On the use of exact lumpability in partially symmetrical Well-formed Nets

S. Baarir, C. Dutheillet
LIP6, Université Paris 6
8 rue du Capitaine Scott
F-75015 Paris
Souheib.Baarir@lip6.fr
Claude.Dutheillet@lip6.fr

S. Haddad
LAMSADE, Université Paris 9
Place du Mal De Lattre de Tassigny
F-75016 Paris
haddad@lamsade.dauphine.fr

J.M. Ilié
LIP6 & IUT Paris5
143, av. de Versailles
F-75016 Paris
Jean-Michel.Ilie@lip6.fr

Abstract

Well-formed Nets (WNs) have proved an efficient model for building quotient reachability graphs that can be used either for qualitative or performance analysis. However, local asymmetries often break any possibility of grouping states into classes, thus drastically reducing the interest of the approach. An efficient solution has been proposed for qualitative analysis, which relies on a separate representation of the asymmetries in a so-called control automaton. The quotient graph is then obtained by synchronising the transitions of the WN model with the transitions of the control automaton. In this paper, we improve this approach to quantitative analysis. We show that it can be used to build an aggregated graph that is isomorphic to a Markov chain which verifies exact lumpability. Theoretical considerations and practical experiments show that our method outperforms previous approaches.

1 Introduction

Continuous Time Markov Chains (CTMC) are a popular model for evaluating the performance of distributed systems. However, as the complexity of systems increases, the fighting of the so-called combinatorial explosion of the state space becomes more and more critical. A possible approach to tackle this problem is the construction of a *lumped CTMC* using behavioural symmetries. Such a CTMC offers a compact representation of the state space because its nodes are no longer states but classes of states.

Whenever the distributed system is specified by a Well-formed Net (WN) with their restricted syntax (w.r.t. general coloured Petri nets), a symmetry-based quotient structure called Symbolic Reachability Graph (SRG) can be computed automatically. On this reduced graph, one solves the reachability problem and more generally the truth of temporal logic formulae whenever the atomic propositions of the formula are symmetrical (e.g. $\mathbf{AND}_{c \in C(p)} m(p)(c) = 1$). [5] establishes the correctness of this checking in a general framework. Moreover, since the SRG verifies *strong and exact lumpability* criterions, a lumped CTMC is automatically derived from it and any performance result obtained by solving the (usually much larger) complete CTMC, is also computed from the lumped one [4]. The SRG technique works well on highly symmetrical systems. However, in the practice of distributed systems, it is often the case that a system behaves in a symmetric way in most situations, *but not all*. Any occurrence of asymmetry, even exceptional, reduces drastically the benefices of this approach.

Many approaches were proposed to study such *asymmetrical systems*. In [6], the authors propose to adapt the symmetry rule in accordance with the system specification, to view some groups of almost symmetrical states as symmetric. To our knowledge, there is no tool to automate this task thus limiting the practical interest of the approach, moreover no quantitative extension exists.

An other technique named ESRG was proposed in [8], as an extension of the SRG technique. It consists in restraining the symmetries but only on the nodes from which the effects of asymmetric events must be considered. This leads to a reduction in the number of nodes, because one node of the ESRG can represent several classes explicitly represented in the SRG. The ESRG technique is still automatic from the well-formed net specification, and some practical studies show that the constructed CTMC remains compact, in particular whether the refinement operation over an equivalence class has no side effect [9]. Nevertheless, the computation of the lumped CTMC is not direct [3] : the building of the ESRG is the starting point from which a refinement process is performed leading to a partial unfolding of nodes up to verify a strong lumpability criteria. Furthermore, the exact probabilities of states cannot be expressed since the chosen criteria does not guaranty the equiprobability of states lumped under the same node. An additional

extension was proposed in [2] named E^{2}SRG. Although there is no current implementation, the first case studies show than it can be more compact than the ESRG, anyway, it is used to solve reachability problems and the adaptation required to obtain a lumped CTMC is still expected.

In this paper, we propose a new symbolic method for building a lumped CTMC automatically and directly. It is based on a alternative construction, primitively used to check the truth of LTL temporal properties for asymmetric systems [1, 7]. Hence, the system is defined as a synchronized product of models: a symmetric system and an *event-based* automaton to model the symmetric behaviour compactly. Symbolic operations are defined in order to split or group symbolic nodes, on-the-fly, during the computation of successors. This allows to adapt, dynamically and locally, the available symmetries for each reachable node. We propose to reuse such an approach for performance purposes, moreover an *exact lumpability* criterion is used to compute the lumped CTMC. Hence, state probabilities can be computed.

The schedule of this paper is the following: section 2 introduces the principles of our method, introducing the notion of *Partially Symmetrical CTMC*; section 3 is our application to WNs to obtain an automatic performance analysis tool; section 4 considers a use case extracted from the literature on which we show the benefit of our construction over the RG approach; section 5 contains our conclusions and perspectives.

2 Principles of the Generic Method

2.1 Markov Chains and Lumpability

Lumping of (finite) Markov chains is a useful method for dealing with large chains [10]. The principle is simple: substitute to the Markov chain an "equivalent" one, where each state of the lumped chain is an equivalence class of states of the original one. There are different versions of lumpability related to the fact that the lumpability condition holds for every initial distribution (*strong lumpability*) or for at least one (*weak lumpability*).

Definition 1 *A CTMC $\mathcal{C}$ is defined by a space set S, an infinitesimal generator Q, and π_0, an initial probability distribution over S. We note $\{X_t\}_{t\in\mathbb{R}^+}$ the associated stochastic process. Let $\mathcal{C}$ be a CTMC and $\{S_i\}_{i\in I}$ be a partition of the state space. Let Y_t be a random variable defined by $Y_t = i \Leftrightarrow X_t \in S_i$. Then:*

- *Q is strongly lumpable w.r.t. $\{S_i\}_{i\in I}$ iff $\forall \pi_0$, $\{Y_t\}_{t\in\mathbb{R}^+}$ is a CTMC,*
- *Q is weakly lumpable w.r.t. $\{S_i\}_{i\in I}$ iff $\exists \pi_0$ s.t. $\{Y_t\}_{t\in\mathbb{R}^+}$ is a CTMC.*

Whereas the characterisation of strong lumpability w.r.t. the infinitesimal generator is straightforward, checking for weak lumpability is much harder. Nevertheless, there is a particular case of weak lumpability whose characterisation is easy: the *exact lumpability* [12].

