FACULDADE DE INFORMÁTICA
PUCRS - Brazil

http://www.pucrs.br/inf/pos/

A Framework to Decompose GSPN models

L. Brenner, P. Fernandes, A. Sales, T. Webber

Technical Report Series

Number 038
November, 2003

Contact:
lbrenner@inf.pucrs.br
www.inf.pucrs.br/~lbrenner

paulof@inf.pucrs.br
www.inf.pucrs.br/~paulof

asales@inf.pucrs.br
www.inf.pucrs.br/~asales

twebber@inf.pucrs.br
www.inf.pucrs.br/~twebber

Published by PPGCC - FACIN - PUCRS
Av. Ipiranga, 6681
90619-900 Porto Algre - RS - Brasil

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 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 formalism [3] 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 works [13] brought a first approach that could be employed to decompose a single model into components. However, their approach do not mention any specific storage or solution technique to numerically suitable. The need of a theoretical tool to represent such structured models naturally lead to *Tensor Algebra* representations [6, 9, 4, 14]. The term *Tensor Algebra* is being employed in this paper, but many authors still prefer the classic denomination *Kronecker Algebra* chosen in honor of the famous german mathematician of the XIX century.

The first complete approaches in this direction were the works of Donatelli in the SGSPN - Superposed Generalized Stochastic Petri Nets [15, 16]. 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, by using the techniques developed to the SAN - Stochastic Automata Networks [5].

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 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 the SAN formalism. At this time, the solution through the shuffle algorithm used in the SAN formalism [18, 8] presents an efficient solution with reasonable memory needs. Evidently, the use of other structured storage and solution techniques instead of the tensor algebra also present 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 tensor algebra based techniques are still worthy [17], at least considering the new improvements to handle tensor structures [7, 10].

This paper presents a reflexive study about the decomposition by 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 [16], but we extend the application scope following the ideas firstly advanced in [13] and employed latter in [20]. The new contribution of our work is justified by the numerical impact of the decomposition choices on the storage demands. In addition, we point out the underestimated benefits of the use of guards in GSPN formalism, which can be clearly demonstrated by the tensor format proposed in our work.

We do not pay much attention in this paper on the computational cost to solve the tensor representations. The recent evolutions in pure tensor solutions [7, 10], the promising ideas of parallel implementations, and the MDD and MxD techniques [11] 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, called *well-defined* GSPN, 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 [4, 14] and Generalized Tensor Algebra [24, 18] are presented.

2.1 CTA - Classical Tensor Algebra

Define two matrices A and B as follows:

$$A = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \qquad B = \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \end{pmatrix}$$

The *tensor product* $C = A \otimes B$ is given by

$$C = \begin{pmatrix} a_{00}B & a_{01}B \\ a_{10}B & a_{11}B \end{pmatrix}$$

In general, to define the tensor product of two matrices: A of dimensions $(\rho_1 \times \gamma_1)$ and B of dimensions $(\rho_2 \times \gamma_2)$; it is convenient to observe that the tensor product matrix has dimensions $(\rho_1\rho_2 \times \gamma_1\gamma_2)$ and 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 product $C = A \otimes B$ 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 sum $C = A \oplus B$ is defined as:

$$C_{[ik][jl]} = a_{ij}\delta_{kl} + b_{kl}\delta_{ij},$$

where δ_{ij} is the element of i^{th} row and j^{th} column of an identity matrix defined as:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Some important properties of tensor product and sum operations defined by Davio [4, 14] are:

- Associativity:
  $A \otimes (B \otimes C) = (A \otimes B) \otimes C$ and $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
- Distributivity over (ordinary matrix) addition:
  $(A + B) \otimes (C + D) = (A \otimes C) + (B \otimes C) + (A \otimes D) + (B \otimes D)$
- Compatibility with (ordinary matrix) multiplication:
  $(A \times B) \otimes (C \times D) = (A \otimes C) \times (B \otimes D)$
- Compatibility over multiplication:
  $A \otimes B = (A \otimes I_{n_B}) \times (I_{n_A} \otimes B)$
- Commutativity of normal factors[1]:
  $(A \otimes I_{n_B}) \times (I_{n_A} \otimes B) = (I_{n_A} \otimes B) \times (A \otimes I_{n_B})$

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

[1] Although this property could be inferred from the *Compatibility with (ordinary matrix) multiplication*, it was defined by Fernandes, Plateau and Stewart [18].

