A Framework to Decompose GSPN Models*

Leonardo Brenner, Paulo Fernandes**, Afonso Sales, and Thais Webber

Pontifícia Universidade Católica do Rio Grande do Sul,
Av. Ipiranga, 6681 – 90619-900 – Porto Alegre, Brazil
{lbrenner, paulof, asales, twebber}@inf.pucrs.br

Abstract. This paper presents a framework to decompose a single GSPN model into a set of small interacting models. This decomposition technique can be applied to any GSPN model with a finite set of tangible markings and a generalized tensor algebra (Kronecker) representation can be produced automatically. The numerical impact of all the possible decompositions obtained by our technique is discussed. To do so we draw the comparison of the results for some practical examples. Finally, we present all the computational gains achieved by our technique, as well as the future extensions of this concept for other structured formalisms.

1 Introduction

It is common knowledge in the research community the advantages in using the GSPN (*Generalized Stochastic Petri Nets*) formalism [2] to model complex systems, *i.e.*, systems with both parallel and synchronous behavior. For a quite long time, the main limitation to use the GSPN formalism was the absence of an efficient numerical support to handle useful, and consequently large, models. Ciardo and Trivedi's work [14] brought a first approach that could be employed to decompose a single model into components. However, their approach does not mention any specific storage or numerically suitable solution technique. The need of a theoretical tool to represent such structured models naturally leads to *Tensor Algebra* representations [4, 8, 3, 15]. The term *Tensor Algebra* is being employed in this paper, but many authors still prefer the classic denomination *Kronecker Algebra* chosen in honor of Leopold Kronecker.

The first complete approaches in this direction were the works of Donatelli in the SGSPN (*Superposed GSPN*) formalism [16, 17]. By complete, we understand that it was proposed a complete framework to: construct a SGSPN model by assembling synchronized components; generate a Markovian descriptor, *i.e.*, a tensor algebra formula, as the infinitesimal generator of the equivalent Markov chain; and consequently an efficient way to solve it (functional elements). However, the SGSPN formalism could only be used to model a rather small class of GSPN models which comply to the restrictive rules of generation defined by

* This work was partially funded by CNPq/Brazil.
** Corresponding author. The order of authors is merely alphabetical.

G. Ciardo and P. Darondeau (Eds.): ICATPN 2005, LNCS 3536, pp. 128–147, 2005.

Donatelli, *i.e.*, a SGSPN model is composed of a set of standard GSPN models which interact only through a set of synchronized transitions.

The SGSPN application scope restriction, and the consequent disadvantages in terms of numerical performance, suggests the use of other formalisms that could be closer to the tensor representation, such as SAN (*Stochastic Automata Networks*) [26]. At this time, the solution through the shuffle algorithm used in SAN [18, 7] presents an efficient solution with reasonable memory needs. Evidently, the use of other structured storage and solution techniques instead of the tensor algebra also presents good alternatives to the limited scope of the SGSPN formalism. This is the case of the quite impressive techniques based on MDD (*Multi-valued Decision Diagrams*) [22] and MxD (*Matrix Diagrams*) [21] proposed by Ciardo and Miner. Furthermore, the MDD and MxD techniques are very efficient to solve very sparse models, *i.e.*, models with a huge product state space and a comparatively small number of reachable states. In fact, we believe that the techniques based on tensor algebra are still worthy, at least considering the new improvements to handle tensor structures [5, 10].

This paper presents a study about the decomposition of a very general class of GSPN models, exploiting the description power of the GSPN formalism. It also proposes a memory efficient tensor algebra format to describe the components and their interactions. As the first step, we formally define the class of GSPN models in which our technique can be applied. The proposed decomposition technique and the consequent tensor format representation are generalizations of the SGSPN formalism [17], but we extend the application scope following the ideas firstly advanced in [14] and employed later in [20]. The new contribution of our work is justified by the numerical impact of the decomposition choices on the storage demands.

We are specifically interested to handle models with a really large reachable state space. Buchholz, Ciardo, Donatelli, Kemper, and Miner [9, 23, 11] already present very efficient methods to deal with absolutely huge models (*e.g.*, 9.18×10^{626} states in 1000 dining philosophers example [22]), but with considerably fewer reachable states. Our decomposition technique intends to split a GSPN model into subnets providing a structured representation. Regardless the number of subnets, we will always have the same reachable state space. With many (small) subnets, we have a very structured (and therefore memory efficient) representation, but also a product state space much larger than the reachable state space. With few (large) subnets, we have a less structured representation, but a product state space equal or a little bit larger than the reachable state space. In fact, we intend to compare possible decompositions in order to show the trade off between many small and few large subnets.

In addition, we point out the underestimated benefits of the use of guards in the GSPN formalism, which can be clearly demonstrated by the tensor format proposed in our work. We do not pay much attention in this paper to the computational cost to solve the tensor representations. The recent evolutions in pure tensor solutions [5, 10], the promising ideas of parallel implementations, and the MDD and MxD techniques [12] suggest many changes in the computational

cost in a near future. We focus our interest in the memory savings due to our decomposition techniques and the corresponding tensor format.

The next section briefly presents the theoretical tool used to the tensor representation: Classical (CTA) and Generalized Tensor Algebra (GTA). Section 3 describes the GSPN formalism, the application scope of our technique and the scope of the technique proposed for SGSPN. Section 4 presents our decomposition technique and the corresponding tensor format. Section 5 draws some considerations about the possible choices of decomposition. Section 6 presents some modeling examples in order to discuss numerical issues about the decomposition technique. Finally, the conclusion summarizes our contribution and suggests the still vast future work to be done.

2 Tensor Algebra

In this section, the concepts of Classical Tensor Algebra [3, 15] and Generalized Tensor Algebra [25, 18] are briefly presented.

2.1 CTA - Classical Tensor Algebra

The tensor product of two matrices: A of dimensions $(\rho_1 \times \gamma_1)$ and B of dimensions $(\rho_2 \times \gamma_2)$ is a tensor with dimensions $(\rho_1\rho_2 \times \gamma_1\gamma_2)$ which may be considered as consisting of $\rho_1\gamma_1$ blocks each having dimensions $(\rho_2\gamma_2)$, *i.e.*, the dimensions of B. To specify a particular element, it suffices to specify the block in which the element occurs and the position within that block of the element under consideration. Thus, as mentioned previously, element c_{36} $(a_{11}b_{02})$ is in the (1, 1) block and at position (0, 2) of that block. The tensor $C = A \otimes B$ is defined by assigning to the element of C that is in the (k, l) position of block (i, j), the value $a_{ij}b_{kl}$, *i.e.*, $c_{[ik][jl]} = a_{ij}b_{kl}$. The *tensor sum* of two *square* matrices A and B is defined in terms of tensor products as:

$$A \oplus B = A \otimes I_{n_B} + I_{n_A} \otimes B$$

where n_A is the order of A; n_B is the order of B; I_{n_i} is the identity matrix of order n_i; and "+" represents the usual operation of matrix addition. Since both sides of this operation (matrix addition) must have identical dimensions, it follows that tensor addition is defined for square matrices only. The value assigned to the element $c_{[ik][jl]}$ of the tensor $C = A \oplus B$ is $c_{[ik][jl]} = a_{ij}\delta_{kl} + b_{kl}\delta_{ij}$, where δ_{ij} is the element of i^{th} row and j^{th} column of an identity matrix defined as $\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$.