Proposition 2 *Let $\mathcal{C}$ be a CTMC and $\{S_i\}_{i\in I}$ be a partition of the state space. Then:*

- *Q is strongly lumpable w.r.t. $\{S_i\}_{i\in I}$ iff $\forall i \neq j \in I, \forall s, s' \in S_i, \sum_{s''\in S_j} Q(s, s'') = \sum_{s''\in S_j} Q(s', s'')$*
- *Q is exactly lumpable w.r.t. $\{S_i\}_{i\in I}$ iff $\forall i \neq j \in I, \forall s, s' \in S_i, \sum_{s''\in S_j} Q(s'', s) = \sum_{s''\in S_j} Q(s'', s')$. If Q is exactly lumpable w.r.t. $\{S_i\}_{i\in I}$, then Q is weakly lumpable w.r.t. $\{S_i\}_{i\in I}$.*

Furthermore, exact lumpability fulfills important properties. As for strong lumpability, the infinitesimal generator of the lumped chain is directly computed from the original generator. Starting with a distribution equidistributed on the states of every subset of the partition, the distribution at any time is still equidistributed. Consequently, if the CTMC is ergodic, its steady-state distribution is equidistributed. In other words, with the knowledge of the lumped chain generator, one may compute its steady-state distribution, and deduce (by equidistribution) the steady-state distribution of the original chain. It must be emphasised that this last step is impossible with strong lumpability. The next proposition summarises these results.

Proposition 3 *Let $\mathcal{C}$ be a CTMC that is exactly lumpable w.r.t. a partition of the state space $\{S_i\}_{i\in I}$. Let Q^{lp} be the matrix associated to this lumped CTMC, then:*

- *$\forall i, j \in I, \forall s \in S_j, Q^{lp}(i, j) = (\sum_{s'\in S_i} Q(s', s)) \times (|S_j|/|S_i|)$*
- *If $\forall i \in I, \forall s, s' \in S_i, \pi_0(s) = \pi_0(s')$ then $\forall t, \forall i \in I, \forall s, s' \in S_i, \pi_t(s) = \pi_t(s')$, where π_t is the probability distribution at time t.*
- *If Q is ergodic and π is its steady-state distribution then $\forall i \in I, \forall s, s' \in S_i, \pi(s) = \pi(s')$*

2.2 A Model of Partially Symmetrical CTMCs

The model of partially symmetrical systems that we develop here is defined as a CTMC obtained by some synchronised product between a (symmetrical) CTMC and a control automaton. Let us first formalise this product. Synchronising the behaviour of the two components requires to "label" the CTMC with events.

Notation Let $\mathcal{C}$ be a CTMC, we associate with each pair of states $s \neq s'$ a label in some alphabet Σ, denoted $\Lambda(s, s')$.

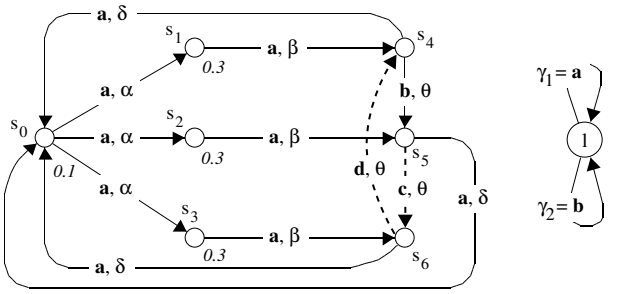

Figure 1. A labeled CTMC and its control automaton

Since the automaton is introduced in order to modify the behaviour of the CTMC, the label of each edge is a predicate that selects the events allowed to occur in the current location of the automaton.

Definition 4 *Let $\mathcal{C}$ be a CTMC, then $\mathcal{A} = \langle L, l_0, \rightarrow \rangle$ a control automaton of $\mathcal{C}$ is defined by:*

- *L, the set of automaton locations,*
- *l_0, the initial location,*
- *$\rightarrow \subseteq L \times 2^{\Sigma} \times L$ the transitions of the automaton. A transition (l, γ, l') will be denoted by $l \xrightarrow{\gamma} l'$.*

Fig.1 represents a CTMC and its control automaton. Bold letters are labels, while greek letters represent transition rates. The numbers associated with states are their initial probabilities.

In the synchronised product defined below, the CTMC is the "active" component whereas the automaton inhibits some behaviours of the product. Consequently, the rates (resp. the initial distribution) associated with the product depends only on the rates (resp. the initial distribution) of the CTMC.

Definition 5 *Let $\mathcal{C}$ be a CTMC and $\mathcal{A}$ some control automaton of $\mathcal{C}$. $\mathcal{C}_{\mathcal{A}} = \langle S \times L, \pi'_0, Q' \rangle$ is a CTMC defined by:*

- $\forall s, \pi'_0(s, l_0) = \pi_0(s) \wedge \forall l \neq l_0, \pi'_0(s, l) = 0$
- $\forall s \neq s' \in S, \forall l, l' \in L$, *if* $l \xrightarrow{\gamma} l' \wedge \Lambda(s, s') \in \gamma$ *then* $Q'((s, l), (s', l')) = Q(s, s')$ *else* $Q'((s, l), (s', l')) = 0$
- $\forall s \in S, \forall l \neq l' \in L, Q'((s, l), (s, l')) = 0$

In the example of Fig.1, the control automaton actually forbids transitions that are not labelled with a or b. Hence, $\mathcal{C}_{\mathcal{A}}$ is obtained from $\mathcal{C}$ by removing the dotted arcs. Formally, the states of $\mathcal{C}_{\mathcal{A}}$ are pairs (s_i, l) but as there is only one location in the automaton, we will omit it in the representation of states throughout the example.

From a theoretical point of view, the specification of the system symmetries relies on group theory, applied to the states and the events of the system. The next definition recalls the appropriate notions.

Definition 6 *Let G be a group, with neutral element id and whose internal operation is denoted ($\bullet$).*

- *Let E be a set, an* operation *of G on E is a mapping from $G \times E$ to E s.t. the image of (g, e), denoted by $g.e$, fulfills:* $\forall e \in E \; id.e = e$
  $\forall g, g' \in G, \; (g \bullet g').e = g.(g'.e)$
- *The* isotropy subgroup *of a subset $E' \subseteq E$ is defined by:* $G_{E'} = \{g \in G \mid \forall e \in E', g.e \in E'\}$
- *Let H be a subgroup of G, the* orbit *of e by H denoted $H.e$, is defined by:* $\{g.e \mid g \in H\}$.
  The set of orbits by H defines a partition of E.

We simultaneously introduce the notions of symmetrical and partially symmetrical CTMC. Informally, a CTMC is *symmetrical* w.r.t. some group if the operation of the group on the state space preserves its initial distribution and stochastic behaviour. A CTMC is *partially symmetrical* if it is a synchronised product involving a symmetrical CTMC.