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 generalized tensor product $C = A(\mathcal{B}) \underset{g}{\otimes} B(\mathcal{A})$ is defined as:

$$C_{[ik][jl]} = a_{ij}(b_k) b_{kl}(a_i).$$

Generalized tensor sum is also analogous to the ordinary tensor sum, and is denoted by $\underset{g}{\oplus}$. The elements of the generalized tensor sum $C = A(\mathcal{B}) \underset{g}{\oplus} B(\mathcal{A})$ are defined as:

$$C_{[ik][jl]} = a_{ij}(b_k)\delta_{kl} + b_{kl}(a_i)\delta_{ij}.$$

GTA properties defined by Fernandes, Plateau and Stewart [18] are:

- Associativity:
  $[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\otimes} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B})\ ]$
  $[\ A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} B(\mathcal{A},\mathcal{C})\ ] \underset{g}{\oplus} C(\mathcal{A},\mathcal{B}) = A(\mathcal{B},\mathcal{C}) \underset{g}{\oplus} [\ B(\mathcal{A},\mathcal{C}) \underset{g}{\oplus} C(\mathcal{A},\mathcal{B})\ ]$
- Distributivity over addition:
  $[A(\mathcal{C},\mathcal{D}) + B(\mathcal{C},\mathcal{D})] \underset{g}{\otimes} [C(\mathcal{A},\mathcal{B}) + D(\mathcal{A},\mathcal{B})] = A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) + A(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B}) +$
  $B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} C(\mathcal{A},\mathcal{B}) \ + \ B(\mathcal{C},\mathcal{D}) \underset{g}{\otimes} D(\mathcal{A},\mathcal{B})$
- Compatibility over multiplication I:
  $A \underset{g}{\otimes} B(\mathcal{A}) = I_{n_A} \underset{g}{\otimes} B(\mathcal{A}) \ \times \ A \underset{g}{\otimes} I_{n_B}$
- Compatibility over multiplication II:
  $A(\mathcal{B}) \underset{g}{\otimes} B = A(\mathcal{B}) \underset{g}{\otimes} I_{n_B} \ \times \ I_{n_A} \underset{g}{\otimes} B$
- Decomposability of Generalized Tensor Product into Ordinary Tensor Product:

$$A \underset{g}{\otimes} B(\mathcal{A}) = \sum_{k=1}^{n_A} \ell_k(A) \otimes B(a_k)$$

3 Generalized Stochastic Petri Nets

We briefly present in this section an informal and a formal definition of *Generalized Stochastic Petri Nets* (GSPN) formalism.

3.1 Informal Definition

GSPN formalism [3] is a performance analysis tool on the graphical system representation typical of Petri Nets [25, 23].

GSPN formalism is derived from 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. 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, are the only restrictions to enable the firing of a given transition.

3.2 Formal Definition

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. π: $\mathcal{T} \rightarrow \{0, 1\}$ 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.

Let

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

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 $p \in \mathcal{P}$. This condition is a function with domain on tokens of $p \in \mathcal{P}$ and counter-domain on* false *and* true.

A condition c allows the firing of a transition with token dependency on a specific place.

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 $\mathcal{T}_T \cap \mathcal{T}_I = \emptyset$.*

3.2.1 Well-defined GSPN.

Definition of a GSPN model cannot be ambiguous. Hence, Restriction 1 must be respected. GSPN models that obey this restriction are called *well-defined* GSPN.

Restriction 1 *A $\mathcal{GSPN}$ is well-defined, if and only if:*

1.1. $\forall t \in \mathcal{T}$, $\exists p \in \mathcal{P}$ *such that* $p \in I(t)$;

1.2. $\forall t \in \mathcal{T}$, $\exists p \in \mathcal{P}$ *such that* $p \in O(t)$;

1.3. set of tangible markings is finite.

3.2.2 SGSPN.

SGSPN formalism was proposed by Donatelli [16]. A SGSPN model is composed of a set of standard GSPN models that interact only through a set of synchronized transitions. A SGSPN model has restrictions to components $\mathcal{GSPN}^{(i)}$ and synchronized transitions.

Restriction 2 *A component $\mathcal{GSPN}^{(i)}$ must obey the following restrictions:*

2.1. if $N > 1$, *set of synchronized transitions* $\mathcal{T}_s^{(i)}$ *must not be empty, or if there is only one component* $\mathcal{GSPN}^{(i)}$ *in the set of components* $\mathcal{GSPN}$ *(*$N = 1$*), set of synchronized transitions* $\mathcal{T}_s^{(i)}$ *must be empty;*

2.2. $\forall t \in \mathcal{T}^{(i)}$, $\exists p \in \mathcal{P}^{(i)}$ *such that* $p \in I^{(i)}(t)$;

2.3. $\forall t \in \mathcal{T}^{(i)}$, $\exists p \in \mathcal{P}^{(i)}$ *such that* $p \in O^{(i)}(t)$;

2.4. set of tangible markings is finite;

2.5. in all markings exists at least one transition enabled lead to a distinct marking.