2.2 GTA - Generalized Tensor Algebra

Generalized Tensor Algebra is an extension of Classical Tensor Algebra. The main distinction of GTA with respect to CTA is the addition of the concept of *functional elements*. However, a matrix can be composed of constant elements

(belonging to $\mathbb{R}$) or functional elements. A functional element is a function evaluated in $\mathbb{R}$ according to a set of parameters composed of the rows of one or more matrices. Generalized tensor product is denoted by $\underset{g}{\otimes}$. The value assigned to the element $c_{[ik][jl]}$ of the tensor $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ is $c_{[ik][jl]} = a_{ij}(b_k)b_{kl}(a_i)$. Generalized tensor sum is also analogous to the ordinary tensor sum, and is denoted by $\underset{g}{\oplus}$. The elements of the tensor $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ are $c_{[ik][jl]} = a_{ij}(b_k)\delta_{kl} + b_{kl}(a_i)\delta_{ij}$.

3 Generalized Stochastic Petri Nets

The GSPN formalism [2] is a performance analysis tool on the graphical system representation typical of Petri Nets [27, 24]. The GSPN formalism is derived from the SPN formalism and contains two types of transitions: *timed* and *immediate*. An exponentially distributed random firing time is associated with each timed transition, whereas immediate transitions, by definition, fire in zero time. Immediate transitions always have precedence to fire over timed transitions. The GSPN models with immediate transitions can always be represented by a model with timed transitions.

In the graphical representation of a GSPN model, places are drawn as circles, timed transitions as rectangles and immediate transitions as bars. Places may contains *tokens*, which are drawn as black dots. A place is an input to a transition if an arc exists from the place to the transition. A place is an output from a transition if an arc exists from the transition to the place. A transition is enabled when all of its input places contain at least one token. Enabled transitions can fire, thus removing one token from each input place and placing one token in each output place. Additionally, a condition can be associated to enable the firing of the transitions. Such conditions are called *guards* and, with the availability of tokens in the input places, they are the only restrictions to enable the firing of a given transition. A formal description is presented as follows.

Let

$\mathcal{C}$ set of conditions associated to transitions of $\mathcal{T}$.

Definition 1. *A $\mathcal{GSPN}$ is defined by tuple ($\mathcal{P}$, $\mathcal{T}$, π, I, O, W, G, M_0), where:*

1.1. $\mathcal{P}$ non-empty set of places;
1.2. $\mathcal{T}$ non-empty set of transitions;
1.3. $\pi: \mathcal{T} \rightarrow \{0, 1\}$ priority function of the transitions;
1.4. I and $O: \mathcal{T} \rightarrow \mathcal{P}$ input and output functions of the transitions;
1.5. $W: \mathcal{T} \rightarrow \mathbb{R}^+$ function that assigns a rate to each transition;
1.6. $G: \mathcal{T} \rightarrow \mathcal{C}$ function, called guard, that associates a necessary, but not sufficient, condition $c \in \mathcal{C}$ to the firing of each transition $t \in \mathcal{T}$;
1.7. $M_0: \mathcal{P} \rightarrow \mathbb{N}$ initial marking in each place.

Definition 2. *$c \in \mathcal{C}$ is a condition that may be associated to a transition $t \in \mathcal{T}$, which depends on tokens of one or more $p \in \mathcal{P}$. This condition is a function with domain on tokens of all places p and counter-domain on $\mathbb{R}$.*

A condition c defines the firing rate of a transition according to the number of tokens in a specific set of places. Although the counter-domain of c is $\mathbb{R}$, only a discrete set of values can be obtained, since the possible combination of markings of places (*i.e.*, the domain of c) is a discrete set.

Definition 3. *Set of timed transitions $\mathcal{T}_T$ of a $\mathcal{GSPN}$ is defined as $\mathcal{T}_T = \{t \in \mathcal{T} \mid \pi(t) = 0\}$.*

Definition 4. *Set of immediate transitions $\mathcal{T}_I$ of a $\mathcal{GSPN}$ is defined as $\mathcal{T}_I = \{t \in \mathcal{T} \mid \pi(t) = 1\}$.*

Definition 5. *Set of transitions $\mathcal{T}$ of a $\mathcal{GSPN}$ is defined as $\mathcal{T} = \mathcal{T}_T \cup \mathcal{T}_I$ and $\mathcal{T}_T \cap \mathcal{T}_I = \emptyset$.*

Numerical Solution Restriction. Although the framework proposed in this paper could be applied to a larger class of GSPN models, we assume a single restriction in order to facilitate the stationary or transient numerical solution: the set of tangible markings of the models must be finite.

4 Framework

We present in this section a framework to decompose GSPN models. Our decomposition technique is shown in Fig. 1.

The basic idea is to decompose a GSPN model into N components $\mathcal{GSPN}^{(i)}$ ($i \in [1..N]$), where each component $\mathcal{GSPN}^{(i)}$ is viewed as a subsystem of the GSPN model. A component $\mathcal{GSPN}^{(i)}$ may not be a GSPN model. It is then necessary to know the possible tangible markings. This is done by the construction of $\mathcal{TRG}^{(i)}$ considering the possible firing of all transitions limited by: availability of tokens; guards; and maximum number of tokens in each place.

A component $\mathcal{GSPN}^{(i)}$ has an independent behavior (*local transitions*) and occasional interdependencies (*synchronized transitions* and/or transitions with guards). It is important to notice that there is a strong equivalence between the decomposition of all $\mathcal{TRG}^{(i)}$ and the $\mathcal{TRG}$ of the whole GSPN (which is the underlying Markov Chain). Nevertheless the computational cost to obtain the $\mathcal{TRG}$ from the composition of all $\mathcal{TRG}^{(i)}$ is usually too high. In fact, the memory needs can be prohibitive as will be seem in the Section 6.

In the next section, we define a component $\mathcal{GSPN}^{(i)}$ and its properties. In Section 4.2, we formally present the tensor format (*Markovian Descriptor*) used to obtain the infinitesimal generator Q of a decomposed GSPN model.

4.1 Decomposition

There is no restriction to decompose a GSPN model with a finite set of tangible marking into N components $\mathcal{GSPN}^{(i)}$ ($i \in [1..N]$). Our technique is less restrictive than the definition of the SGSPN formalism.

Definition 6. *Each component $\mathcal{GSPN}^{(i)}$ is defined as a* GSPN*, i.e., it is defined by tuple ($\mathcal{P}^{(i)}$, $\mathcal{T}^{(i)}$, $\pi^{(i)}$, $I^{(i)}$, $O^{(i)}$, $W^{(i)}$, $G^{(i)}$, $M_0^{(i)}$), where:*

6.1. $\mathcal{P}^{(i)}$ *non-empty set of places, such that* $p^{(i)} \in \mathcal{P}^{(i)} \rightarrow p^{(i)} \in \mathcal{P}$ *and* $\bigcup_{i=1}^{N} \mathcal{P}^{(i)} = \mathcal{P}$ *and* $\nexists \mathcal{P}^{(i)} = \mathcal{P}$*;*
6.2. $\mathcal{T}^{(i)}$ *non-empty set of transitions, such that* $t^{(i)} \in \mathcal{T}^{(i)} \rightarrow t^{(i)} \in \mathcal{T}$ *and* $\exists p \in I^{(i)}(t)$ *or* $\exists p \in O^{(i)}(t)$ *such that* $p \in \mathcal{P}^{(i)}$*;*
6.3. $\pi^{(i)}$*:* $\mathcal{T}^{(i)} \rightarrow \{0, 1\}$ *priority function of the transitions;*
6.4. $I^{(i)}$ *and* $O^{(i)}$*:* $\mathcal{T}^{(i)} \rightarrow \mathcal{P}^{(i)*}$ *input and output functions of the transitions in which* $\mathcal{P}^{(i)*}$ *denotes a possibly empty set of places;*
6.5. $W^{(i)}$*:* $\mathcal{T}^{(i)} \rightarrow \mathbb{R}^{+}$ *function that assigns a rate to each transition;*
6.6. $G^{(i)}$*:* $\mathcal{T}^{(i)} \rightarrow \mathcal{C}$ *function guard that associates a necessary, but not sufficient, condition* $c^{(i)} \in \mathcal{C}$ *to the firing of each transition* $t^{(i)} \in \mathcal{T}^{(i)}$*;*
6.7. $M_0^{(i)}$*:* $\mathcal{P}^{(i)} \rightarrow \mathbb{N}$ *initial marking in each place* $p^{(i)} \in \mathcal{P}^{(i)}$*.*