Definition 7 *A CTMC $\mathcal{C}$ is symmetrical w.r.t. G a group operating on S and Σ iff:* $\forall g \in G, \forall s \neq s' \in S, \pi_0(g.s) = \pi_0(s) \wedge Q(g.s, g.s') = Q(s, s')$ *and* $\Lambda(g.s, g.s') = g.\Lambda(s, s')$.

Let $\mathcal{C}$ be symmetrical w.r.t. G and $\mathcal{A}$ be a control automaton of $\mathcal{C}$, then $\mathcal{C}_{\mathcal{A}}$ is said to be partially symmetrical w.r.t. G.

We associate with each γ occurring in a transition of $\mathcal{A}$ a subgroup $H_\gamma \subseteq G$ defined by: $g \in H_\gamma$ iff $\forall a \in \Sigma, a \in \gamma \Leftrightarrow g.a \in \gamma$.

The size of the subgroup H_γ is an indicator of the symmetry of the associated edge. When $H_\gamma = G$, the edge is "fully" symmetrical whilst when $H_\gamma = \{id\}$, the edge is "fully" asymmetrical.

Back to the example of Fig.1, let $G = \{id, r, r \bullet r\}$, where r is defined by :

$r.s_0 = s_0 \quad r.s_1 = s_2 \quad r.s_2 = s_3 \quad r.s_3 = s_1$
$r.s_4 = s_5 \quad r.s_5 = s_6 \quad r.s_6 = s_4$
$r.a = a \quad r.b = c \quad r.c = d \quad r.d = b$

It is easy to verify that G is a group and that the CTMC is symmetrical w.r.t. G. The subgroups associated with the labels of $\mathcal{A}$ are $H_{\gamma_1} = G$ and $H_{\gamma_2} = \{id\}$.

2.3 Partially Symmetrical CTMCs and Lumpability

Given a partially symmetrical CTMC $\mathcal{C}_\mathcal{A}$, our method builds a smaller (but equivalent) CTMC. However, in order to prove the soundness of this construction, we first introduce a CTMC $\mathcal{C}_\mathcal{A}^G$, which is actually bigger than $\mathcal{C}_\mathcal{A}$.

In $\mathcal{C}_\mathcal{A}^G$, states of $\mathcal{C}_\mathcal{A}$ are replicated in instances, and instances are organised in subsets. All the instances that belong to the same subset must have the same associated location of the automaton. We will thus consider subsets R of states of the initial CTMC $\mathcal{C}$, and denote (s, l, R) the instance of (s, l) s.t. s belongs to R.

Intuitively, given two states (s, l, R) and (s', l, R) of $\mathcal{C}_\mathcal{A}^G$, any path leading to (s, l, R) may be transformed by the operation of some element of G into a path to (s', l, R).

Definition 8 *Let $\mathcal{C}_\mathcal{A}$ be partially symmetrical w.r.t. G, then the CTMC $\mathcal{C}_\mathcal{A}^G = \langle S'', \pi_0'', Q'' \rangle$ is inductively defined by:*

- *The set of states S'' is a union of subsets of items defined from a set $R \subseteq S$ and a location l by $\{(s, l, R) \mid s \in R\}$,*
- $\forall s \in S, \forall l \in L, \forall R \subseteq S$,
  if ($l = l_0 \wedge R$ is an orbit by $G \wedge s \in R$) then
  $\pi_0''(s, l, R) = \pi_0'(s, l_0)$ $(= \pi_0(s))$
  else $\pi_0''(s, l, R) = 0$,
- *The "initial" subsets of states are $\{(s, l, R)\}$ s.t. R is the orbit of s by $G \wedge \pi_0''(s, l, R) > 0$,*
- *If $\{(s, l, R)\}$ is a subset of states and $\exists s^* \in R, \exists s'^* \in S, \exists l \xrightarrow{\gamma} l' \wedge \Lambda(s^*, s'^*) \in \gamma$ then the subset $\{(s', l', R')\}$ with $R' = (G_R \cap H_\gamma).s'^*$ is another subset of states,*
- $\forall g \in G_R \cap H_\gamma$, *let* $s = g.s^*$ *and* $s' = g.s'^*$ *then* $Q''((s, l, R), (s', l', R')) = Q(s, s')$

Remarks

1. Let $s = g.s^*$ and $s' = g.s'^*$, since $Q(s, s') = Q(g.s^*, g.s'^*) = Q(s^*, s'^*)$, the transition rate does not depend on the chosen pair.
2. Furthermore the above subset construction does not depend on the choice of s^* and s'^* in the following sense. Let us pick some $s' \in (G_R \cap H_\gamma).s'^*$, thus $s' = g.s'^*$ with $g \in G_R \cap H_\gamma$. Define $s = g.s^*$, then $s \in R (\supseteq G_R.s^*)$ and $\Lambda(s, s') \in \gamma$. Now it is routine to show that $(G_R \cap H_\gamma).s' = (G_R \cap H_\gamma).s'^*$.

Fig.2 describes CTMC $\mathcal{C}_\mathcal{A}^G$ for our example. Dotted rectangles represent the subsets of states. The initial subsets are those with associated orbits $S_0 = \{s_0\}$ and $S_{123} = \{s_1, s_2, s_3\}$. A group is associated with each subset: the group is G for the initial subsets and remains G for the constructed subsets until there is a synchronisation with γ_2. At that moment, the group is reduced to identity by intersection with H_{γ_2} and the constructed subset contains a single state. As a consequence, there are two instances of s_5 in the resulting CTMC, each with a different associated orbit.

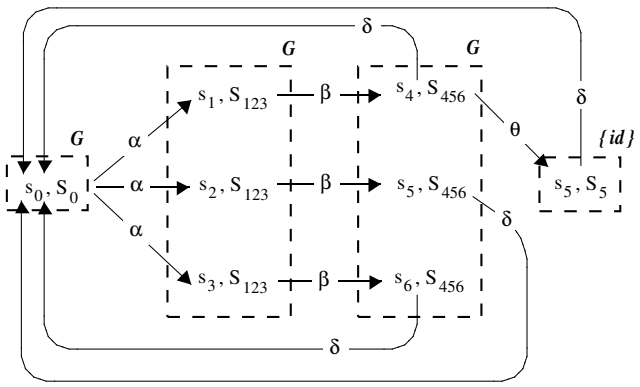

Figure 2. CTMC $\mathcal{C}_\mathcal{A}^G$

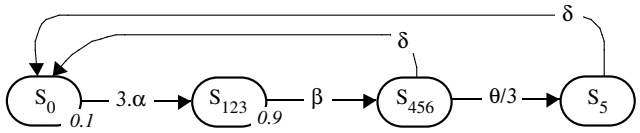

Figure 3. CTMC $(\mathcal{C}_\mathcal{A}^G)^{lp}$

In fact, the stochastic process we want to build is obtained by forgetting the instances and only memorising the subsets.