Restriction 3 *Synchronized transition* $t \in \mathcal{T}_s$ *must obey the following restrictions:*

3.1. t is a timed transition;

3.2. $\forall i, j \in [1..N]$ *such that* $i \neq j$,

$$|I^{(i)}(t)|=|I^{(j)}(t)| \text{ and } |O^{(i)}(t)|=|O^{(j)}(t)|.$$

4 Framework

We present in this section a framework to decompose well-defined GSPN models. Our decomposition technique is showed in Fig. 1.

The basic idea is to decompose a well-defined 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 well-defined GSPN. 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:

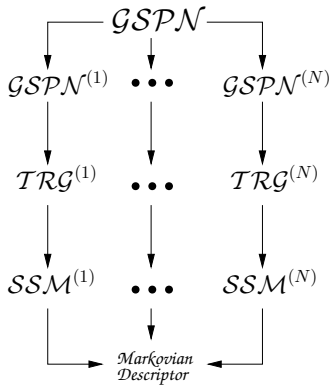


Figure 1: Decomposition technique

- 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).

In the next section, we define a component $\mathcal{GSPN}^{(i)}$ and its proprieties. 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 well-defined GSPN model into N components $\mathcal{GSPN}^{(i)}$ $(i \in [1..N])$.

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, where $\pi^{(i)}(t^{(i)}) = \pi(t^{(i)})$;

6.4. $I^{(i)}$ and $O^{(i)}$: $\mathcal{T}^{(i)} \rightarrow \mathcal{P}^{(i)*}$ input and output functions of the transitions, where $I^{(i)}(t^{(i)}) = I(t^{(i)})$ and $O^{(i)}(t^{(i)}) = O(t^{(i)})$;

6.5. $W^{(i)}$: $\mathcal{T}^{(i)} \rightarrow \mathbb{R}^{+}$ function that assigns a rate to each transition, where $W^{(i)}(t^{(i)}) = W(t^{(i)})$;

6.6. $G^{(i)}$*:* $\mathcal{T}^{(i)} \rightarrow \mathcal{C}^{(i)}$ *function guard that associates a necessary, but not sufficient, condition* $c^{(i)} \in \mathcal{C}^{(i)}$ *to the firing of each transition* $t^{(i)} \in \mathcal{T}^{(i)}$*, where* $G^{(i)}(t^{(i)}) = G(t^{(i)})$*;*

6.7. $M_0^{(i)}$*:* $\mathcal{P}^{(i)} \rightarrow \mathbb{N}$ *initial marking in each place* $p^{(i)} \in \mathcal{P}^{(i)}$*, where* $M_0^{(i)}(p^{(i)}) = M_0(p^{(i)})$*.*

Note that set of places $\mathcal{P}^{(i)}$ is a subset of $\mathcal{P}$, as well as set of transitions $\mathcal{T}^{(i)}$ is a subset of $\mathcal{T}$. Obviously, 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. Although the same restriction does not apply to $\mathcal{T}^{(i)}$, because it can be equal 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.

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 correspond element in that original GSPN (*e.g.* if $t^{(i)}$ correspond to t, then $W^{(i)}(t^{(i)})$ is equal to $W(t)$).

Definition 7 *A decomposed GSPN model composed of N components* $\mathcal{GSPN}^{(i)}$ *is defined by each component* $\mathcal{GSPN}^{(i)}$*, where* $i \in [1..N]$*.*

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 showed in Fig. 1, 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).

Let

C	incidence matrix of a GSPN (dimensions: $\lvert\mathcal{P}\rvert \times \lvert\mathcal{T}\rvert$);
c_{jk}	element from row j and column k of an incidence matrix.

Definition 8 *Elements of an incidence matrix C are defined by:*

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

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

Definition 9 *P-invariants of a GSPN are defined by vector solutions* σ *composed of non-negative integer:* 0 *and* 1*, given by equation* $\sigma C = 0$ *[25], 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* [2]. Therefore, a well-defined GSPN (Restriction 1) must have all places covered by P-invariants (all places are bounded) in order to have a finite set of tangible markings.

