligned}$$

The first row of the formal sum results from applying the first tuple of the local enabling function to the marking. According to the function linear extension, $(S - X_2)(Z_2^2)$ results in $(S - Z_{2,1}^2) + (S - Z_{2,2}^2)$, $|Z_{2,i}^2| = 1$, that is, $2 \cdot Z_1^2 + Z_2^2$. Searching for locally enabled symbolic instances corresponds to finding multiset terms having cardinality 3 (the cardinality of $W^-(decision\_invalid, P_5)$ application): it straightforwardly comes out, by intersecting non disjoint multiset terms whose multiplicity sum is equal to 3:

$$\langle Z_1^1, Z_1^2, S\rangle \cap \langle Z_1^1, Z_1^2, S_{3,1}\rangle \quad\equiv\quad \langle Z_1^1, Z_1^2, S_{3,1}\rangle$$

With very simple arguments we might convince ourselves that locally enabled symbolic instances of *decision_invalid* respective to place P_6 are expressed by the (formal) set $\langle Z_1^1, S, S\rangle + \langle Z_2^1, S, S\rangle$, so the only enabled symbolic instance in $\widehat{\mathbf{m}}'$ (obtained by intersecting the locally enabled sets) is $\langle Z_1^1, Z_1^2, S_{3,1}\rangle$.

The last case we consider is the local enabling of transition *no_accept* ($\mathcal{C}(no\_accept)$: C_1) respective to place P_5 on marking $\widehat{\mathbf{m}}''$ (this marking is reached after symbolic color instance $\langle Z_1^1, Z_1^2, S_{3,1}\rangle$ of *decision_invalid* on $\widehat{\mathbf{m}}'$ has fired)

$$\begin{aligned}\widehat{\mathbf{m}}''(P_5): &\quad \langle Z_1^1, Z_1^2, S_{3,1}\rangle \\ \widehat{\mathbf{m}}''(P_6): &\quad \langle Z_1^1\rangle \\ &\quad |Z_1^1| = |Z_1^2| = 1\end{aligned}$$

The local enabling of transition *no_accept* only depends on the function labeling the inhibitor arc between *no_accept* and place P_5. Transposing that function results in:

$$\begin{aligned}H(no\_accept, P_5)^t \quad &: \quad C_1 \times C_2 \times C_3 \to C_1 \\ &= \quad \langle X_1\rangle[d(X_3) = C_{3,1}]\end{aligned}$$

The local enabling function takes the form below:

$$\lfloor 0, 0\rfloor\, \langle X_1\rangle[d(X_3) = C_{3,1}]$$

For the absence of the input arc between *no_accept* and place P_5, we only need to apply step 2 of the procedure described in section IV-A in order to obtain the locally enabled color instances of transition *no_accept*, i.e., we have to consider the complementary of the of the local enabling function application to colors occurring on the marking:

$$\langle S - Z_1^1\rangle \quad\equiv\quad \langle Z_2^1\rangle \;\; |Z_2^1| = 2$$

Since the locally enabled set of transition *no_accept* with respect to its input place P_6 on SM $\widehat{\mathbf{m}}''$ is trivially equal to $\langle Z_1^1\rangle$, the set of enabled symbolic instances of *no_accept* on $\widehat{\mathbf{m}}''$ shows to be (dynamic subclasses are pairwise disjoint):

$$\langle Z_1^1\rangle \cap \langle Z_2^1\rangle \quad\equiv\quad \emptyset$$

VII. Conclusion

This paper illustrates how the recently developed algebraic approach to SWN enabling test may be easily integrated to the SWN symbolic firing rule (that exploits behavioral symmetries) to give rise to a fully algebraic discrete-event simulation engine. An application example to a SWN workflow model has been presented showing the benefits of the new algebraic simulation engine in terms of efficiency. The outcome make us confident in a consistent reduction of the computational overhead due to enabling test, that affects all techniques based on state-space exploration. Current and future works are in two directions: (1) producing a full implementation of the algorithm (2) using SWN structural relations such as conflict and mutual exclusion to further improve the enabling-test task.

References

[1] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Volume 1, Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag. ISBN: 3-540-60943-1., 1997.

[2] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, “Stochastic well-formed coloured nets for symmetric modelling applications,” *IEEE TC*, vol. 42(11), no. 11, pp. 1343–1360, 1993.

[3] ——, “A symbolic reachability graph for coloured petri nets,” *Theoretical Computer Science B (Logic, semantics and theory of programming)*, vol. 176, no. 1&2, pp. 39–65, 1997.

[4] L. Capra, M. De Pierro, and G. Franceschinis, “A High Level Language for Structural Relations in Well-Formed Nets,” in *Proceeding of the 26th International Conference on Application and Theory of Petri Nets*, ser. LNCS 3536, G. Ciardo and P. Darondeau, Eds., Miami, USA, June 2005, pp. 168–187.

[5] L. Capra and M. De Pierro, “Efficient enabling Test in Simulation of SWN,” in *Proceeding of the 20th annual European Simulation and Modelling Conference (ESM’2006).* Toulouse, Fra: EUROSIS-ETI, Oct. 2006.

[6] R. Gaeta, “Efficient discrete-event simulation of colored petri nets,” *IEEE TSE*, vol. 22(9), no. 9, pp. 629–639, September 1996.

[7] K. Jensen and G. Rozenberg, Eds., *High-Level Petri Nets. Theory and Application.* Springer Verlag, 1991.

[8] Jean-Michel Ilie and Omar Rojas, “On well-formed nets and optimizations in enabling test,” in *Application and Theory of Petri Nets 1993, LNCS*, vol. 691. Springer-Verlag, 1993, pp. 300–318.