Definition 9 *Let $\mathcal{C}_\mathcal{A}$ be partially symmetrical w.r.t. G, then the stochastic process $(\mathcal{C}_\mathcal{A}^G)^{lp}$ is defined by: $X_t^{lp} = (R, l)$ iff $X_t'' \in \{(s, l, R)\}$.*

The resulting process for the example is given in Fig.3. The initial distribution and the transition rates are computed according to Prop.3.

The next proposition is the theoretical core of our method. It states that $(\mathcal{C}_\mathcal{A}^G)^{lp}$ is obtained from $\mathcal{C}_\mathcal{A}$ by the inverse of a (strong) lumping followed by an exact lumping.

Proposition 10 *Let $\mathcal{C}_\mathcal{A}$ be partially symmetrical w.r.t. G, then:*

- *Denoting $(s_0, l_0) \ldots, (s_n, l_n)$ the state space of $\mathcal{C}_\mathcal{A}$, $\mathcal{C}_\mathcal{A}$ is a strong lumping of $\mathcal{C}_\mathcal{A}^G$ w.r.t. the partition $\{(s_0, l_0, R)\}_{R \supseteq \{s_0\}}, \ldots, \{(s_n, l_n, R)\}_{R \supseteq \{s_n\}}$*

- *Denoting* $\{(R_0, l_0), \ldots, (R_k, l_k)\}$ *the state space of* $(\mathcal{C}_{\mathcal{A}}^G)^{lp}$, $(\mathcal{C}_{\mathcal{A}}^G)^{lp}$ *is an exact lumping of* $\mathcal{C}_{\mathcal{A}}^G$ *w.r.t. the partition* $\{(s, l_0, R_0)\}_{s \in R_0}, \ldots, \{(s, l_k, R_k)\}_{s \in R_k}$

Proof
Let (s, l) be a state of $\mathcal{C}_{\mathcal{A}}$ and let (s, l, R) be a state of $\mathcal{C}_{\mathcal{A}}^G$, we show that there is a bijective mapping from the transitions out of (s, l) onto the transitions out of (s, l, R). Due to the above remark we suppose that (s, l, R) is examined when looking for successors of $\{(s', l, R) \mid s' \in R\}$ in Def. 8. Then $\exists s', \exists l \xrightarrow{\gamma} l'$ s.t. $\Lambda(s, s') \in \gamma \Leftrightarrow \exists R', \exists s' \in R', \exists l \xrightarrow{\gamma} l'$ s.t. $\Lambda(s, s') \in \gamma$ with $R' = (G_R \cap H_\gamma).s'$. Since this mapping preserves the rate of the transitions the condition of Prop. 2 for strong lumpability is fulfilled.

Let (s_1, l, R) and (s_2, l, R) be two states of $\mathcal{C}_{\mathcal{A}}^G$, we show that there is a bijective mapping from the input transitions of (s_1, l, R) onto the input transitions of (s_2, l, R). Let (v_1, l', R') be such that $\exists l' \xrightarrow{\gamma} l$ and $\Lambda(v_1, s_1) \in \gamma$. Due to the same remark, $\exists g \in G_{R'} \cap H_\gamma$ s.t. $s_2 = g.s_1$. Now define $v_2 = g.v_1$, then $v_2 \in R'$ and $\Lambda(v_2, s_2) \in \gamma$. This implies the existence of the required mapping. Since this mapping preserves the rates of transitions, the condition of Prop. 2 for exact lumpability is fulfilled. $\diamondsuit$

Our generic method can now be described. Assume first that the CTMC $\mathcal{C}_{\mathcal{A}}$ associated with the high-level model $\mathcal{M}$ we want to analyse is partially symmetrical. Assume also that we are able to compute directly $(\mathcal{C}_{\mathcal{A}}^G)^{lp}$ from $\mathcal{M}$. Note π_t the unknown distribution of $\mathcal{C}_{\mathcal{A}}$ at time t and $\pi_t^{(lp)}$ the (computed) distribution of $(\mathcal{C}_{\mathcal{A}}^G)^{lp}$ at time t. Then $\pi_t(s, l) = \sum_{s \in R} (1/|R|) \times \pi_t^{(lp)}(R, l)$. The equality also holds for the steady-state distributions. The next section will show that the assumptions above are satisfied in the framework of SWNs. In fact, we believe that our method is applicable to any model where symmetry is automatically handled.

Although theoretically difficult, we can give some hints of how the space complexity decreases using our approach. In the lumped CTMC, the original states have been substituted by subsets. Note that these subsets may intersect. However these subsets are always the orbit of a state by a subgroup of G. Thus, the larger these subgroups, the better the method. Note that each time a new subset is built, the group is reduced (by intersection with H_γ) and then is enlarged by implicitly substituting to $G \cap H_\gamma$ the isotropy subgroup of the subset. Interpreting this phenomenon at the model level, we deduce that the complexity reduction factor is high whenever the effect of an asymmetrical event is forgotten in a close future. Experimentations will illustrate this interpretation.

3 Application to Stochastic Well-formed Nets

3.1 Presentation of the model and the symbolic reachability graph

WNs are a model of high-level Petri nets whose syntax has been the starting point of numerous efficient analysis methods. Below, we describe the main features of WNs. The reader can refer to [4] for a formal definition:

- In a WN (and more generally in high-level nets) a colour domain is associated with places and transitions. The colours of a place label the tokens contained in this place, whereas the colours of a transition define different ways of firing it. In order to specify these firings, a colour function is attached to every arc which, given a colour of the transition connected to the arc, determines the number of coloured tokens that will be added to or removed from the corresponding place. Finally the initial marking is defined by a multi-set of coloured tokens in each place.

- A colour domain is a cartesian product of colour classes which may be viewed as primitive domains. This product is possibly empty (e.g., a place which contains neutral tokens) and may include repetitions (e.g., a transition which synchronises two colours inside a class). A class can be divided into static subclasses. The colours of a class have the same nature (processes, resources, etc.), whereas the colours inside a static subclass have the same potential behaviour (batch processes, interactive processes, etc.).