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

10.1. $M_0^{(i)} \in \mathcal{RS}^{(i)}(M_0^{(i)})$

10.2. $M_l^{(i)} \in \mathcal{RS}^{(i)}(M_0^{(i)})$*, 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 11 *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 12 *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:*

12.1. $\mathcal{TARC}^{(i)} \subseteq \mathcal{TRS}^{(i)}(M_0^{(i)}) \times \mathcal{TRS}^{(i)}(M_0^{(i)}) \times \mathcal{T}_T^{(i)} \times \mathcal{T}_I^{(i)*}$

12.2. $a_j^{(i)} = [M_k^{(i)}, M_l^{(i)}, t_0, \sigma] \in \mathcal{TARC}^{(i)}$*, if and only if* $M_k^{(i)}[t_0 > M_1^{(i)}, \sigma = t_1, \ldots, t_n, (n \geq 0)$ *and*

12.3. $\exists M_2^{(i)}, \ldots, M_n^{(i)}$ *such that* $M_1^{(i)}[t_1 > M_2^{(i)}[t_2 > \ldots M_n^{(i)}[t_n > M_l^{(i)}$

Definition 13 *A Stochastic State Machine (SSM) is defined by tuple $(\mathcal{P}, \mathcal{T}, F, \Lambda)$, where:*

13.1. $\mathcal{P}$ *set of non-empty places;*

13.2. $\mathcal{T}$ *set of non-empty transitions;*

13.3. $F \subseteq ((\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P}))$ *with* $dom(F) \cup codom(F) = \mathcal{P} \cup \mathcal{T}$ *is the flow relation. It has to satisfy the following restriction*[2]*:* $\forall t \in \mathcal{T}$*:* $|^\circ t| = |t^\circ| = 1$*;*

13.4. $\Lambda : \mathcal{T} \rightarrow \mathbb{R}^+$*, 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 $a_j^{(i)} \in \mathcal{TARC}^{(i)}$ corresponds to $[p_y^{(i)}, t^{(i)}] \in F^{(i)}$ and $[t^{(i)}, p_z^{(i)}] \in F^{(i)}$, if and only if exist $a_j^{(i)} = [M_k^{(i)}, M_l^{(i)}, t, \sigma]$ such that $M_k^{(i)}$ corresponds to $p_y^{(i)}$, $M_l^{(i)}$ corresponds to $p_z^{(i)}$, $t \in \mathcal{T}_T^{(i)}$, and $\sigma \in \mathcal{T}_I^{(i)*}$.

The transition rate of $t^{(i)} \in \mathcal{T}^{(i)}$ (obtained from $[M_k^{(i)}, M_l^{(i)}, t, \sigma]$) can be computed as $\Lambda(t).\Lambda(\sigma)$[3], 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 in two types: *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 14 *Transition* $t \in \mathcal{T}_l^{(i)}$ *is defined as local transition, if and only if* $\forall j \in [1..N]$ *such that* $i \neq j$*,* $t \notin \mathcal{T}_l^{(j)}$*.*

Definition 15 *Transition* t *is defined as synchronized transition, if and only if* $t \in \mathcal{T}_s^{(i)}$ *and* $\exists j \in [1..N]$ *such that* $i \neq j, t \in \mathcal{T}_s^{(j)}$*.*

Definition 16 *Set of synchronized transitions* $\mathcal{T}_s$ *of a decomposed GSPN model is defined as* $\mathcal{T}_s = \mathcal{T}_s^{(1)} \cup \mathcal{T}_s^{(2)} \cup \ldots \cup \mathcal{T}_s^{(N)}$*.*

Given Definition 14, 15, and 16, we can define the *Markovian Descriptor* of a decomposed GSPN model. *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 of the set of local transitions $\mathcal{T}_l^{(i)}$;

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

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

- $2|\mathcal{T}_s|$ tensors $Q_{t^+}^{(i)}$ and $Q_{t^-}^{(i)}$, which have all transition rates of the set of synchronized transitions $\mathcal{T}_s^{(i)}$.

Let

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

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

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

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

Definition 17 *Tensor elements $Q_l^{(i)}$, which represent all local transitions $t \in \mathcal{T}_l^{(i)}$ of component $\mathcal{SSM}^{(i)}$, are defined by:*

17.1. $\forall p_x^{(i)}, p_y^{(i)} \in \mathcal{P}^{(i)}$ *such that* $p_y^{(i)} \in succ_t(p_x^{(i)})$ *and* $p_x^{(i)} \neq p_y^{(i)}$
$Q_l^{(i)}(p_x^{(i)}, p_y^{(i)}) = \sum_{t \in \mathcal{T}_l^{(i)}} \tau_t(p_x^{(i)}, p_y^{(i)})$;

17.2. $\forall p_x^{(i)} \in \mathcal{P}^{(i)}$ *such that* $p_y^{(i)} \in succ_t(p_x^{(i)})$
$Q_l^{(i)}(p_x^{(i)}, p_x^{(i)}) = -\sum_{t \in \mathcal{T}_l^{(i)}} \tau_t(p_x^{(i)}, p_y^{(i)})$;

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

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 master[4] component $\mathcal{SSM}$ 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 18 *Tensor elements $Q_{t^+}^{(i)}$, which represent the occurrence of synchronized transition $t \in \mathcal{T}_s^{(i)}$, are defined by:*

18.1. $\forall i \notin \eta^{(t)}$
$Q_{t^+}^{(i)} = I_{|\mathcal{P}^{(i)}|}$;

[4]The master/slave semantic is used to the formal definition of synchronized transitions. However, almost any other semantics can be used without loss of generality.

