Structured Solution of Asynchronously Communicating Stochastic Modules[*]

Javier Campos[†], Susanna Donatelli[‡], and Manuel Silva[†]

Abstract

Asynchronously Communicating Stochastic Modules (SAM) are Petri nets that can be seen as a set of modules that communicate through buffers, so they are not (yet another) Petri net subclass, but they complement a net with a structured view. *This paper considers the problem of exploiting the compositionality of the view to generate the state space and to find the steady-state probabilities of a stochastic extension of SAM in a* net-driven, efficient way.

Essentially, we give an expression of an auxiliary matrix, $\mathbf{G}$*, which is a supermatrix of the infinitesimal generator of a SAM.* $\mathbf{G}$ *is a tensor algebra expression of matrices of the size of the components for which it is possible to numerically solve the characteristic steady-state solution equation* $\boldsymbol{\pi} \cdot \mathbf{G} = \mathbf{0}$*, without the need to explicitly compute* $\mathbf{G}$*. Therefore, we obtain a method that computes the steady-state solution of a SAM without ever explicitly computing and storing its infinitesimal generator, and therefore without computing and storing the reachability graph of the system.*

Some examples of application of the technique are presented and compared to previous approaches.

Keywords: *Petri net models. Performance analysis. Structural decomposition. Kronecker algebra.*

1 Introduction and motivations

Stochastic Petri Nets (SPN's) [29] and *Generalized Stochastic Petri Nets* (GSPN's) [11, 1] are well-known interpreted extensions of autonomous Petri net (PN) models allowing the consideration of performance aspects in the design of complex concurrent systems, in addition to the PN's capability of functional validation. Product-form expressions and efficient algorithms for the computation of the steady-state distribution are known only for some particular classes of GSPN's [35, 25], so that, in general, numerical solution of the embedded Continuous Time Markov Chain (CTMC) must be performed to get exact performance indices. In this case the *state space explosion problem* may make the evaluation of large systems intractable due to the storage cost for the infinitesimal generator matrix and/or to the time complexity of solution algorithms.

Net-driven techniques reduce the memory and time complexity of solution algorithms by using information extracted from the structure of the net model. Examples of net-driven techniques are the

[*]This work has been developed within the project HCM CT94-0452 (MATCH) of the European Union.

[†]J. Campos and M. Silva are with the Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna, 3, 50015 Zaragoza, Spain; e-mails: {jcampos,silva}@posta.unizar.es.

[‡]S. Donatelli is with the Dip.to di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy; e-mail: susi@di.unito.it.

elimination of immediate transitions in GSPN [12], the exploitation of symmetries in Stochastic Well-Formed Coloured Nets [13], and the use of linear programming to compute performance bounds [7].

Net structure has also been used for approximation in a *divide and conquer* approach for the reduction of the state explosion problem. Examples can be found in the context of approximation techniques for the computation of throughput of *Marked Graphs* (MG) [8] or *Deterministically Synchronized Sequential Processes* (DSSP) [31]. Given a net model, a net-driven decomposition of the model is implemented in order to use a response time approximation algorithm.

Net structure also plays a relevant role in *tensor algebra* approaches that express the infinitesimal generator $\mathbf{Q}$ of an SPN in terms of $\mathbf{Q}_i$ matrices computed on some *components* (subnets with smaller state space) and allow the solution of the characteristic steady-state solution equation $\boldsymbol{\pi} \cdot \mathbf{Q} = \mathbf{0}$ without computing and storing $\mathbf{Q}$ [2, 20, 21, 6, 26]: this technique allows to move the storage bottleneck from the infinitesimal generator matrix to the probability vector. All the works for SPN cited above were inspired by the pioneering work of Plateau [32] on Stochastic Automata Networks. Since the tensor algebra method is based on components of the net, and therefore on the *structure* of the net, we shall often refer to it as a "structured solution method."

In the structured technique, a certain Cartesian product of reachable states of components is built, leading eventually to a *product space* PS that includes the actual *reachability set* RS: the main problem is that, in general, the product space can be much bigger than the actual state space. In other words, the PS may contain many non-reachable states (that we shall call *spurious*). According to this structured view of the state space, a tensor expression of $\mathbf{Q}$ in terms of $\mathbf{Q}_i$ matrices can be derived. Even if spurious states appear, the tensor algebra approach leads to exact solution [32, 21]. Nevertheless the storage and computational complexity may be increased in practice to the point that the advantages of the technique are lost.