- A colour function is built by standard operations (linear combination, composition, etc.) on basic functions. There are three basic functions: a projection which selects an item of a tuple and is denoted by a typed variable (e.g., p, q); a diffusion, a constant function which returns the bag composed by all the colours of a class or a subclass and is denoted S_C where C is the corresponding (sub)class; and a successor function which applies on an *ordered* class and returns the colour following a given colour.

- Transitions and colour functions can be guarded by expressions. An expression is a boolean combination of atomic predicates. An atomic predicate either identifies two variables $[p = q]$ or restricts the domain of a variable to a static subclass.

We illustrate these features on the WN model in Fig. 4. It represents a distributed critical section algorithm. There is a single class C: the set of processes that interact in the system. The colour domain of all the places of the net is C, except for place *TK*, which contains neutral tokens. As

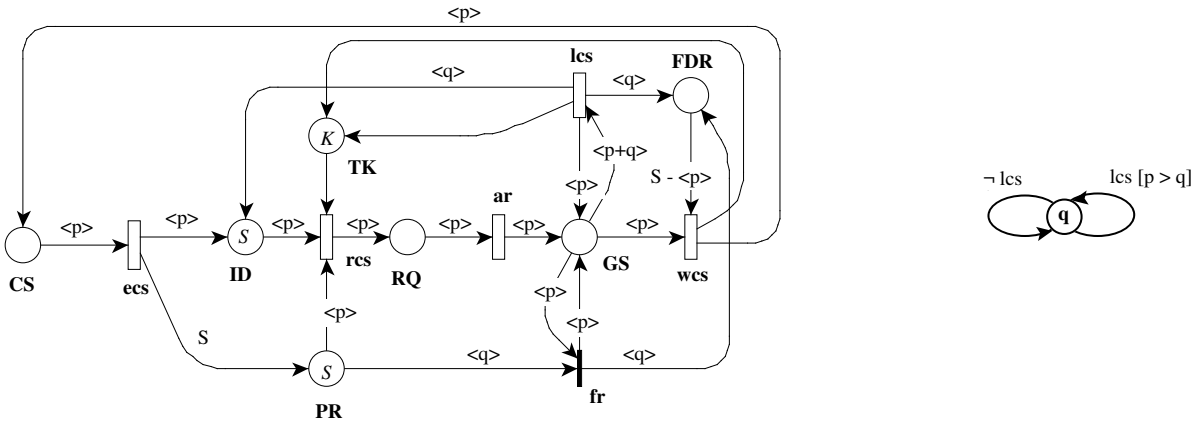

Figure 4. A WN model of a distributed critical section with its control automaton

there is a single class, the constant function representing the set of all processes will be simply denoted S.

Initially, all processes are idle (place *ID*), meaning that they do not request the critical section. The firing of transition *rcs* represents a process requesting the critical section. Only up to *K* processes can apply simultaneously and additional candidacies are rejected, which is represented by *K* tokens in place *TK*. This constant K depends on the parameters of the physical access to the system (e.g., network topology).

As soon as a process reaches the state where others are aware of its request (place *GS*), no process can become candidate any longer : permissions for applying are removed from place *PR* by the firing of *fr* (with priority over other transitions). If there are several candidates, i.e., the number of tokens in places *RQ* and *GS* is greater than one, all but one will be discarded through the successive firings of *lcs*. When there is more than one token in *GS*, the firing of *lcs* non deterministically chooses two of them and discards one. When there is only one left (which is guaranteed by the number of tokens in *FDR*), it can enter the critical section (place *CS*) by firing *wcs*. When the process releases the critical section (firing of transition *ecs*), all processes become idle again and a new round can start.

The implicit symmetry of a WN is associated with a group G_{srg} operating on colour classes (and by extension on markings and firing instances). G_{srg} is the intersection of the isotropy subgroups of static subclasses. In other words, any permutation in G_{srg} maps any static subclass onto itself. Given a marking m and a permutation g of G_{srg}, the behaviour of the net from the marking $g.m$ is the same as the behaviour from m up to permutation g. We say that this two markings are equivalent and we use $\hat{m}$ as a symbolic representation for the orbit $G_{srg}.m$.

The symbolic reachability graph (SRG) construction lies on symbolic markings, namely a compact representation for a set of equivalent ordinary markings. A symbolic marking is a generic representation, where the actual identity of tokens is forgotten and only their distributions among places are stored. Tokens with the same distribution and belonging to the same static subclass are grouped into a so-called dynamical subclass.

In the rest of the paper, we will use a notation where only the cardinality of dynamical subclasses is represented. For instance, in the case where $C = \{c_1, c_2\}$, the symbolic marking $\hat{m} = ID(1) + RQ(1)$ will represent the two ordinary markings $ID(c_1) + RQ(c_2)$ and $ID(c_2) + RQ(c_1)$.

Then, the SRG can be constructed automatically using a symbolic firing rule that directly applies on symbolic markings [4].

Various behaviourial properties may be directly checked on the SRG. Furthermore, this construction leads to an efficient performance evaluation of Stochastic WNs (SWNs). A SWN is obtained from a WN by associating an exponentially distributed delay with every transition. The rate of this transition may depend on the static subclasses to which the firing colours belong. The key result is that the related CTMC may be (strongly and exactly) lumped and that the lumped CTMC is isomorphic to the SRG. As for stochastic Petri nets, the definition can be extended with immediate transitions and a similar result holds for the semi-Markovian process.

However, the SRG approach is not adapted when dealing with asymmetrical systems: in our example, let us now decide that when several processes request the critical section, the selected process is the candidate with the highest identity. Hence, the set of bindings of transition *lcs* must be restricted to pairs (p, q) s.t. $p > q$. In SWNs, this could be done by adding a guard to transition *lcs*. Yet, the only way to express this guard is to partition colour class C into static subclasses reduced to singletons $\{c_i\}$. Then, the guard is $\bigvee_{i>j}(p = c_i \wedge q = c_j)$. Consequently, the SRG is isomor-

phic to the ordinary reachability graph (RG) and there is no more gain in complexity.

3.2 The Dynamic Symbolic Reachability Graph (DSRG) construction

The drawback of the SRG approach is that asymmetries are defined statically and taken into account throughout the construction of the graph. Yet, very often, they have only local effects. Back to our example, except when a process is involved in a selection, there is no need to know its actual identity. Thus, the asymmetry is local to the selection process.

Hence, the challenge is to adapt the method of Section 2, where asymmetrical behaviours are treated as locally as possible, to the SWN model.