It is important to notice that the set of places $\mathcal{P}^{(i)}$ is a subset of $\mathcal{P}$, as well as the set of transitions $\mathcal{T}^{(i)}$ is a subset of $\mathcal{T}$. Obviously, the subset of places $\mathcal{P}^{(i)}$ of a component $\mathcal{GSPN}^{(i)}$ cannot be the whole set of places $\mathcal{P}$, otherwise there is no decomposition. The same restriction does not apply to $\mathcal{T}^{(i)}$, since it can be identical to $\mathcal{T}$.

There is no restriction to places superposition. A place $p \in \mathcal{P}$ can be in as many subsets of places $\mathcal{P}^{(i)}$ as wanted. Obviously, the same applies to transitions. The sole restriction regards the immediate transitions that cannot be used to synchronized two or more partitions. However such restriction is a minor discomfort since all GSPN model can be described by an equivalent SPN (*i.e.*, without immediate transitions) model. Elements of tuple ($\mathcal{P}^{(i)}$, $\mathcal{T}^{(i)}$, $\pi^{(i)}$, $I^{(i)}$, $O^{(i)}$, $W^{(i)}$, $G^{(i)}$, $M_0^{(i)}$) are conservatives, *i.e.*, an element in component $\mathcal{GSPN}^{(i)}$ has the same value of the corresponding element in that original GSPN, *e.g.*, if $t^{(i)}$ correspond to t, then $W^{(i)}(t^{(i)})$ has the same value of $W(t)$.

4.2 Tensor Format

We now formally present the tensor format (*Markovian Descriptor*) used to obtain the infinitesimal generator Q of a decomposed GSPN model. As shown in Fig. 1, the decomposition technique uses the concepts of *Tangible Reachability Graph* and *Stochastic State Machine*.

So we firstly remind the classical definitions of *P-invariants*, *Reachability Set* (RS), *Tangible Reachability Set* (TRS), *Tangible Reachability Graph* (TRG) and *Stochastic State Machine* (SSM).

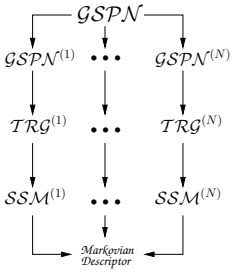


Fig. 1. Decomposition technique

Let

C incidence matrix of a GSPN (dimensions: $|\mathcal{P}| \times |\mathcal{T}|$);

c_{jk} element from row j and column k of an incidence matrix.

Definition 7. *Elements of an incidence matrix C are defined by:*

7.1. $\forall p_j \in \mathcal{P}, \forall t_k \in \mathcal{T}$

$$c_{jk} = \begin{cases} +1 & \text{if } p_j \in O(t_k) \\ -1 & \text{if } p_j \in I(t_k) \\ 0 & \text{if } p_j \notin O(t_k) \text{ and } p_j \notin I(t_k) \end{cases}$$

Definition 8. *P-invariants of a GSPN are defined by vector solutions σ composed of non-negative integer: 0 and 1, given by equation $\sigma C = 0$ [27], where value 1 in i^{th} position of σ means that i^{th} place of GSPN belongs to the P-invariant.*

Let

$\mathcal{PI}$ minimal set of P-invariants, where $\mathcal{PI} = \{\mathcal{PI}_1, \mathcal{PI}_2, ...\}$.

The scalar product between a P-invariant and any marking M produces a *constant*. If in a GSPN all places are covered by P-invariant, the maximum number of tokens in any place in any reachable marking is finite, and the net is said to be *bounded* [1]. Therefore, a GSPN must have all places covered by P-invariants (all places are bounded) in order to have a finite set of tangible markings. We assume a minimal set of P-invariants as a set with the smaller number of P-invariants that covers all places of the whole net.

Let

$M_i(p)$ number of tokens in place p in marking M_i;

$B(\mathcal{PI}_i)$ number of tokens in any P-invariant $\mathcal{PI}_i$ (*bound*);

$max(p)$ maximum number of tokens in place p defined as the minimum $B(\mathcal{PI}_i)$ for all $\mathcal{PI}_i$, where $p \in \mathcal{PI}_i$;

$M_k[t > M_l$ change from marking M_k to M_l due to the firing of t.

Definition 9. *Reachability Set* $\mathcal{RS}^{(i)}(M_0^{(i)})$ *of component* $\mathcal{GSPN}^{(i)}$ *is defined as the smallest set of markings, such that:*

9.1. $M_0^{(i)} \in \mathcal{RS}^{(i)}(M_0^{(i)})$;
9.2. $M_l^{(i)} \in \mathcal{RS}^{(i)}(M_0^{(i)})$, *if and only if* $\forall p^{(i)}, M_l^{(i)}(p^{(i)}) \leq max(p^{(i)})$; *and* $\exists M_k^{(i)} \in \mathcal{RS}^{(i)}(M_0^{(i)})$ *and* $\exists t \in \mathcal{T}^{(i)}$ *such that* $M_k^{(i)}[t > M_l^{(i)}$.

Definition 10. *Tangible Reachability Set* $\mathcal{TRS}^{(i)}(M_0^{(i)})$ *of component* $\mathcal{GSPN}^{(i)}$ *is composed of all tangible markings of* $\mathcal{RS}^{(i)}(M_0^{(i)})$.

Definition 11. *Tangible Reachability Graph* $\mathcal{TRG}^{(i)}(M_0^{(i)})$ *of component* $\mathcal{GSPN}^{(i)}$ *given an initial marking* $M_0^{(i)}$ *is a labelled directed multigraph whose set of nodes* $\mathcal{TM}^{(i)}$ *is composed of markings of Tangible Reachability Set* $\mathcal{TRS}^{(i)}(M_0^{(i)})$ *and whose set of arcs* $\mathcal{TARC}^{(i)}$ *is defined as follows:*

11.1. $\mathcal{TARC}^{(i)} \subseteq \mathcal{TRS}^{(i)}(M_0^{(i)}) \times \mathcal{TRS}^{(i)}(M_0^{(i)}) \times \mathcal{T}_T^{(i)} \times \mathcal{T}_I^{(i)*}$;
11.2. $a^{(i)} = [M_k^{(i)}, M_l^{(i)}, t_0, \sigma] \in \mathcal{TARC}^{(i)}$, *if and only if* $M_k^{(i)}[t_0 > M_1^{(i)}, \sigma = t_1, \ldots, t_n, (n \geq 0)$; *and*
11.3. $\exists M_2^{(i)}, \ldots, M_n^{(i)}$ *such that* $M_1^{(i)}[t_1 > M_2^{(i)}[t_2 > \ldots M_n^{(i)}[t_n > M_l^{(i)}$.