A way to overcome the creation of spurious states is to generate an *abstract representation* of the full model that acts like a "filter" of spurious solutions. The abstract model —called here *basic skeleton*— constraints the product space PS, leading to a *restricted product space* RPS expressed as the union of the Cartesian product of subsets of the reachability sets of the components. The $\mathbf{Q}$ matrix has a block structure determined by the high level view, and a tensor expression for each block of $\mathbf{Q}$ in terms of blocks of $\mathbf{Q}_i$ can be derived.

We refer to the product space method as *flat*, or *single level*, and to the restricted product space method as *two levels*.

Two level techniques proved very effective in a number of cases. For certain net subclasses like MG no spurious states are generated so that equality between RS and RPS is obtained [6] (previously proved for response time approximation in [8]). In [9] the MG case is generalized to DSSP where RPS may strictly include RS.

In this paper, the extension of the two level approach for DSSP is developed for arbitrary but bounded SPN models for which a *SAM view* is given, where SAM stands for System of Asynchronously communicating Modules. A SAM view identifies a subset of places as *buffers* and a number of disjoint subnets called *modules*; the only exchange of information among modules is through buffers, so that SAM are a natural model for asynchronously communicating systems. A SAM view of the net can be provided by the model construction process, or it can be defined only for solution purposes.

Technically speaking, the abstract representation required by the two level approach is constructed using *implicit* places [17]. More precisely, in order to improve the efficiency of the algorithms we use the linear relaxation, i.e., those places that are also *structurally* implicit.

The contribution of the paper is two-fold. First, by removing the storage bottleneck of the infinitesimal generator the size of systems that can be solved is significantly increased. Second, by exploiting the net structure it provides a thorough view of the relationship between net structure and solution costs of the underlying stochastic process. The paper is based on a previous work [9] for the restricted DSSP class. A significant improvement is that the method presented in this paper applies to arbitrary SPN systems.

The paper is organised as follows: Section 2 reviews basic definitions and notations and overviews the literature on flat and two level structured solution for SPN subclasses. Section 3 defines the SAM view of SPN and the abstract models. The two level solution method is presented in Section 4 (for what concerns reachability graph construction), and in Section 5 (for what concerns the infinitesimal generator construction). Complexity issues are also discussed in Section 5. All the concepts are explained on a simple running example, while a more complete set of examples is presented in Section 6, that compares single level and two level approaches. Finally, possible extensions and concluding remarks are given in Section 7.

2 SPN's and structured solution methods

In this section, we present a conceptual framework in which the flat and the two level structured solution methods are considered. We overview different tensor algebra solutions for SPN that have appeared in the literature, with the goal of presenting them in a unified framework. Readers not familiar with the basics of tensor algebra can find them in the appendix. We assume that the reader is familiar with basic concepts of Petri nets [30, 37], therefore in the following only basic definitions are listed to establish notation.

2.1 Definitions

A *Petri net* (PN) is a 4–tuple $\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post}\rangle$, where P and T are disjoint sets of *places* and *transitions*, and **Pre** and **Post** are the *pre- and post-incidence functions* representing (in vector form) the input and output arcs: $\mathbf{Pre}[p,t] \in \mathbb{N}$ ($\mathbf{Post}[p,t] \in \mathbb{N}$). *Ordinary* nets are Petri nets whose pre- and post-incidence functions take values in $\{0,1\}$. The incidence function of a given arc in a non-ordinary net is called *weight* or *multiplicity*. The *pre-* and *post-set* of a transition $t \in T$ are defined respectively as ${}^\bullet t = \{p \mid \mathbf{Pre}[p,t] > 0\}$ and $t^\bullet = \{p \mid \mathbf{Post}[p,t] > 0\}$. The *pre-* and *post-set* of a place $p \in P$ are defined respectively as ${}^\bullet p = \{t \mid \mathbf{Post}[p,t] > 0\}$ and $p^\bullet = \{t \mid \mathbf{Pre}[p,t] > 0\}$. Transition t is a *join* (*fork*) if $|{}^\bullet t| > 1$ ($|t^\bullet| > 1$).