18.2. $\forall p_x^{(\iota^{(t)})}, p_y^{(\iota^{(t)})} \in \mathcal{P}^{(\iota^{(t)})}$ *such that* $p_y^{(\iota^{(t)})} \in succ_t(p_x^{(\iota^{(t)})})$
$Q_{t^+}^{(\iota^{(t)})}(p_x^{(\iota^{(t)})}, p_y^{(\iota^{(t)})}) = \tau_t(p_x^{(\iota^{(t)})}, p_y^{(\iota^{(t)})})$;

18.3. $\forall i \in \eta^{(t)}$ *such that* $i \neq \iota^{(t)}$, $\forall p_x^{(i)}, p_y^{(i)} \in \mathcal{P}^{(i)}$ *such that* $p_y^{(i)} \in succ_t(p_x^{(i)})$
$Q_{t^+}^{(i)}(p_x^{(i)}, p_y^{(i)}) = 1$;

18.4. $\forall i \in \eta^{(t)}$, $\forall p_x^{(i)}, p_y^{(i)} \in \mathcal{P}^{(i)}$ *such that* $p_y^{(i)} \notin succ_t(p_x^{(i)})$
$Q_{t^+}^{(i)}(p_x^{(i)}, p_y^{(i)}) = 0$.

Definition 19 *Tensor elements* $Q_{t^-}^{(i)}$*, which represent the adjustment of synchronized transition* $t \in \mathcal{T}_s^{(i)}$*, are defined by:*

19.1. $\forall i \notin \eta^{(t)}$
$Q_{t^-}^{(i)} = I_{|\mathcal{P}^{(i)}|}$;

19.2. $\forall p_x^{(\iota^{(t)})} \in \mathcal{P}^{(\iota^{(t)})}$
$$Q_{t^-}^{(\iota^{(t)})}(p_x^{(\iota^{(t)})}, p_x^{(\iota^{(t)})}) = \begin{cases} 0 & \text{if } \nexists p_y^{(\iota^{(t)})} \in succ_t(p_x^{(\iota^{(t)})}) \\ -\tau_t(p_x^{(\iota^{(t)})}, p_y^{(\iota^{(t)})}) & \text{if } \exists p_y^{(\iota^{(t)})} \in succ_t(p_x^{(\iota^{(t)})}) \end{cases}$$

19.3. $\forall i \in \eta^{(t)}$, $i \neq \iota^{(t)}$ *and* $\forall p_x^{(i)} \in \mathcal{P}^{(i)}$
$$Q_{t^-}^{(i)}(p_x^{(i)}, p_x^{(i)}) = \begin{cases} 0 & \text{if } \nexists p_y^{(i)} \in succ_t(p_x^{(i)}) \\ 1 & \text{if } \exists p_y^{(i)} \in succ_t(p_x^{(i)}) \end{cases}$$

19.4. $\forall i \in \eta^{(t)}$, $\forall p_x^{(i)}, p_y^{(i)} \in \mathcal{P}^{(i)}$ *and* $p_x^{(i)} \neq p_y^{(i)}$
$Q_{t^-}^{(i)}(p_x^{(i)}, p_y^{(i)}) = 0$.

Definition 20 *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:

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

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

5 Choosing a decomposition

In this section, we present several approaches to decompose a well-defined 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 well-defined GSPN model. Based on this model, we show our decomposition technique applied in three different approaches. To all the possible decomposition approaches, the demonstration in Section 4.2 can be used to obtain the equivalent tensor algebra representation automatically.

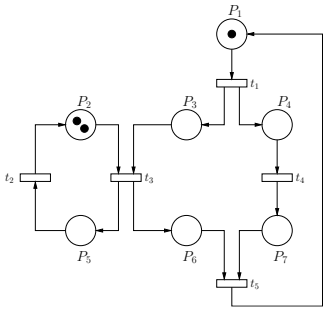


Figure 2: An example of a GSPN model

5.1 Decomposing by places

Firstly, we analyze a quite naive approach, which is based on decompose a GSPN model by *places*. Each place has a maximum number of tokens K, and so we can view each place as a 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 is no 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-invariants 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 is three minimal solution of σ given by equation $\sigma C = 0$ (see Definition 9). 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.