Definition 12. *A Stochastic State Machine (SSM) is defined by tuple* $(\mathcal{P}, \mathcal{T}, F, \Lambda)$, *where:*

12.1. $\mathcal{P}$ *set of non-empty places;*
12.2. $\mathcal{T}$ *set of non-empty transitions;*
12.3. $F \subseteq ((\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P}))$ *with* $dom(F) \cup codom(F) = \mathcal{P} \cup \mathcal{T}$ *is the flow relation. It has to satisfy the following restriction*[1]*:* $\forall t \in \mathcal{T}$*:* $|{}^\circ t| = |t^\circ| = 1$;
12.4. $\Lambda : \mathcal{T} \to \mathbb{R}^+$, *where* $\Lambda(t)$ *is the rate of the exponential probability distribution associated to transition* t.

A decomposed GSPN model has N components $\mathcal{GSPN}^{(i)}$, where $i \in [1..N]$. Each component $\mathcal{GSPN}^{(i)}$ has a tangible reachability graph $\mathcal{TRG}^{(i)}$ (Definition 12). Each tangible reachability graph $\mathcal{TRG}^{(i)}$ has an equivalent stochastic state machine $\mathcal{SSM}^{(i)}$ such that:

1. Each node $M_j^{(i)} \in \mathcal{TM}^{(i)}$ corresponds to $p^{(i)} \in \mathcal{P}^{(i)}$ of $\mathcal{SSM}^{(i)}$;
2. Each arc $a^{(i)} \in \mathcal{TARC}^{(i)}$ corresponds to $[p^{(i)}, t^{(i)}] \in F^{(i)}$ and $[t^{(i)}, q^{(i)}] \in F^{(i)}$, if and only if exist $a^{(i)} = [M_k^{(i)}, M_l^{(i)}, t, \sigma]$ such that $M_k^{(i)}$ corresponds to place $p^{(i)} \in \mathcal{P}^{(i)}$, $M_l^{(i)}$ corresponds to place $q^{(i)} \in \mathcal{P}^{(i)}$, $t \in \mathcal{T}_T^{(i)}$, and $\sigma \in \mathcal{T}_I^{(i)*}$.

[1] $|{}^\circ t|$ and $|t^\circ|$ indicate the number of input and output places of t.

The transition rate of $t^{(i)} \in \mathcal{T}^{(i)}$ (obtained from $[M_k^{(i)}, M_l^{(i)}, t, \sigma]$) can be computed as $\Lambda(t).\Lambda(\sigma)^2$, where $\Lambda(t)$ is the transition rate of t. Any transition $t^{(i)} \in \mathcal{T}^{(i)}$, whose guard has dependency on markings of other components $\mathcal{GSPN}^{(j)}$ $(i, j \in [1..N]$ and $i \neq j)$, has a function f multiplied by its rate. Function f is evaluated as *true* for all markings whose its guard is satisfied, and *false* otherwise. So we can now classify a transition as *local* or *synchronized*.

Let

$\mathcal{T}_l^{(i)}$ set of local transitions of component $\mathcal{SSM}^{(i)}$;

$\mathcal{T}_s^{(i)}$ set of synchronized transitions of component $\mathcal{SSM}^{(i)}$.

Definition 13. *Set of synchronized transitions $\mathcal{T}_s$ of a decomposed GSPN model is defined as $\mathcal{T}_s = \mathcal{T}_s^{(1)} \cup \mathcal{T}_s^{(2)} \cup \ldots \cup \mathcal{T}_s^{(N)}$.*

Markovian Descriptor is an algebraic formula that allows to store, in a compact form, the infinitesimal generator of an equivalent Markov chain. This mathematical formula describes the infinitesimal generator through the transition tensors of each component. Each component $\mathcal{SSM}^{(i)}$ has associated:

- 1 tensor $Q_l^{(i)}$, which has all transition rates for local transitions in $\mathcal{T}_l^{(i)}$;
- $2|\mathcal{T}_s|$ tensors $Q_{t^+}^{(i)}$ and $Q_{t^-}^{(i)}$, which have all transition rates for synchronized transitions in $\mathcal{T}_s^{(i)}$.

Let

$\mathcal{Q}_k^{(i)}(p^{(i)}, q^{(i)})$ tensor element $Q_k^{(i)}$ from row $p^{(i)}$ and column $q^{(i)}$, where $i \in [1..N]$ and $k \in \{l, t^+, t^-\}$;

$I_{|\mathcal{P}^{(i)}|}$ identity tensor of order $|\mathcal{P}^{(i)}|$, where $i \in [1..N]$;

$\tau_t(p^{(i)}, q^{(i)})$ occurrence rate of transition $t \in \mathcal{T}^{(i)}$, where $[p^{(i)}, t] \in F^{(i)}$ and $[t, q^{(i)}] \in F^{(i)}$;

$succ_t(p^{(i)})$ successor place $q^{(i)}$ such that $[p^{(i)}, t] \in F^{(i)}$ and $[t, q^{(i)}] \in F^{(i)}$.

Definition 14. *Tensor elements $Q_l^{(i)}$, which represent all local transitions $t \in \mathcal{T}_l^{(i)}$ of component $\mathcal{SSM}^{(i)}$, are defined by:*

14.1. $\forall p^{(i)}, q^{(i)} \in \mathcal{P}^{(i)}$ *such that* $q^{(i)} \in succ_t(p^{(i)})$ *and* $p^{(i)} \neq q^{(i)}$

$$Q_l^{(i)}(p^{(i)}, q^{(i)}) = \sum_{t \in \mathcal{T}_l^{(i)}} \tau_t(p^{(i)}, q^{(i)});$$

14.2. $\forall p^{(i)} \in \mathcal{P}^{(i)}$ *such that* $q^{(i)} \in succ_t(p^{(i)})$

$$Q_l^{(i)}(p^{(i)}, p^{(i)}) = -\sum_{t \in \mathcal{T}_l^{(i)}} \tau_t(p^{(i)}, q^{(i)});$$

[2] See [1] for the computation of $\Lambda(\sigma)$ (sequence of immediate transitions).

14.3. $\forall p^{(i)}, q^{(i)} \in \mathcal{P}^{(i)}$ *such that* $q^{(i)} \notin succ_t(p^{(i)})$ *and* $p^{(i)} \neq q^{(i)}$
$Q_l^{(i)}(p^{(i)}, q^{(i)}) = 0.$

Let

$\eta^{(t)}$ set of indices i ($i \in [1..N]$) such that component $\mathcal{SSM}^{(i)}$ has at least one transition $t \in \mathcal{T}^{(i)}$;

$\iota^{(t)}$ index of the component $\mathcal{SSM}$ which has the transition rate of synchronized transition $t \in \mathcal{T}_s$, where $\iota^{(t)} \in [1..N]$.

Actually a transition t can be viewed as local transition if $|\eta^{(t)}|= 1$ or as synchronized transition if $|\eta^{(t)}|> 1$.