The *incidence matrix* of the net is defined as $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$. *Flows* (*semiflows*) are integer (natural) annullers of $\mathbf{C}$. Right and left annullers are called T- and P-(semi)flows respectively. A semiflow is *minimal* when its support is not a proper superset of the support of any other semiflow and the greatest common divisor of its elements is one. A net is *consistent* if it has a T-semiflow $\mathbf{x} \geq \mathbf{1}$. A net is *conservative* if it has a P-semiflow $\mathbf{y} \geq \mathbf{1}$.

A function $\mathbf{m}: P \to \mathbb{N}$ (usually represented in vector form) is called *marking*. A *Petri net system*, or *marked Petri net*, $\mathcal{S}$, is a Petri net $\mathcal{N}$ with an *initial marking* $\mathbf{m_0}$. A transition $t \in T$ is *enabled* at marking $\mathbf{m}$ if $\forall p \in P$: $\mathbf{m}[p] \geq \mathbf{Pre}[p,t]$. A transition t enabled at $\mathbf{m}$ can *fire* yielding a new marking $\mathbf{m}'$ defined by $\mathbf{m}'[p] = \mathbf{m}[p] - \mathbf{Pre}[p,t] + \mathbf{Post}[p,t]$ (denoted by $\mathbf{m}\xrightarrow{t}\mathbf{m}'$). A sequence of transitions $\sigma = t_1 t_2 \ldots t_n$ is a *firing sequence* in $\mathcal{S}$ if there exists a sequence of markings such that $\mathbf{m_0}\xrightarrow{t_1}\mathbf{m}_1\xrightarrow{t_2}\mathbf{m}_2 \ldots \xrightarrow{t_n}\mathbf{m}_n$. In this case, marking $\mathbf{m}_n$ is said to be *reachable* from $\mathbf{m_0}$ by firing σ(denoted by $\mathbf{m_0}\xrightarrow{\sigma}\mathbf{m}_n$). The *reachability set* $\mathrm{RS}(\mathcal{S})$ of a system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$ is the set of all markings reachable from the initial marking. $\mathrm{L}(\mathcal{S})$ is the *language of firing sequences* of a system $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$ ($\mathrm{L}(\mathcal{S}) = \{\sigma \mid \mathbf{m_0}\xrightarrow{\sigma}\mathbf{m}\}$).

A place $p \in P$ is said to be *k–bounded* in $\mathcal{S}$ if $\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S})$, $\mathbf{m}[p] \leq k$. A PN system is said to be k–bounded if every place is k–bounded, and bounded if there exists some k for which it is k–bounded. A net is *structurally bounded* if it is bounded for any $\mathbf{m_0}$. A PN system is *live* when every transition can ultimately occur from every reachable marking. A state $\mathbf{m}$ is called a *home state* if it is reachable from every reachable marking.

State Machines (SM's) are ordinary PN's such that every transition has only one input and only one output place ($\forall t \in T$: $|{}^\bullet t| = |t^\bullet| = 1$). SM's allow the modelling of sequences, decisions (or conflicts), and re-entrance (when they are marked with more than one token) but not synchronization. SM's marked with a single token model sequential processes. *Marked Graphs* (MG's) are ordinary PN's such that every place has exactly one input and one output transition ($\forall p \in P$: $|{}^\bullet p| = |p^\bullet| = 1$). MG's allow the modelling of sequences and synchronization but not decisions.

We indicate with $\otimes$ ($\oplus$) the tensor product (sum). The tensor product of a $n \times m$ matrix by a $p \times q$ produces a $n \cdot p \times m \cdot q$. The same is true for tensor sum, but the operator can be applied only to square matrices.

A *Stochastic Petri Net* (SPN) [29] is a pair $\langle \mathcal{S}, w \rangle = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0}, w \rangle$, where $\mathcal{S}$ is a PN system and $w : T \to \mathbb{R}^+$ is a positive real function that associates to each transition $t \in T$ an exponentially distributed firing time of rate $w(t)$. We shall indicate with $\mathbf{Q}$ the infinitesimal generator of the CTMC associated to an SPN, and with $\boldsymbol{\pi}$ the steady-state probability vector.