Each $\mathcal{PI}_i$ corresponds to a component $\mathcal{SSM}^{(i)}$ of GSPN model. So, in this example, we can decomposed the GSPN model in three components $\mathcal{SSM}$. Note that, besides the

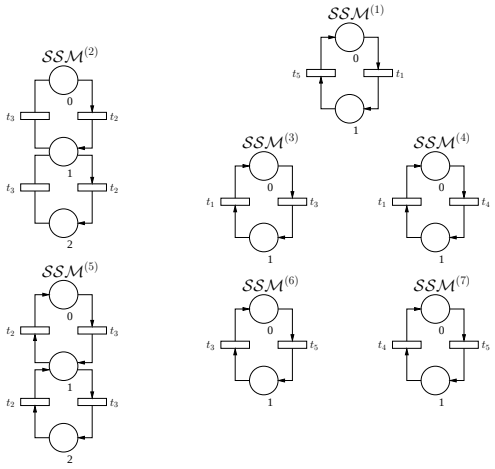


Figure 3: Decomposed GSPN model by *places*

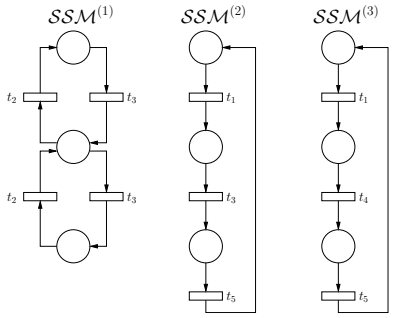


Figure 4: Decomposed GSPN model by *P-invariants*

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 [16]. Donatelli proposed the SGSPN formalism 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).

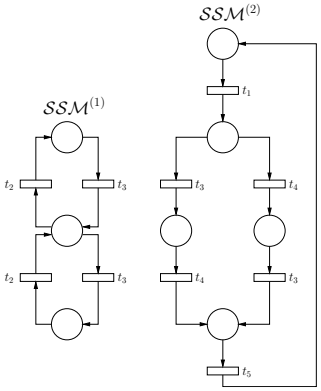


Figure 5: Decomposed GSPN model by *SGSPN*

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, and 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 $\mathcal{TRG}$, as well as 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; the difference between reachable and unreachable states.

5.5 Side effect of guards

The concept of *guards* in GSPN formalism allows marking dependency on places. Guards in GSPN formalism are quite similar to functional elements in SAN formalism [5, 18]. A natural decomposition among components $\mathcal{GSPN}$ of a GSPN model can be viewed through the use of guards. Therefore, guards in GSPN models can allow to produce disconnected GSPN models, which have synchronization through the guards on theirs transitions.

As showed 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 is no guards defined to transitions, they can be classical tensor products. Hence, guards on transitions are evaluated in *Markovian Descriptor* by GTA operators.

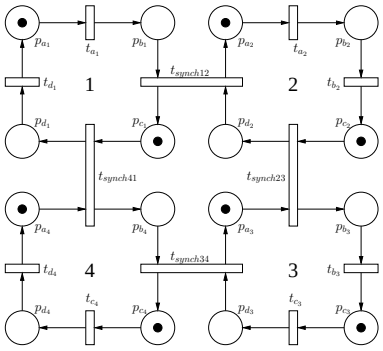
Another, disturbing to some, consequence of the use of guards is the possibility to define GSPN models with "disconnect parts", *i.e.*, models where there not only a single net, but two or more nets with no arcs connecting them. In this case, there must be guards referring to places of other components 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. The first one presents a *Structured* model. The second one describes a *Simultaneous Synchronized Tasks* model. And 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. Note that guards are defined to determine the possible firing sequence of transitions. This model was introduced by Miner [20].



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

Figure 6: Example of a structured model

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 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. Those tasks have synchronization behavior among them, and these synchronization behaviors occur in different levels of the task execution.

Table 1 presents some indices to compare the decomposition alternatives. In this example, we use the following approaches: decomposing by Superposed GSPN (Section 5.3) and decomposing by P-invariants (Section 5.2). N represents the number of tokens in place P_0. The number of $\mathcal{SSM}$ components, decomposed by both approaches, is indicated by *#SSM*. *SSM sizes* represents the number of states in each component $\mathcal{SSM}$. *Product State Space*,

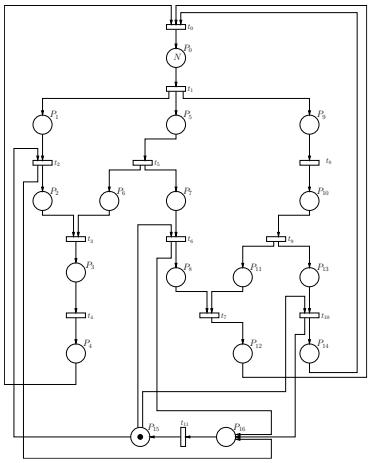


Figure 7: Simultaneous Synchronized Tasks

Reachable State Space, and memory needs to store the *Markovian Descriptor* of the model are denoted by *pss*, *rss*, and *mem* respectively.