The approach we develop here reuses and extends the SWN symbolic framework to automatise the construction of the lumped CTMC $(\mathcal{C}_{\mathcal{A}}^{G})^{lp}$. We call DSRG the symbolic structure that represents this CTMC. It is based on a symbolic representation for $\{(s,l,R) \mid s \in R\}$ and a firing rule that directly applies on it.

The symmetrical features of the system are captured by the SWN and the asymmetries are represented by a control automaton. The definition of the latter requires only that we precise alphabet Σ. Since the CTMC is isomorphic to the RG, the labels associated with it are the firing instances of the transitions. Formally, $\Sigma = \{(t,c) \mid t \in T \wedge c \in C(t)\}$ where T is the set of transitions of the SWN and $C(t)$ is the colour domain of t. Labels of the automaton are subsets of Σ. In our example, there are two labels: $\neg lcs = \{(t,c) \mid t \neq lcs \wedge c \in C(t)\}$ and $lcs[p > q] = \{(lcs,(p,q)) \mid (p,q) \in C^2 \wedge p > q\}$.

Symbolic representation of states in DSRG

In SWNs, a state s is a marking m, and we want a symbolic representation for the set $\{(m,l,R) \mid m \in R\}$, s.t. $R = G'.m^*$, G' a subgroup of G and m^* any marking of R. We choose a notation similar to that of the SRG, namely $\langle D, \hat{m}, l\rangle$, where D is the set of orbits by G' of the colour classes and $\hat{m}$ is the symbolic representation of $G'.m$. We call this representation *symbolic state*.

The symbolic state defined by $D = \{\{c_1,c_2\},\{c_3\}\}$, $\hat{m} = ID(\{c_1,c_2\}_0) + RQ(\{c_1,c_2\}_1 + \{c_3\})$ and an associated location l represents the set $\{(m_1,l,S_{12}),(m_2,l,S_{12})\}$ s.t. $m_1 = ID(c_1) + RQ(c_2 + c_3)$ and $m_2 = ID(c_2) + RQ(c_1 + c_3)$.

Symbolic firing rule in DSRG

Let $l \xrightarrow{\gamma} l'$ be a transition of the control automaton and H_γ be the subgroup of permutations associated to γ. We want to compute the successors of node $\langle D, \hat{m}, l\rangle$ w.r.t. γ, by use of the symbolic firing rule of SWNs.

The key observation is that the restriction imposed by γ can be expressed by a SWN guard that is injected dynamically to the treated net. Thus, H_γ must be represented as a colour class partition in static subclasses, namely D_γ. These static subclasses are used to express the above guard. In our example, the label $lcs[p > q]$ splits C in singletons in order to express $p > q$ as a SWN guard attached to lcs.

To be able to perform a classical symbolic firing, we have to compute a new partition $D' = D \cap D_\gamma$ and refine $\langle D, \hat{m}, l\rangle$ in a family $F = \{\langle D', \widehat{m_1}, l\rangle, \langle D', \widehat{m_2}, l\rangle, \ldots\}$. In the second line of Fig. 5, six among the thirty symbolic markings of the partition are shown. In this case, due to the partition in singletons, each symbolic representation includes only one ordinary state.

Now, the classical SRG symbolic firing rule can be applied on each element of F with the additional control induced by the label. This control is performed at the symbolic level due to the previous splitting.

Back to the generic method, we have built the subset $\{(s,l',R')\}$ with $R' = (G_R \cap H_\gamma).s'^*$. The substitution of $G_R \cap H_\gamma$ by the isotropy subgroup $G_{R'}$ is explicitly performed in SWNs as follows. Two static subclasses reduced to a single dynamic subclass and with the same distribution in places are merged. Observe that the subset of states is unchanged whereas the static partition is rougher. For instance in any marking of the third line of Fig. 5, the three static subclasses (here reduced to a colour) in place ID, can be merged in a single one.

Computation of the transition rates of the lumped CTMC

Using the method described in section 2, the graph we obtain is isomorphic to a lumped Markov chain, whose rates can be computed directly from information obtained during the construction of the DSRG. From the first equation of proposition 3, we know that the transition rate between two classes depends on the cardinalities of the source and destination classes, namely S_i and S_j, and the input rate of any state s of the destination class, i.e., $\sum_{s' \in S_i} Q(s',s)$. Let us consider the contribution to $\sum_{s' \in S_i} Q(s',s)$ of an arc representing the firing of a transition t. After the splitting step, a symbolic instantiation of t is possible for all or none of the markings that still belong to the same symbolic representation. Assume that such a firing of t is possible in a split representation and let us denote S'_i the ordinary states contained in this representation and $|S'_i|$ the number of such states. The global rate out of S_i caused by the firing we consider is $|S'_i|.e.\mu(t)$, where e is the number of ordinary firings represented by the symbolic firing of t and $\mu(t)$ is the rate of transition t. Note that if different bindings of t have different rates, this can be taken into account in the control automaton, thus we consider here only the case where equivalent bindings have equivalent rates.