In this paper we assume that transitions are of the *single server* type. For the solution to be feasible the system must be bounded. In order to use extensively structural techniques, we shall assume in the sequel that the net is conservative (thus structurally bounded).

2.2 Single level structured approach: using a flat view

The basic idea behind the flat technique can be explained using the model of Figure 1, that shows a GSPN $\mathcal{S}$ that can be considered as the composition of two GSPNs $\mathcal{S}_1$ and $\mathcal{S}_2$ over three common transitions $T1, T2$ and $T3$. Places whose names start with letter a define component $\mathcal{S}_1$, and those starting with b define $\mathcal{S}_2$. We assume that there is a sequence of n places and transitions between $b21$ and $b2n$, and of m places and transitions between $a31$ and $a3m$. $\mathcal{S}_1$ has therefore $m+2$ states, while $\mathcal{S}_2$ has $n+2$. A product state space PS can then be defined as

$$\mathrm{PS} = \mathrm{RS}_1 \times \mathrm{RS}_2$$

and it is straightforward to observe that $\mathrm{RS} \subseteq \mathrm{PS}$: indeed PS has $(m+2)\cdot(n+2)$ states, but the reachability set of $\mathcal{S}$ has only $m+n+1$.

According to the techniques presented in [21] the following matrix $\mathbf{G}$ of size $|\mathrm{PS}| \times |\mathrm{PS}|$ can be constructed:

$$\mathbf{G} \;=\; \mathbf{Q}'_1 \oplus \mathbf{Q}'_2 \;-\; \sum_{t \in \{T1,T2,T3\}} w(t)[\mathbf{K}_1(t) \otimes \mathbf{K}_2(t)] \;+\; \sum_{t \in \{T1,T2,T3\}} w(t)[\mathbf{K}'_1(t) \otimes \mathbf{K}'_2(t)]$$

where $\mathbf{Q}'_i$, $\mathbf{K}_i(t)$, and $\mathbf{K}'_i(t)$ (for $i \in \{1,2\}$) are $|\mathrm{RS}_i| \times |\mathrm{RS}_i|$ matrices that can be derived from the infinitesimal generator $\mathbf{Q}_i$ of $\mathcal{S}_i$.

The idea behind this formula is to split the behaviour of each component into *local behaviour* (related to transitions local to a single component), and *dependent behaviour* (related to "synchronizing transitions" $T1, T2$, and $T3$). The local behaviour of each GSPN is represented by $\mathbf{Q}'_i$, and since the local behaviour is independent, the global behaviour due to local transitions can be obtained as the tensor sum of the $\mathbf{Q}'_i$ matrices. The behaviour related to synchronization requires that, for a synchronization

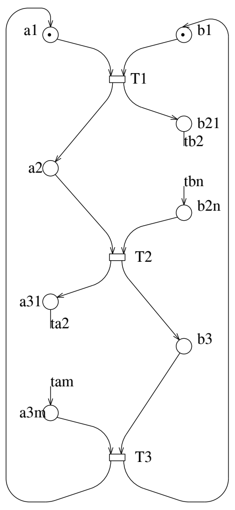

Figure 1: Explaining abstract views.

transition to fire, both $\mathcal{S}_i$ must be in a state that enables the transition. $\mathbf{K}_i(t)$, the correcting matrix for transition t, has a 1 in each entry of the matrix that corresponds to a change of state due to t in $\mathcal{S}_i$. The tensor product will indeed realize the required condition that *a synchronization transition fires only in global states whose corresponding local states in the* $\mathcal{S}_i$ *enable* t. The term with the $\mathbf{K}'_i(t)$ matrices is used to compute the portion of the diagonal elements expression that accounts for synchronization transitions.