N	approach	$\#SSM$	*SSM sizes*	*pss*	*rss*	*mem*
1	SGSPN	2	(49×2)	9.80×10^{1}	98	3 KB
	P-Inv	6	$(5 \times 5 \times 5 \times 5 \times 5 \times 2)$	6.25×10^{3}	98	1 KB
2	SGSPN	2	(738×2)	1.47×10^{3}	1,476	55 KB
	P-Inv	6	$(15 \times 15 \times 15 \times 15 \times 15 \times 2)$	1.51×10^{6}	1,476	4 KB
4	SGSPN	2	$(33,969 \times 2)$	6.79×10^{4}	67,938	3,458 KB
	P-Inv	6	$(70 \times 70 \times 70 \times 70 \times 70 \times 2)$	3.36×10^{9}	67,938	22 KB
8	SGSPN	2	$(4,469,388 \times 2)$	8.93×10^{6}	8,938,776	568 MB
	P-Inv	6	$(495 \times \cdots \times 495 \times 2)$	5.94×10^{13}	8,938,776	175 KB
10	SGSPN	2	$(25,943,775 \times 2)$	5.18×10^{7}	51,887,550	-
	P-Inv	6	$(1,001 \times \cdots \times 1.001 \times 2)$	2.01×10^{15}	51,887,550	375 KB
12	SGSPN	2	$(115,518,078 \times 2)$	2.31×10^{8}	231,036,156	-
	P-Inv	6	$(1,820 \times \cdots \times 1,820 \times 2)$	3.99×10^{16}	231,036,156	710 KB
15	SGSPN	2	$(763,842,674 \times 2)$	1.52×10^{9}	1,527,685,348	-
	P-Inv	6	$(3,876 \times \cdots \times 3,876 \times 2)$	1.74×10^{18}	1,527,685,348	2 MB
20	SGSPN	2	$(9,497,373,831 \times 2)$	1.89×10^{10}	18,994,747,662	-
	P-Inv	6	$(10,626 \times \cdots \times 10,626 \times 2)$	2.68×10^{20}	18,994,747,662	5 MB

Table 1: Indices of decomposed SST models

The decomposition based on places for this example is not indicated in Table 1. Nevertheless, some interesting facts can be observed for the largest model ($N = 20$), *e.g.*: the decomposition by places would reduce the large amount of memory needs from 5 MBytes to only 125 KBytes. Of course, it will also correspond to an increase of the product state space size from 2.68×10^{20} to 1.20×10^{23}, which is not that significative, considering the memory saves.

The second important phenomenon to observe in Table 1 is the increasing gains of the P-invariant approach achieved to models with large N values. For the small values of N, there

is much waste in the product state space that is not significant compared to the memory savings. The model with $N = 8$ is a turning point, since the memory savings becomes more significant. In fact, the really large models could not even be generated using the SGSPN decomposition. The relationship between product and reachable state space for decomposition based on P-invariants is considerably large, but we believe that an optimized solution to models with sparse reachable state space could take great advantage from this decomposition 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, where guards impose a restriction to the firing of each transition ta_i.

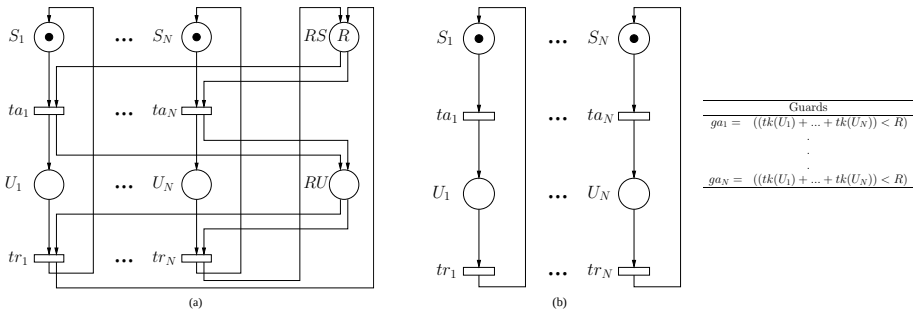


Figure 8: (a) Resource Sharing - (b) Resource Sharing with guards

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 of Fig. 8 (a) are showed in the *single net* field, whereas indices of Fig. 8 (b) are presented in the *with guards* field. 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*[5] is denoted by *c.c.*

The results in Table 2 shows the decomposition based on the P-invariants, since decomposition based on SGSPN could only be applied to single net model. Note that SGSPN approach would result on 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 and the *pss* sizes are always smaller than those in single net models. It also has smaller memory needs and a more efficient solution (smaller computational cost). In fact, such use of guards gives to this GSPN model an efficiency as good as the one achieved by an equivalent SAN model. Obviously, it happens due to the *Markovian Descriptor* representation using GTA.