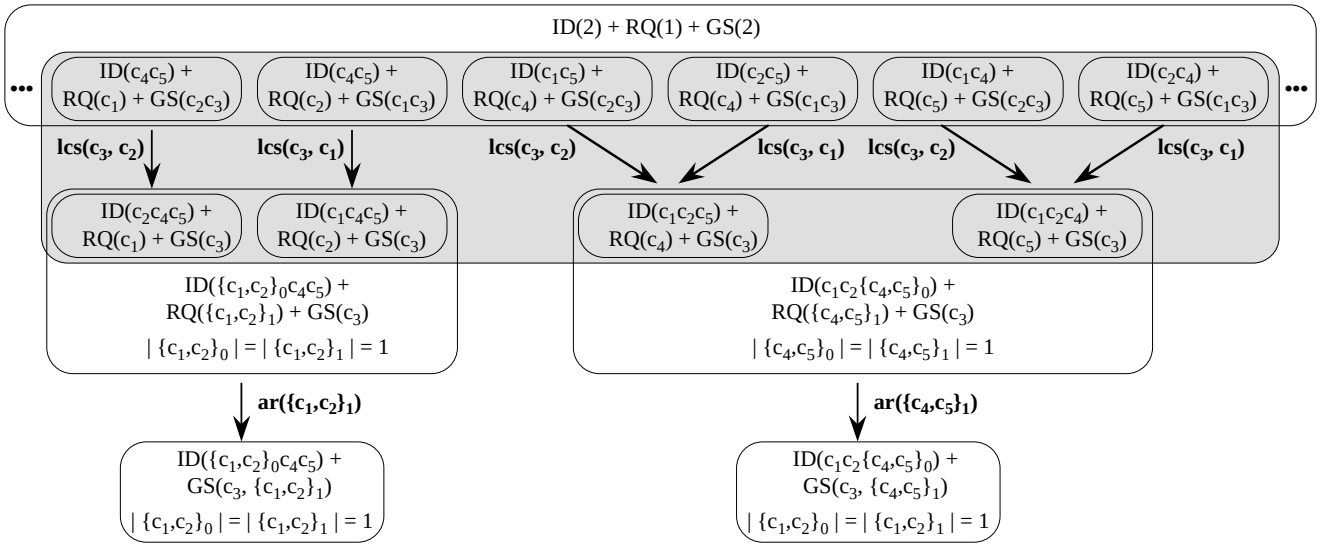

Figure 5. Firing and grouping

As all the states in S_j have the same input rate, this rate is given by $|S_i'|.e.\mu(t)/|S_j|$. Hence $Q^{lp}(i,j) = \frac{|S_i'|.e.\mu(t)}{|S_i|}$. The computations of the number of ordinary markings contained in a symbolic marking and the number of ordinary firings represented by a symbolic firing for the SRG are detailed in [4].

3.3 Optimisations for WNs

In this paragraph, we show that the SWN formalism leads to further optimisations of the generic method. The first one consists in grouping the symbolic representations obtained after a symbolic firing provided that the condition for exact lumpability still holds. This optimisation is feasible since the transition rates of the lumped CTMC can be computed on-the-fly. It appears that after this optimisation, the overall strategy for choosing the next symbolic firing affects the size of the lumped CTMC (which was not the case previously). Hence our second optimisation heuristically tries to minimize this size.

Grouping of symbolic markings

In fact, this optimisation was already proposed in [1]. However, the conditions of this merging were weaker as they require to preserve the existence of particular paths in the graph. When dealing with performance evaluation, states can no longer be grouped on qualitative criteria only. Input rates must be taken into account, which often restricts the possibilities of grouping. For the sake of simplicity, we will consider here a uniform rate of 1.0 for any binding of transition *lcs* and use this example to illustrate the problems that may arise. Let $P = 5$ be the number of processes and

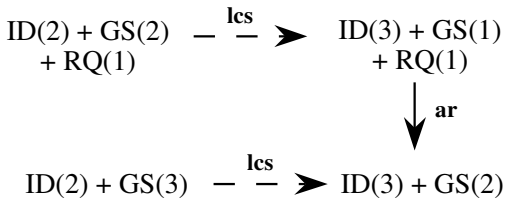

Figure 6. Firings to be considered

$K = 3$ the maximum number of simultaneous candidacies. Fig. 6 represents the distributions of tokens in significant places and the firings of transitions we are going to detail throughout this section.

We consider the firing sequence starting from the symmetrical representation where two processes are idle, one has sent a request and two are ready to perform a selection. The synchronisation of the SWN and the control automaton for the firing of transition lcs ends up in a complete refinement of the source marking, as the only symmetry that is compatible with the label $lcs[p > q]$ is the identity. For each of these refined markings, there is exactly one possible binding of transition lcs, because $lcs[p > q]$ fixes the order between p and q. A significant subset of the refined markings and the corresponding firings is represented in the shaded part of Fig. 5. At this point, we try to group the markings that are obtained from the firing. Obviously, whichever pair we consider, there exists a permutation between the markings. But even if all the source markings belong to the same class, not any pair satisfies the exact lumpability condition. Looking only at the represented markings, as we have considered a uniform rate of 1.0 for transition

lcs, the only possibility we have is to group the two right markings, and also the two left ones. The shaded part is then removed, and only the white portion is actually stored in the graph. The notation $\{c_i, c_j\}_k$ defines a partition of the set $\{c_i, c_j\}$: in the left class for instance, $RQ(\{c_1, c_2\}_1)$ with $|\{c_1, c_2\}_1| = 1$ means that either c_1 or c_2 is in *RQ*, the other one belongs to $\{c_1, c_2\}_0$, hence it is in place *ID*. We have already detailed how transition rates associated with arcs are computed. Once this is done, we can compute the transition rates between classes using the formula in Section 2.

From the classes we have built, we can fire transition *ar*. There is no restriction associated with this transition in the control automaton, hence we use the classical firing rule of SWNs from which we can directly build the class of reached markings : whatever the identity of the token in *RQ*, it is moved to place *GS*.

Overall strategies for construction

We show now that the previous optimisation requires an efficient strategy for choosing the next transition to fire, in order to minimize the size of the lumped CTMC. For instance, what may happen is that a set of markings that is represented by a single class is reached through another firing that prevents them from staying in the same group. We show an example in Fig. 7: the class with two processes in place *ID* and three in place *GS* enables transition *lcs*. Its different bindings will lead to any combination of two processes in place *GS*, except (c_1c_2), and the three other processes in place *ID*. We already encountered such a configuration in the previous firing sequence. However, we did not have to separate the markings where *GS* contained (c_3c_4) or (c_3c_5) because they were obtained from a symmetrical firing, which guaranteed that they had the same input rate. We can see that this is no longer true when we take this firing into account. If we have not built any firing from the subclass representing the two markings yet, we remove the subclass and dispatch its input rate on the individual markings. If we have already built the firings, we keep both representations because removing the subclass could have a domino effect on the downward part of the graph. To avoid as much as possible the construction of redundant subclasses, we try to favour the construction of individual markings first by firing asymmetrical prior to symmetrical transitions.

As the construction of subclasses and individual markings may happen anyway, the same state can be represented several times in the graph. In this case, for any class it appears in, we compute the probability of a state of the class, which is obtained by dividing the probability of the class by its cardinality. The actual probability of the state will then be computed by summing the values obtained for any class it appears in.

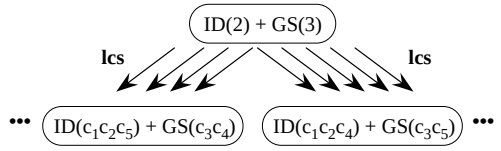

Figure 7. Construction of included markings

3.4 Extension to immediate transitions

When the model includes immediate transitions, the underlying stochastic process becomes semi-Markovian.We use the embedded Markov Chain approach to compute steady-state probabilities. We thus handle a discrete-time process, but the exact lumpability criterion still holds on this process and the steady-state probabilities can be computed in the same way as in the continuous-time case.