By definition of the tensor sum and product, $\mathbf{G}$ is a $|\mathrm{PS}| \times |\mathrm{PS}|$ matrix, and it is shown in [21, 32] how the non null entries of the vector $\boldsymbol{\pi}$, solution of the equation $\boldsymbol{\pi} \cdot \mathbf{G} = \mathbf{0}$, are the steady-state solution of $\mathcal{S}$. Moreover, a solution process may be devised [18, 32] that does not require the explicit computation and storing of $\mathbf{G}$, so that the biggest memory requirement is that of the vector $\boldsymbol{\pi}$. The technique is extended to transient analysis in [28]. The computational cost, under full matrix implementation assumption, is smaller than the classical vector to matrix multiplication [32]. Recent results have shown [5] that under sparse matrix implementation the cost is instead bigger for matrices with a mean number of elements per row less than $K^{\frac{1}{K-1}}$ (K being the number of components).

The above technique produces a significant storage saving whenever the size of PS, and therefore that of $\boldsymbol{\pi}$, is inferior to the number of non null elements of $\mathbf{Q}$, since, otherwise, it would be better to store $\mathbf{Q}$ explicitly.

The solution procedure outlined above is the basic idea behind a number of works that have appeared in the literature. The work in [32] defines the basics of the method and applies it to networks of stochastic automata, while classes of SPN's for which a single level structured solution has been applied are Superposed Stochastic Automata [20] and Superposed GSPN (SGSPN) [21] (nets that can be interpreted as the superposition over a subset of timed transitions of a set of GSPN's).

The distance between RS and PS can limit the applicability of the technique. Nevertheless if a bit vector of size $|\mathrm{PS}|$ can be allocated in memory, then a state space exploration can be performed [26]. That exploration, with an additional tree-like data structure of size $O(|\mathrm{RS}| \cdot \log |\mathrm{RS}_i|)$ allows the definition of multiplication algorithms that consider only reachable states (i is the index of the component whose state space is represented in the tree leaves). The computational overhead is at most $O(\log |\mathrm{RS}|)$ [5].

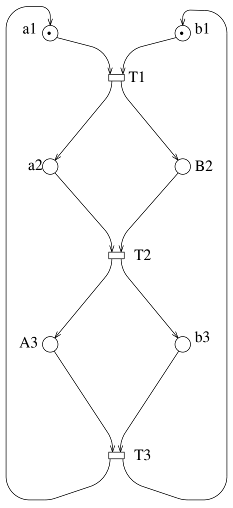

Figure 2: Explaining abstract views.

2.3 Two level structured approach: using a high level view

An abstract description of the system can be used to limit the number of spurious states by appropriately pre-selecting the subsets of the states that should be combined by the Cartesian product.

For example, we can consider the net $\mathcal{S}_a$ of Figure 2 as an abstract representation of the one in Figure 1, with place $B2$ "summarizing" the structure of places $b21, \ldots, b2n$ and $A3$ "summarizing" the structure of places $a31, \ldots, a3m$. $\mathcal{S}_a$ has three reachable states: $\mathbf{z}_1 = (\mathsf{a1}, \mathsf{b1})$, $\mathbf{z}_2 = (\mathsf{a2}, \mathsf{B2})$, and $\mathbf{z}_3 = (\mathsf{A3}, \mathsf{b3})$. The states of $\mathcal{S}_1$ and $\mathcal{S}_2$ can be partitioned according to the states of $\mathcal{S}_a$: the $m+2$ states of $\mathcal{S}_1$ are partitioned in three equivalence classes: $\mathrm{RS}_{\mathbf{z}_1}(\mathcal{S}_1) = \{\mathsf{a1}\}$, $\mathrm{RS}_{\mathbf{z}_2}(\mathcal{S}_1) = \{\mathsf{a2}\}$, and $\mathrm{RS}_{\mathbf{z}_3}(\mathcal{S}_1) = \{\mathsf{a31}, \ldots, \mathsf{a3m}\}$. Similarly, for $\mathcal{S}_2$ we get: $\mathrm{RS}_{\mathbf{z}_1}(\mathcal{S}_2) = \{\mathsf{b1}\}$, $\mathrm{RS}_{\mathbf{z}_2}(\mathcal{S}_2) = \{\mathsf{b21}, \ldots, \mathsf{b2n}\}$, and $\mathrm{RS}_{\mathbf{z}_3}(\mathcal{S}_2) = \{\mathsf{b3}\}$. The restricted product state space RPS can then be built as:

$$\begin{aligned} \mathrm{RPS}(\mathcal{S}) &= \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{S}_a)} \mathrm{RS}_{\mathbf{z}}(\mathcal{S}_1) \times \mathrm{RS}_{\mathbf{z}}(\mathcal{S}_2) = \\ \{\mathsf{a1}\} \times \{\mathsf{b1}\} &\cup \{\mathsf{a2}\} \times \{\mathsf{b21}, \ldots, \mathsf{b2n}\} \cup \{\mathsf{a31}, \ldots, \mathsf{a3m}\} \times \{\mathsf{b3}\} \end{aligned}$$

where $\biguplus$ is the *disjoint* set union. We refer to methods based on a construction of a restricted state space as *two levels.* Note that for this example we obtain a precise characterization of the state space, but the union of Cartesian products can in general produce a superset of the reachable state space, depending on the accuracy of the abstract representation.

Since the state space is no longer the Cartesian product of sets of local state spaces, but the union of Cartesian products, we cannot expect to have for the infinitesimal generator a tensor expression as simple as before: we can build a matrix $\mathbf{G}$, of size $|\mathrm{RPS}| \times |\mathrm{RPS}|$, and then consider it as block structured according to the states of $\mathcal{S}_a$. Each block refers to the set of states obtained by a single Cartesian product, and a tensor expression for each block can be derived. Diagonal blocks $\mathbf{G}(\mathbf{z}, \mathbf{z})$ can be expressed as[1]

$$\mathbf{G}(\mathbf{z}, \mathbf{z}) = \mathbf{Q}_1(\mathbf{z}, \mathbf{z}) \oplus \mathbf{Q}_2(\mathbf{z}, \mathbf{z})$$

[1]For simplicity, we are not considering diagonal elements, that can be computed on the fly, or stored explicitly, as explained in Section 5

where the $\mathbf{Q}_i(\mathbf{z},\mathbf{z})$ are the submatrices of the infinitesimal generator of $\mathcal{S}_i$ determined by the states whose abstract representation is $\mathbf{z}$ ($\mathbf{z} \in \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$). Each $\mathbf{G}(\mathbf{z},\mathbf{z}')$ block, with $\mathbf{z} \neq \mathbf{z}'$, ($\mathbf{z},\mathbf{z}' \in \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$), describes instead the behaviour that changes the high level state; each block $\mathbf{Q}(\mathbf{z},\mathbf{z}')$ can be written as

$$\mathbf{G}(\mathbf{z},\mathbf{z}') = \sum_{t:\mathbf{z} \xrightarrow{t} \mathbf{z}'} w(t)[\mathbf{K}'_1(t)(\mathbf{z},\mathbf{z}') \otimes \mathbf{K}'_2(t)(\mathbf{z},\mathbf{z}')]$$

where the $\mathbf{K}'_i(t)(\mathbf{z},\mathbf{z}')$ are the submatrices of the infinitesimal generator of $\mathcal{S}_i$ whose rows (columns) are abstractly represented by $\mathbf{z}$ ($\mathbf{z}'$), projected to include only the contribution due to t. Observe that, in our example, there is a single t for each given pair $(\mathbf{z},\mathbf{z}')$.

Although the approach can be applied to SGSPN, in the literature it was developed and it has been used only in the context of the solution of "asynchronous systems" either queueing networks or SPN, where subnets interact by exchanging customers or tokens. In [4] and [2] the two level method was developed to solve hierarchical queueing networks and hierarchical colored GSPN's: the simplest case of hierarchy consists of one model at the top level and K leaf models, that are activated by the top level through customers or token exchange. In [6] the method was adapted to marked graphs where components are defined through a cut (in graph sense) of the net over a set of places.

The construction of RPS for hierarchies uses the top level of the hierarchy as abstract representation (called High Level Model —HLM— in the queueing network hierarchy and High-level coloured GSPN —HCGSPN— in the coloured GSPN hierarchy). The component models are instead the leaves of the hierarchy (in the simplified case of a two level hierarchy).