Definition 15. *Tensor elements* $Q_{t^+}^{(i)}$*, which represent the occurrence of synchronized transition* $t \in \mathcal{T}_s^{(i)}$*, are defined by:*

15.1. $\forall i \notin \eta^{(t)}$
$Q_{t^+}^{(i)} = I_{|\mathcal{P}^{(i)}|}$;
15.2. $\forall p^{(\iota^{(t)})}, q^{(\iota^{(t)})} \in \mathcal{P}^{(\iota^{(t)})}$ *such that* $q^{(\iota^{(t)})} \in succ_t(p^{(\iota^{(t)})})$
$Q_{t^+}^{(\iota^{(t)})}(p^{(\iota^{(t)})}, q^{(\iota^{(t)})}) = \tau_t(p^{(\iota^{(t)})}, q^{(\iota^{(t)})})$;
15.3. $\forall i \in \eta^{(t)}$ *such that* $i \neq \iota^{(t)}$, $\forall p^{(i)}, q^{(i)} \in \mathcal{P}^{(i)}$ *such that* $q^{(i)} \in succ_t(p^{(i)})$
$Q_{t^+}^{(i)}(p^{(i)}, q^{(i)}) = 1$;
15.4. $\forall i \in \eta^{(t)}$, $\forall p^{(i)}, q^{(i)} \in \mathcal{P}^{(i)}$ *such that* $q^{(i)} \notin succ_t(p^{(i)})$
$Q_{t^+}^{(i)}(p^{(i)}, q^{(i)}) = 0.$

Definition 16. *Tensor elements* $Q_{t^-}^{(i)}$*, which represent the adjustment of synchronized transition* $t \in \mathcal{T}_s^{(i)}$*, are defined by:*

16.1. $\forall i \notin \eta^{(t)}$
$Q_{t^-}^{(i)} = I_{|\mathcal{P}^{(i)}|}$;
16.2. $\forall p^{(\iota^{(t)})} \in \mathcal{P}^{(\iota^{(t)})}$

$$Q_{t^-}^{(\iota^{(t)})}(p^{(\iota^{(t)})}, p^{(\iota^{(t)})}) = \begin{cases} 0 & \text{if } \nexists q^{(\iota^{(t)})} \in succ_t(p^{(\iota^{(t)})}) \\ -\tau_t(p^{(\iota^{(t)})}, q^{(\iota^{(t)})}) & \text{if } \exists q^{(\iota^{(t)})} \in succ_t(p^{(\iota^{(t)})}) \end{cases}$$

16.3. $\forall i \in \eta^{(t)}$, $i \neq \iota^{(t)}$ *and* $\forall p^{(i)} \in \mathcal{P}^{(i)}$

$$Q_{t^-}^{(i)}(p^{(i)}, p^{(i)}) = \begin{cases} 0 & \text{if } \nexists q^{(i)} \in succ_t(p^{(i)}) \\ 1 & \text{if } \exists q^{(i)} \in succ_t(p^{(i)}) \end{cases}$$

16.4. $\forall i \in \eta^{(t)}$, $\forall p^{(i)}, q^{(i)} \in \mathcal{P}^{(i)}$ *and* $p^{(i)} \neq q^{(i)}$
$Q_{t^-}^{(i)}(p^{(i)}, q^{(i)}) = 0.$

Definition 17. *Infinitesimal generator* Q *corresponding to the Markov chain associated to a decomposed GSPN model is represented by tensor formula called* Markovian Descriptor*:*

$$Q = \bigoplus_{i=1}^{N}{}_g \, Q_l^{(i)} + \sum_{t \in \mathcal{T}_s} \left(\bigotimes_{i=1}^{N}{}_g \, Q_{t^+}^{(i)} + \bigotimes_{i=1}^{N}{}_g \, Q_{t^-}^{(i)} \right) \tag{1}$$

Once a tensor sum is equivalent to a sum of particular product tensors, the *Markovian Descriptor* may be represented as:

$$Q = \sum_{j=1}^{(N+2|\mathcal{T}_s|)} \bigotimes_{i=1}^{N}{}_{g} Q_j^{(i)}, \tag{2}$$

$$\text{where } Q_j^{(i)} = \begin{cases} I_{|\mathcal{P}^{(i)}|} & \text{if } j \leq N \text{ and } j \neq i \\ Q_l^{(i)} & \text{if } j \leq N \text{ and } j = i \\ Q^{(i)}_{t^+_{(j-N)}} & \text{if } N < j \leq (N+ \mid \mathcal{T}_s \mid) \\ Q^{(i)}_{t^-_{(j-(N+|\mathcal{T}_s|))}} & \text{if } j > (N+ \mid \mathcal{T}_s \mid) \end{cases}$$

5 Choosing a Decomposition

In this section, we present several approaches to decompose a GSPN model. We show the necessary steps to obtain all components $\mathcal{SSM}$. Afterwards, we comment about the side effect of guards and its consequences.

Fig. 2 presents an example of a GSPN model. Based on this model, we show our decomposition technique applied in three different approaches. For all the possible decomposition approaches, the demonstration in Section 4.2 can be used to obtain the equivalent tensor algebra representation automatically.

5.1 Decomposing by Places

Firstly, we analyse a quite naive approach, which is based on decomposing a GSPN model by *places*. Each place has a maximum number of tokens K, and so we can view each place as a $\mathcal{SSM}$ with $K+1$ states. A decomposed GSPN model by *places* of Fig. 2 is presented in Fig. 3.

Each component $\mathcal{SSM}^{(i)}$ represents the possible states of place p_i of a GSPN model. Note that places p_2 and p_5 in Fig. 2 are 2-bounded, *i.e.*, there are no

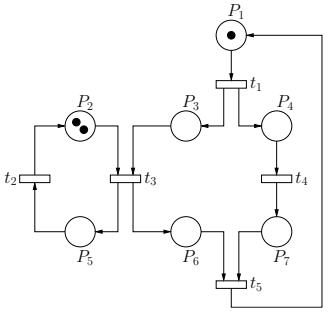


Fig. 2. An example of a GSPN model

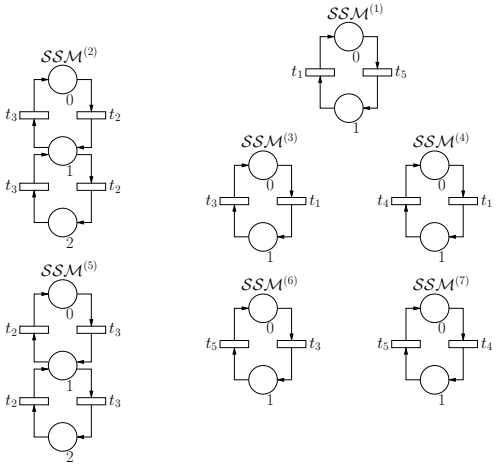


Fig. 3. Decomposed GSPN model by *places*

more than 2 tokens in each place in any marking $M \in \mathcal{RS}$. Hence $\mathcal{SSM}^{(2)}$ and $\mathcal{SSM}^{(5)}$ have three places which represent the states (0, 1 and 2) of places p_2 and p_5 respectively. Analogously, places p_1, p_3, p_4, p_6, and p_7 are 1-bounded. So $\mathcal{SSM}^{(1)}$, $\mathcal{SSM}^{(3)}$, $\mathcal{SSM}^{(4)}$, $\mathcal{SSM}^{(6)}$, and $\mathcal{SSM}^{(7)}$ have two states (0 and 1).

5.2 Decomposing by P-Invariants

Other decomposition of GSPN models is based on *P-invariants*. A P-invariant is composed of a set of places with constant token count. Fig. 4 presents a decomposed model of Fig. 2 using the *P-invariants* concept.

There are three minimal solutions of σ given by equation $\sigma C = 0$ (see Definition 8). So we can define three P-invariants to GSPN model of Fig. 2. $\mathcal{PI}_1$ has two places (p_2 and p_5), $\mathcal{PI}_2$ has three places (p_1, p_3 and p_6), and $\mathcal{PI}_3$ also has three places (p_1, p_4 and p_7).