Special care must be taken however if some class enables both timed and immediate transitions. This happens for instance if all the colours in a class have had a symmetrical behaviour so far, and an asymmetrical immediate transition is enabled for some of them in the current class of markings, while a timed transition is enabled for others. In this case, the class must be split into a vanishing subclass and a tangible one. But as we consider only input rates for testing the lumping condition, this splitting has no effect on the upward part of the graph.

4 Numerical Results

To test the efficiency of our method, we have implemented the *DSRG* on the same kernel as the standard *(S)RG*. For that, we have modified the GreatSPN package (www.di.unito.it/∼greatspn) on which the *(S)RG* is implemented. Hence, one can specify a SWN to obtain the results of both constructions. The machine used for our tests is a PC/Linux of 3.2 GHz and 3 Gb of RAM.

In this section, we will consider the net of Fig. 4 once again. An examination of its structure would show that the complexity of the model is strongly related to the settings of parameters *P* and *K*, respectively the number of processes in the system and the number of processes that are able to concurrently apply for the critical section ($K \leq P$). For instance, by focusing on the two places *RQ* and *GS*, one notes that increasing *K* acts on the number of tokens in these two places, while increasing *P* widens the possibilities of choosing the identities of the tokens that they contain.

Let us now compare the effects of increasing the values of *P* and *K* on the (S)RG and DSRG methods.

Table 1 summarizes our experiments. The columns noted *(S)RG* (respectively *DSRG*), shows the number of constructed nodes in the *(S)RG* (respectively *DSRG*) structure for a given K and P.

Table 1. Size of the (S)RG and DSRG w.r.t. P and K

P \ K	3			5			7		
	(S)RG	DSRG	Ratio	(S)RG	DSRG	Ratio	(S)RG	DSRG	Ratio
3	45	23	1.66	–	–	–	–	–	–
5	441	49	9.00	573	186	3.08	–	–	–
7	3704	83	44.63	6231	772	8.07	6849	2150	3.18
9	28159	125	225.27	59281	1805	32.82	73549	11150	6.60
14	860371	199	4323.47	7210715	6148	1172.85	17176671	68476	250.84

For a fixed value of *K* and w.r.t. the increasing of *P*, we observe empirically that the *RG* grows exponentially whereas the *DSRG* progresses *almost linearly*. This is easily explained by the fact that the complexity induced by the different possibilities to select K concurrent processes are explicitly represented in the *(S)RG*, whereas they are symbolically represented in the *DSRG*. More precisely, in the *DSRG*, no asymmetry among processes is taken into account until asymmetrical transition *lcs* is enabled. Moreover, the symbolic grouping optimisations make it possible to regain part of the symmetries that are lost due an asymmetrical firing.

For a fixed value of *P*, one should observe that the sizes of both structures increase exponentially. However, there is a limitation as the number K is closed to its maximum, P. Such a limitation is clearly seen for the *(S)RG*, where the rate of augmentation of the size decreases for any given P. Unfortunately, we cannot compare the *DSRG* to the *(S)RG* since the effect of our symbolic grouping operations is contextual (thus not controllable).

Nevertheless, we are able to compare the relative gain of our method (see columns noted *Ratio* of Table 1). Thus, the ratio between the two structures progresses exponentially w.r.t. P and this proves the efficiency of our approach. W.r.t. K, a regression can be noted, it is caused by the exponential observed above.

Last, we notice that the *DSRG* construction time is relatively high. As an example for values $P = 9$ and $K = 5$, the building requires 275 seconds, while it requires 126 seconds for the RG. In fact, 47% of the total construction time is spent in comparisons of sets of colours and this percentage remains constant for all constructions. Therefore, we are working to integrate cache techniques in our software to solve this problem.

5 Conclusion and perspectives

We have proposed a new automatised and symbolic method to build a lumped CTMC which verifies exact lumpability. Our method is generic and should be applicable to a large category of performance models. Moreover, in practical cases, the additional specification of the control automaton remains straightforward. Applied to a (common) use case, the DSRG construction appears to be very relevant in terms of used memory. We need now to improve our tool in order to gain efficiency in time. Our next research perspective will be to extend the proposed method to probabilistic model checking.

References

[1] S. Baarir, S. Haddad, and J.-M. Ilié. Exploiting Partial Symmetries in Well-formed nets for the Reachability and the Linear Time Model Checking Problems. In *Proc. of WODES'04 - IFAC Workshop on Discrete Event Systems, part of 7th CAAP*, Reims - France, 2004. Springer Verlag.

[2] C. Bellettini and L. Capra. A quotient graph for asymmetric distributed systems. *12th IEEE Int. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 560–568, october 2004.

[3] L. Capra, C. Dutheillet, G. Franceschinis, and J.-M. Ilié. Exploiting partial symmetries for Markov chain aggregation. *E. Notes in Theoretical Computer Science*, 39(3), 2000.

[4] 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, nov 1993.

[5] E. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting Symmetry in Temporal Logic Model Chacking. *Formal Methods and System Design*, 9:77–104, 1996.

[6] E. A. Emerson and R. J. Trefler. From Asymmetry to Full Symmetry: New Techniques For Symmetry Reduction in Model Checking. In *Proc of CHARME99*, Lecture Notes in Computer Science, pages 142–156, Bad Herrenalb - Germany, Sept. 1999. Springer Verlag.

[7] S. Haddad, J. Ilié, and K. Ajami. A model checking method for partially symmetric systems. In *Proceedings of FORTE/PSTV'00*, pages 121–136, Pisa, Italy, Oct. 2000. Kluwer Academic Publishers.

[8] S. Haddad, J. Ilié, M. Taghelit, and B. Zouari. Symbolic Reachability Graph and Partial Symmetries. In *Proc. of the* 16^{th} *Intern. Conference on Application and Theory of Petri Nets*, volume 935 of *LNCS*, pages 238–257, Turin, Italy, June 1995. Springer Verlag.

[9] J. Ilié, S. Baarir, M. Beccuti, S. Donatelli, C. Dutheillet, G. Franceschinis, R. Gaeta, and P. Moreaux. Extended SWN Solvers in GreatSPN. In *Tool paper for the 1st Int. Conf. on Quantitative Evaluation of Systems (QEST'04)*, LNCS, Twente, Netherland, 2004. Springer Verlag.

[10] J. Kemeny and J. Snell. Finite Markov chains. New York, NY, 1960. D. Van Nostrand-Reinhold.

[11] G. Rubino and B. Sericola. On weak lumpability in Markov chains. *Journal of Appl. Prob.*, 26:446–457, 1989.

[12] P. J. Schweitzer. Aggregation methods for large Markov chains. In *Proceedings of the International Workshop on Computer Performance and Reliability*, pages 275–286. North-Holland, 1984.