[5]The vector-descriptor multiplication is the basic operation for most of the iterative solutions, *e.g.*, Power method, Uniformization [26].

R	model	$\#SSM$	$SSM\ sizes$	pss	rss	$c.c.$	mem
1	single net	17	$(2 \times \cdots \times 2 \times 2)$	$131,072$	17	1.34×10^8	18 KB
	with guards	16	$(2 \times \cdots \times 2)$	$65,536$	17	2.09×10^6	2 KB
5	single net	17	$(2 \times \cdots \times 2 \times 6)$	$393,216$	$6,885$	4.11×10^8	20 KB
	with guards	16	$(2 \times \cdots \times 2)$	$65,536$	$6,885$	2.09×10^6	2 KB
10	single net	17	$(2 \times \cdots \times 2 \times 11)$	$720,896$	$58,651$	7.57×10^8	23 KB
	with guards	16	$(2 \times \cdots \times 2)$	$65,536$	$58,651$	2.09×10^6	2 KB
15	single net	17	$(2 \times \cdots \times 2 \times 16)$	$1,048,576$	$65,535$	1.10×10^9	25 KB
	with guards	16	$(2 \times \cdots \times 2)$	$65,536$	$65,535$	2.09×10^6	2 KB

Table 2: Indices of decomposed RS models

7 Conclusion

The main contribution of this paper is to follow up the pioneer works of Ciardo and Trivedi [13], Donatelli [16], 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 can also be applied to Generalized Stochastic Petri Nets. To expand these gains to other formalisms, like PEPANETs [19], is a natural future theoretical work. A more immediate future work would be the integration of this decomposition analysis to the solvers for SAN and GSPN (PEPS [1] and SMART [12] 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 greater and greater models.

References

[1] PEPS: Performance Evaluation of Parallel Systems. http://www-apache.imag.fr/software/peps/.

[2] M. Ajmone-Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis. An Introduction to Generalized Stochastic Petri Nets. *Microelectronics Reliability*, 31(4):699–725, 1991.

[3] 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.

[4] 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.

[5] K. Atif and B. Plateau. Stochastic Automata Networks for modelling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108, 1991.

[6] R. Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, New York, 1960.

[7] A. Bennoit, L. Brenner, P. Fernandes, and B. Plateau. Aggregation of Stochastic Automata Networks with replicas. In 4^{th} *International Conference on the Numerical Solution of Markov Chains*, pages 145–166, Urbana, Illinois, USA, September 2003.

[8] L. Brenner, P. Fernandes, and A. Sales. The Need and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. Technical Report 37, Porto Alegre, Brazil, 2003.

[9] J. W. Brewer. Kronecker Products and Matrix Calculus in System Theory. *IEEE Transactions on Circuits and Systems*, CAS-25(9):772–780, 1978.

[10] P. Buchholz and T. Dayar. Block SOR for Kronecker structured representations. In *Proceedings of the 2003 International Conference on the Numerical Solution of Markov Chains*, pages 121–143, Urbana and Monticello, Illinois, USA, 2003. Springer-Verlag.

[11] 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, Illinois, USA, September 2003.

[12] G. Ciardo, A. S. Miner, III Robert L. Jones, A. S. Mangalam, R. Marmorstein, and R. I. Siminiceanu. SMART: Stochastic Model checking Analyzer for Reliability and Timing. http://www.cs.wm.edu/~ciardo/SMART/.

[13] 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.

[14] M. Davio. Kronecker Products and Shuffle Algebra. *IEEE Transactions on Computers*, C-30(2):116–125, 1981.

[15] S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18:21–36, 1993.

[16] S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. In *Proceedings of the* 15^{th} *International Conference on Applications and Theory of Petri Nets*, pages 258–277, Springer-Verlag, Berlin Heidelberg, 1994. R. Valette.

[17] S. Donatelli. Kronecker Algebra and (Stochastic) Petri Nets: Is It Worth the Effort? In *Proceedings of the* 22^{nd} *International Conference on Applications and Theory of Petri Nets*, pages 1–18, Springer-Verlag, Berlin Heidelberg, 2001. J.M. Colom and M. Koutny.

[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)*, Aachen, Germany, September 2001. Lecture Notes in Computer Science.

[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*, pages 6–25, Williamsburg, VA, USA, June 1999. Springer-Verlag.

[23] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[24] B. Plateau. *De l'Evaluation du Parallélisme et de la Synchronisation.* PhD thesis, Paris-Sud, Orsay, 1984.

[25] W. Reisig. *Petri nets: an introduction.* Springer-Verlag, Berlin, 1985.

[26] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.