For marked graphs, an abstract representation of the system is not given by construction, and subnets defined by a cut are open subnets, for which it is not possible to compute a reachability set to be used in the RPS formula. Following a decomposition defined in the context of approximation [8], the work in [6] defines an abstract representation called High level System ($\mathcal{HS}$) (it was called *Basic Skeleton*, $\mathcal{BS}$ in [8], and we shall follow the original terminology). Given $\mathcal{BS}$ and a cut of the MG into K subnets, it is possible to build a set of K *low level systems*, $\mathcal{LS}_i$: in each $\mathcal{LS}_i$ the full subnet i is completed by the abstract representation, taken from $\mathcal{BS}$, of the rest of the system. This allows the generation of the reachability set of the low level components and a restricted product space can be defined as:

$$\mathrm{RPS}(\mathcal{S}) \;=\; \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_1) \times \cdots \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_K)$$

where $\mathrm{RS}(\mathcal{X})$ indicates the state space of system $\mathcal{X}$, and $\mathrm{RS}_{\mathbf{z}}(\mathcal{X})$ is the set of markings of $\mathrm{RS}(\mathcal{X})$ with projection over the high level behaviour equal to $\mathbf{z}$. By construction of the abstract representation $\mathrm{RS}(\mathcal{S}) = \mathrm{RPS}(\mathcal{S})$. A similar formula can be devised for HCGSPN, but $\mathrm{RS}(\mathcal{S}) \subseteq \mathrm{RPS}(\mathcal{S})$.

The structured definition of RPS can be used, as in the example shown above, to express matrix $\mathbf{G}$, supermatrix of the infinitesimal generator.

In summary, if we have a construction rule for RS as a disjoint union of Cartesian products, we can limit the problem of the difference $|\mathrm{PS}| - |\mathrm{RS}|$, while still maintaining the benefits of the solution in structured form as in the SGSPN case: indeed it is only necessary to organize the classical vector by matrix multiplication $\boldsymbol{\pi} \cdot \mathbf{G}$ in terms of subvector by submatrix multiplication. Details for the computational costs of this technique under sparse and full matrix storage scheme have been reported in [3].

Tensor algebra techniques have been applied also for the solution of Stochastic Well-formed Nets [13]. The case of modules that interact through transition superposition is presented in [23], while the case of modules that interact through buffers is presented in [24]: in both cases the solution requires the construction of component models of the low level system type, but the solution presented in the papers is of the flat type.

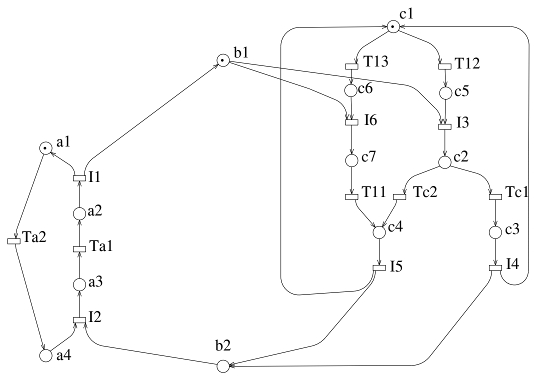

Figure 3: A DSSP system.

3 SAM view: low level systems and basic skeleton

This paper is based upon a previous work of the same authors [9] that considers *Deterministically Synchronized Sequential Processes* (DSSP) [34]. DSSP can be used for the modelling and analysis of distributed systems composed by sequential processes communicating through output-private buffers. Each sequential process is modelled by a *safe* (1–bounded) strongly connected State Machine (SM). The communication among them is described by *buffers* (places) which contain *products/messages* (tokens), that are produced by certain processes and consumed by others. Each buffer is *output-private*: it is an input place of only one SM. A well-developed theory exists for the analysis of qualitative behaviour of these systems [33, 34] that allows, in particular, to efficiently check necessary and sufficient conditions for *finiteness* and *ergodicity* of the embedded CTMC.

An example of DSSP with two buffers and two state machines is depicted in Figure 3. The first SM, $\mathcal{SM}_1$, is identified by places labeled with a while the second one, $\mathcal{SM}_2$, is identified by places labeled with c. Places $b1$ and $b2$ are the buffers. Transitions $I1$ and $I2$ are the *interface transitions* between $\mathcal{SM}_1$ and the buffers while $I3, I4, I5, I6$ are the interface between $\mathcal{SM}_2$ and the buffers. The rest of transitions are termed *internal* (since they model actions that change only the internal state of the corresponding SM).