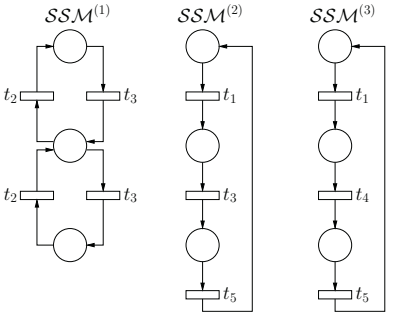


Fig. 4. Decomposed GSPN model by *P-invariants*

Each $\mathcal{PI}_i$ corresponds to a component $\mathcal{SSM}^{(i)}$ of the GSPN model. So, in this example, we can decompose the GSPN model in three components $\mathcal{SSM}$. It is important to note that, besides the transition superposition, there is a place superposition between components $\mathcal{SSM}^{(2)}$ and $\mathcal{SSM}^{(3)}$.

5.3 Decomposing as Superposed GSPN

Another possible approach to decompose a GSPN model is through transition superposition proposed by Donatelli [17]. Donatelli proposed the SGSPN formalism in which components (subsystems) interact each other through transition superposition.

Example of Fig. 2 can be decomposed by SGSPN, since there is a transition t_3 which synchronizes two components $\mathcal{GSPN}$. Component $\mathcal{GSPN}^{(1)}$ is composed of two places (p_2 and p_5), whereas component $\mathcal{GSPN}^{(2)}$ is composed of five places (p_1, p_3, p_4, p_6, and p_7). Once defined the components $\mathcal{GSPN}^{(i)}$, it is possible to obtain the equivalent components $\mathcal{SSM}^{(i)}$. Fig. 5 presents the equivalent components $\mathcal{SSM}$ of the GSPN model (Fig. 2).

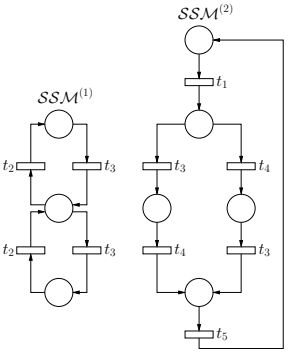


Fig. 5. Decomposed GSPN model by *SGSPN*

5.4 Decomposing Arbitrarily

It is also possible to decompose a GSPN model according to an arbitrarily chosen *semantic*. We can decompose the GSPN model of Fig. 2 as follows: markings of places p_1, p_2, p_3, and p_4, as well as markings of places p_5, p_6, and p_7. Hence component $\mathcal{SSM}^{(1)}$ is obtained from distinct markings of places p_1, p_2, p_3, and p_4 of $\mathcal{TRG}$, and component $\mathcal{SSM}^{(2)}$ is also obtained from distinct markings of places p_5, p_6, and p_7 of $\mathcal{TRG}$.

Note that the decomposition choice may privilege some features of the tensor format which is important to the solution method. In some cases, it may be important to decompose a GSPN model considering: a large or small number of components $\mathcal{SSM}$; a small number of reachable states; or even the difference between reachable and unreachable states.

5.5 Side Effect of Guards

The concept of *guards* in the GSPN formalism allows transition firing dependency according to the number of tokens in each place. Guards in GSPN are quite similar to functional elements in the SAN formalism [26, 18]. A natural decomposition among components $\mathcal{GSPN}$ of a GSPN model can be viewed through the use of guards. Therefore, guards in the GSPN models can allow to produce disconnected GSPN models, which have synchronization through the guards on theirs transitions.

As shown in Section 4.2, tensor format (*Markovian Descriptor*) of a decomposed GSPN model uses generalized tensor sum and products. GTA operators in the Markovian descriptor are used to represent the functional rates of transitions, but as long as there are no guards defined to transitions, they can be classical tensor products. Hence, guards on transitions are evaluated in *Markovian Descriptor* by GTA operators.

Another consequence of the use of guards is the possibility to define GSPN models with "disconnected parts", *i.e.*, models where there is not only a single net, but two or more nets with no arcs connecting them. In this case, there must be guards referring to places of other components in order to establish an interdependency (not a synchronization) among parts. The last example (Section 6.3) shows a net with disconnected parts and the use of guards to establish the interdependency.

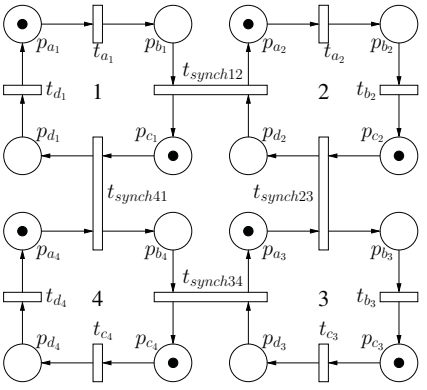
6 Modeling Examples

We now present three modeling examples in order to present the approaches discussed in the previous section. The first one presents a *Structured* model, the second one describes a *Simultaneous Synchronized Tasks* model, whereas the last one shows a *Resource Sharing* model.

6.1 Structured Model

Fig. 6 presents an example of a *Structured* model composed of four submodels. The submodel i is composed of four places (p_{a_i}, p_{b_i}, p_{c_i}, p_{d_i}) and two local transitions. There are also four synchronized transitions responsible for the synchronization of the submodels. It is important to observe that guards were chosen to define the possible firing sequence of transitions. This model was introduced by Miner [20].

In this model, the decomposition by *places* is rather catastrophic, since there is a correlation among marking of places. It results in a quite large product state space (65,536 states) for a rather small reachable state space (only 486 states). According to the SGSPN and P-invariants approaches, we have exactly the same decomposition and a more reasonable product state space (1,296 states). As a general conclusion one may discard the decomposition by places approach, but this is not really a fair conclusion, since this model is quite particular. Models



Guards
$g_{a_1} = (tk(p_{b_1}) == 0)$
$g_{d_1} = (tk(p_{a_1}) == 0)$
$g_{a_2} = (tk(p_{b_2}) == 0)$
$g_{b_2} = (tk(p_{c_2}) == 0)$
$g_{b_3} = (tk(p_{c_3}) == 0)$
$g_{c_3} = (tk(p_{d_3}) == 0)$
$g_{c_4} = (tk(p_{d_4}) == 0)$
$g_{d_4} = (tk(p_{a_4}) == 0)$
$g_{12} = ((tk(p_{c_1}) == 0) \& (tk(p_{a_2}) == 0))$
$g_{23} = ((tk(p_{d_2}) == 0) \& (tk(p_{b_3}) == 0))$
$g_{34} = ((tk(p_{a_3}) == 0) \& (tk(p_{c_4}) == 0))$
$g_{41} = ((tk(p_{d_1}) == 0) \& (tk(p_{b_4}) == 0))$

Fig. 6. Example of a structured model

with places with a larger bound (nets with more tokens) may be more interesting, as the next example will demonstrate.

6.2 Simultaneous Synchronized Tasks

Fig. 7 describes a *Simultaneous Synchronized Tasks* (SST) model in which five tasks are modeled. Such tasks have synchronization behavior among them, and those synchronization behaviors occur in different levels of the task execution.

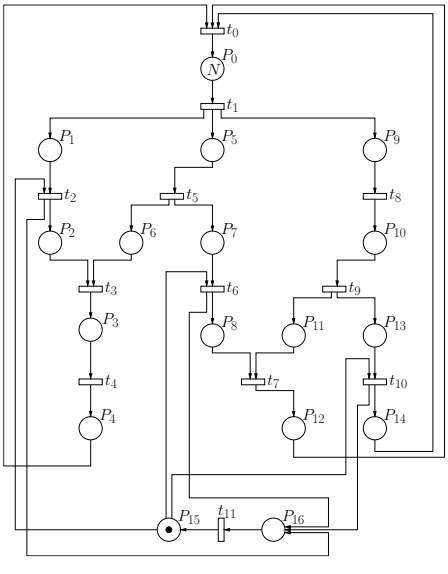


Fig. 7. Simultaneous Synchronized Tasks

Table 1. Indices of decomposed SST models

N	approach	$\#SSM$	*SSM sizes*	*pss*	*rss*	*mem*
1	Places	17	$(2 \times \cdots \times 2 \times 2 \times 2)$	1.31×10^5	98	1 KB
	P-Inv	6	$(5 \times 5 \times 5 \times 5 \times 5 \times 2)$	6.25×10^3	98	1 KB
	SGSPN	2	(49×2)	9.80×10^1	98	1 KB
3	Places	17	$(4 \times \cdots \times 4 \times 2 \times 2)$	4.29×10^9	12,100	2 KB
	P-Inv	6	$(35 \times 35 \times 35 \times 35 \times 35 \times 2)$	1.05×10^8	12,100	7 KB
	SGSPN	2	$(6,050 \times 2)$	1.21×10^4	12,100	236 KB
9	Places	17	$(10 \times \cdots \times 10 \times 2 \times 2)$	4.00×10^{15}	22,391,512	6 KB
	P-Inv	6	$(715 \times \cdots \times 715 \times 2)$	3.74×10^{14}	22,391,512	201 KB
	SGSPN	2	$(11,195,756 \times 2)$	$2.24 \times 10'$	22,391,512	672 MB
10	Places	17	$(11 \times \cdots \times 11 \times 2 \times 2)$	1.67×10^{16}	51,887,550	6 KB
	P-Inv	6	$(1,001 \times \cdots \times 1,001 \times 2)$	2.01×10^{15}	51,887,550	288 KB
	SGSPN	2	$(25,943,775 \times 2)$	$5.19 \times 10'$	51,887,550	-
20	Places	17	$(21 \times \cdots \times 21 \times 2 \times 2)$	2.72×10^{20}	18,994,747,662	12 KB
	P-Inv	6	$(10,626 \times \cdots \times 10,626 \times 2)$	2.71×10^{20}	18,994,747,662	3 MB
	SGSPN	2	$(9,497,373,831 \times 2)$	1.90×10^{10}	18,994,747,662	-
21	Places	17	$(22 \times \cdots \times 22 \times 2 \times 2)$	5.48×10^{20}	29,368,986,350	13 KB
	P-Inv	6	$(12,650 \times \cdots \times 12,650 \times 2)$	6.48×10^{20}	29,368,986,350	4 MB
	SGSPN	2	$(14,684,493,175 \times 2)$	2.94×10^{10}	29,368,986,350	-
27	Places	17	$(28 \times \cdots \times 28 \times 2 \times 2)$	2.04×10^{22}	286,448,238,746	16 KB
	P-Inv	6	$(31,465 \times \cdots \times 31,465 \times 2)$	6.17×10^{22}	286,448,238,746	10 MB
	SGSPN	2	$(143,224,119,373 \times 2)$	2.86×10^{11}	286,448,238,746	-

Table 1 presents some indices to compare the decomposition alternatives. In this example, we use the following approaches: decomposing by Places (Section 5.1), decomposing by P-invariants (Section 5.2) and decomposing by Superposed GSPN (Section 5.3). N represents the number of tokens in place P_0. The number of $\mathcal{SSM}$ components, decomposed by all approaches, is indicated by *#SSM*. *SSM sizes* represents the number of states in each component $\mathcal{SSM}$. *Product State Space*, *Reachable State Space*, and memory needs to store the *Markovian Descriptor*[3] of the model are denoted by *pss*, *rss*, and *mem* respectively.

The first important phenomenon to observe in Table 1 is the increasing gains of the *P-invariant* and *Places* approaches achieved to models with large N values. For small N values, there is much waste in the product state space that is not significant compared to the memory savings, specially for the *Places* approach. Regarding the comparison between *SGSPN* and *P-invariant* approaches, the model with $N = 9$ is a turning point, since the memory savings are already quite significant. In fact, larger models could not even be generated using the SGSPN decomposition. The relationship between product and reachable state space for

[3] We do not take into account the memory needs to store neither the probability vector to compute solution, nor any special structure to represent the reachable state space.

decomposition based on P-invariants is considerably large, but we believe that an optimized solution for models with sparse reachable state space could take great advantage from this decomposition approach.

The second very impressive results taken from Table 1 is the very consistent gains of the *Places* approach. Even though the product state space waste is considerably large for smaller models (roughly one or two orders of magnitude), the difference between the product state space for the *P-invariant* and *Places* approaches becomes insignificant for the $N = 20$ model. Taking the model to its limits ($N = 21$ to 27) we observe an inversion, since the *Places* approach has a smaller product state space than the *P-invariant* approach.

6.3 Resource Sharing

Fig. 8 (a) shows a traditional example of a *Resource Sharing* (RS) model, which has N process sharing R resources. Each process i is composed of two places: S_i (*sleeping*) and U_i (*using*). Tokens in place RS represent the number of available resources, whereas they represent the number of using resources in place RU. Fig. 8 (b) is an equivalent model in which guards impose a restriction to the firing of each transition ta_i.

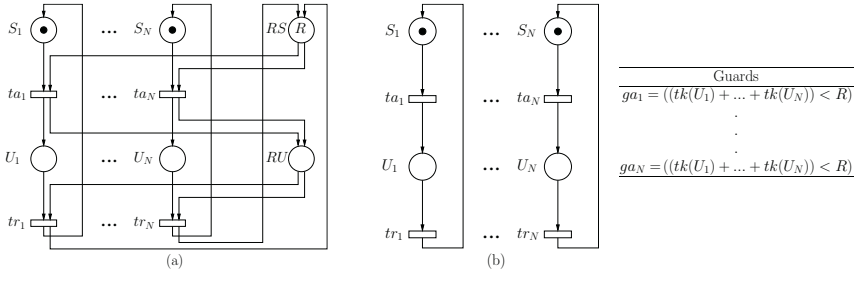


Fig. 8. (a) Resource Sharing without guards - (b) Resource Sharing with guards

Table 2 shows some indices to compare the use of guards in a GSPN model producing, in this case, an equivalent disconnected model. The indices for the model of Fig. 8 (a) are shown in the *without guards* rows, whereas indices for the model of Fig. 8 (b) are presented in the *with guards* rows. R indicates the number of tokens in place RS of Fig. 8 (a), as well as the number of available resources in the model of Fig. 8 (b). The computational cost (number of multiplications) to evaluate the product of a probability vector by the *Markovian Descriptor*[4], the memory needs and the CPU time to perform one single power iteraction are denoted by *c.c.*, *mem* and *time* respectively. The numerical results for those

[4] The vector-descriptor multiplication is the basic operation for most of the iterative solutions, *e.g.*, Power method, Uniformization [28].

Table 2. Indices of decomposed RS models ($N = 16$)

R	model	$\#SSM$	*SSM sizes*	*pss*	*rss*	*c.c.*	*mem*	*time*
1	without guards	17	$(2 \times \cdots \times 2 \times 2)$	131,072	17	1.34×10^8	6 KB	0.33 s
	with guards	16	$(2 \times \cdots \times 2)$	65,536	17	2.10×10^6	1 KB	0.45 s
3	without guards	17	$(2 \times \cdots \times 2 \times 4)$	262,144	697	2.73×10^8	8 KB	0.71 s
	with guards	16	$(2 \times \cdots \times 2)$	65,536	697	2.10×10^6	1 KB	0.49 s
9	without guards	17	$(2 \times \cdots \times 2 \times 10)$	655,360	50,643	6.88×10^8	14 KB	1.81 s
	with guards	16	$(2 \times \cdots \times 2)$	65,536	50,643	2.10×10^6	1 KB	0.53 s
15	without guards	17	$(2 \times \cdots \times 2 \times 16)$	1,048,576	65,535	1.10×10^9	20 KB	3.07 s
	with guards	16	$(2 \times \cdots \times 2)$	65,536	65,535	2.10×10^6	1 KB	0.53 s

examples were obtained on a 2.8 GHz Pentium IV Xeon under Linux operating system with 2 GBytes of memory.

The results in Table 2 show the decomposition based on P-invariants, since decomposition based on SGSPN could only be applied to the *without guards* model. Observe that SGSPN approach would result in exactly the same decomposition as P-invariants. The main conclusion observing this table is the absolute gains represented by the use of guards. It results in a model which has the same product state space independently from the number of resources, as well as the *pss* sizes are always smaller than those in the *without guards* models. It also has a smaller memory need and a more efficient solution (smaller computational cost).

It is a common mistake in some segments of the research community to assume that a model which requires functional evaluations (GTA operators) has a bigger CPU time to perform vector-descriptor multiplication than equivalent models described only with CTA operators. In fact, such use of guards gives to this GSPN model an efficiency as good as the one achieved by an equivalent SAN model [7]. Obviously, it happens due to the *Markovian Descriptor* representation using GTA.

7 Conclusion

The main contribution of this paper is to follow up the pioneer works of Ciardo and Trivedi [14], Donatelli [17], and Miner [20]. Our starting point is the assumption that for really large (and therefore structured) models the main difficulty is the storage of the infinitesimal generator. The solution techniques are rapidly evolving and many improvements, probably based on efficient parallel solutions, will soon enough be available. Using this assumption, we do believe that the tensor format based on Generalized Tensor Algebra has an important role to play.

For the moment, it benefits the Stochastic Automata Networks and it can also be applied to Generalized Stochastic Petri Nets. A natural future theoretical work is to expand those gains to other formalisms, such as PEPANETs [19]. A more immediate future work would be the integration of this decompo-

sition analysis to the solvers for SAN and GSPN (PEPS [6] and SMART [13] software tools respectively). It is easy to precompute the possible decomposition with their memory and computational costs. Therefore, the integration of such precalculation may automatically suggest the best decomposition approach according to the amount of memory available.

Finally, we would like to conclude stating that the use of tensor based storage can still give very interesting results and allows the solution (which cannot be done without the storage) of larger and larger models.

References

1. M. Ajmone-Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis. An Introduction to Generalized Stochastic Petri Nets. *Microelectronics and Reliability*, 31(4):699–725, 1991.
2. M. Ajmone-Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, 1984.
3. V. Amoia, G. De Micheli, and M. Santomauro. Computer-Oriented Formulation of Transition-Rate Matrices via Kronecker Algebra. *IEEE Transactions on Reliability*, R-30(2):123–132, 1981.
4. R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, 1960.
5. A. Benoit, L. Brenner, P. Fernandes, and B. Plateau. Aggregation of Stochastic Automata Networks with replicas. *Linear Algebra and its Applications*, 386:111–136, July 2004.
6. A. Benoit, L. Brenner, P. Fernandes, B. Plateau, and W. J. Stewart. The PEPS Software Tool. In *Computer Performance Evaluation / TOOLS 2003*, volume 2794 of *LNCS*, pages 98–115, Urbana, IL, USA, 2003. Springer-Verlag Heidelberg.
7. L. Brenner, P. Fernandes, and A. Sales. The Need for and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. *International Journal of Simulation: Systems, Science & Technology*, 6(3-4):52–60, February 2005.
8. J. W. Brewer. Kronecker Products and Matrix Calculus in System Theory. *IEEE Transactions on Circuits and Systems*, CAS-25(9):772–780, 1978.
9. P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 13(3):203–222, 2000.
10. P. Buchholz and T. Dayar. Block SOR for Kronecker structured representations. *Linear Algebra and its Applications*, 386:83–109, July 2004.
11. P. Buchholz and P. Kemper. Hierarchical reachability graph generation for Petri nets. *Formal Methods in Systems Design*, 21(3):281–315, 2002.
12. G. Ciardo, M. Forno, P. L. E. Grieco, and A. S. Miner. Comparing implicit representations of large CTMCs. In 4^{th} *International Conference on the Numerical Solution of Markov Chains*, pages 323–327, Urbana, IL, USA, September 2003.
13. G. Ciardo, R. L. Jones, A. S. Miner, and R. Siminiceanu. SMART: Stochastic Model Analyzer for Reliability and Timing. In *Tools of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pages 29–34, Aachen, Germany, September 2001.

14. G. Ciardo and K. S. Trivedi. A Decomposition Approach for Stochastic Petri Nets Models. In *Proceedings of the* 4^{th} *International Workshop Petri Nets and Performance Models*, pages 74–83, Melbourne, Australia, December 1991. IEEE Computer Society.
15. M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, 1981.
16. S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18:21–36, 1993.
17. S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In R. Valette, editor, *Proceedings of the* 15^{th} *International Conference on Applications and Theory of Petri Nets*, pages 258–277. Springer-Verlag Heidelberg, 1994.
18. P. Fernandes, B. Plateau, and W. J. Stewart. Efficient descriptor - Vector multiplication in Stochastic Automata Networks. *Journal of the ACM*, 45(3):381–414, 1998.
19. J. Hillston and L. Kloul. An Efficient Kronecker Representation for PEPA models. In L. de Alfaro and S. Gilmore, editors, *Proceedings of the first joint PAPM-PROBMIV Workshop)*, pages 120–135, Aachen, Germany, September 2001. Springer-Verlag Heidelberg.
20. A. S. Miner. *Data Structures for the Analysis of Large Structured Markov Models*. PhD thesis, The College of William and Mary, Williamsburg, VA, 2000.
21. A. S. Miner. Efficient solution of GSPNs using Canonical Matrix Diagrams. In 9^{th} *International Workshop on Petri Nets and Performance Models (PNPM'01)*, pages 101–110, Aachen, Germany, September 2001. IEEE Computer Society Press.
22. A. S. Miner and G. Ciardo. Efficient Reachability Set Generation and Storage Using Decision Diagrams. In *Proceedings of the* 20^{th} *International Conference on Applications and Theory of Petri Nets*, volume 1639 of *LNCS*, pages 6–25, Williamsburg, VA, USA, June 1999. Springer-Verlag Heidelberg.
23. A. S. Miner, G. Ciardo, and S. Donatelli. Using the exact state space of a Markov model to compute approximate stationary measures. In *Proceedings of the 2000 ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems*, pages 207–216, Santa Clara, California, USA, June 2000. ACM Press.
24. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
25. B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proceedings of the 1985 ACM SIGMETRICS conference on Measurements and Modeling of Computer Systems*, pages 147–154, Austin, Texas, USA, 1985. ACM Press.
26. B. Plateau and K. Atif. Stochastic Automata Networks for modelling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.
27. W. Reisig. *Petri nets: an introduction*. Springer-Verlag Heidelberg, 1985.
28. W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.