The technique for DSSP is here extended to general P/T systems. In order to do that, a "DSSP-like" *structured view* of the model is considered in which functional units (now modules with general structure) communicate through buffers without any interconnection constraint. A net model with a given structured view is referred as *System of Asynchronously Communicating Modules* (SAM). Notice that SAM is not a PN subclass (as DSSP) but just provides a *structured view* of a general P/T model.

In this section we define SAM views and rules to build partial models, *components.* They are going to be used in the next sections for the definition of RS and infinitesimal generator expressions according to the two levels method. To achieve this goal, a *reduction* rule for the internal behaviour of modules of a SAM is defined (*internal behaviour* means firing of internal transitions). Using that reduction a collection of *low level systems* and one *basic skeleton* are built. In each low level system, only one of the modules of the original system is kept while the internal structure of the others is reduced. In the basic skeleton, the internal structure of all modules is reduced.

3.1 The SAM view of general P/T nets

Consider a general P/T system that (possibly by construction) admits a structured view as a set of K *modules* (disjoint P/T systems) that asynchronously communicate by means of a set of places called *buffers*. No constraint is assumed on the interconnection between buffers and modules, therefore the modelling of general situations of *competition* and/or *cooperation* among modules is possible.

Definition 1 *A PN system, $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$, is a* System of Asynchronously Communicating Modules *view, or simply a SAM, if:*

1. $P_i \cap P_j = \emptyset$, $\forall i, j \in \{1, \ldots, K\}, i \neq j$;
2. $T_i \cap T_j = \emptyset$, $\forall i, j \in \{1, \ldots, K\}, i \neq j$;
3. $P_i \cap B = \emptyset$, $\forall i \in \{1, \ldots, K\}$;
4. $\mathbf{Pre}[P_i, T_j] = \mathbf{Post}[P_i, T_j] = \mathbf{0}$, $\forall i, j \in \{1, \ldots, K\}, i \neq j$.

$\langle \mathcal{N}_i, \mathbf{m_0}_i \rangle = \langle P_i, T_i, \mathbf{Pre}_i, \mathbf{Post}_i, \mathbf{m_0}_i \rangle$, $i \in \{1, \ldots, K\}$, *are called* modules *of $\mathcal{S}$ (where $\mathbf{Pre}_i$,$\mathbf{Post}_i$, and $\mathbf{m_0}_i$ are the restrictions of $\mathbf{Pre}$, $\mathbf{Post}$, and $\mathbf{m_0}$ to P_i and T_i).*
Places in B are called buffers.
Transitions belonging to the set $\mathrm{TI} = {}^\bullet B \cup B^\bullet$ *are called* interface *transitions. Remaining ones* $((T_1 \cup \ldots \cup T_K) \setminus \mathrm{TI})$ *are called* internal *transitions.*

In the sequel, *strong connection* of the whole net is assumed since it is a necessary condition for live and bounded systems [36] (and we are interested in systems with such fundamental properties). We also assume that the net is *conservative* (observe that if a system is bounded, by adding complementary places it can always be transformed into a conservative net).

The reader should notice that in fact *all* PN systems can be provided with SAM views varying between the following two extreme positions:

- a single module and an empty set of buffers; or, in the other extreme,
- one module per transition and all places are buffers.

Therefore, the effect of the above definition is to assume that a *partitioned view* of the system into modules connected through buffers is given (or known a priori; for example, coming from the net construction process).

Let us consider the P/T system depicted in Figure 4. Apart from the two extreme SAM views mentioned above (a single module or as many modules as transitions), three alternative SAM views for the system could be easily considered. The first one considers the system partitioned into three modules (subnets generated by places labeled with a, c, and d, respectively) connected through four buffers ($b1, b2, b3, b4$). The second one considers as a single module the subnet generated by places $b1$, $b3$ and those labeled with a and c, while the subnet generated by places labeled with d is the second module and the buffers are $b2$ and $b4$. The last one considers as a module the subnet generated by places labeled with a; the second module is the subnet generated by places $b2$, $b4$, and those labeled with c and d. Places $b1$ and $b3$ are buffers. The existence of many SAM views of a net system opens the question of which one is the best (e.g., the more efficient) for the computation of the solution.

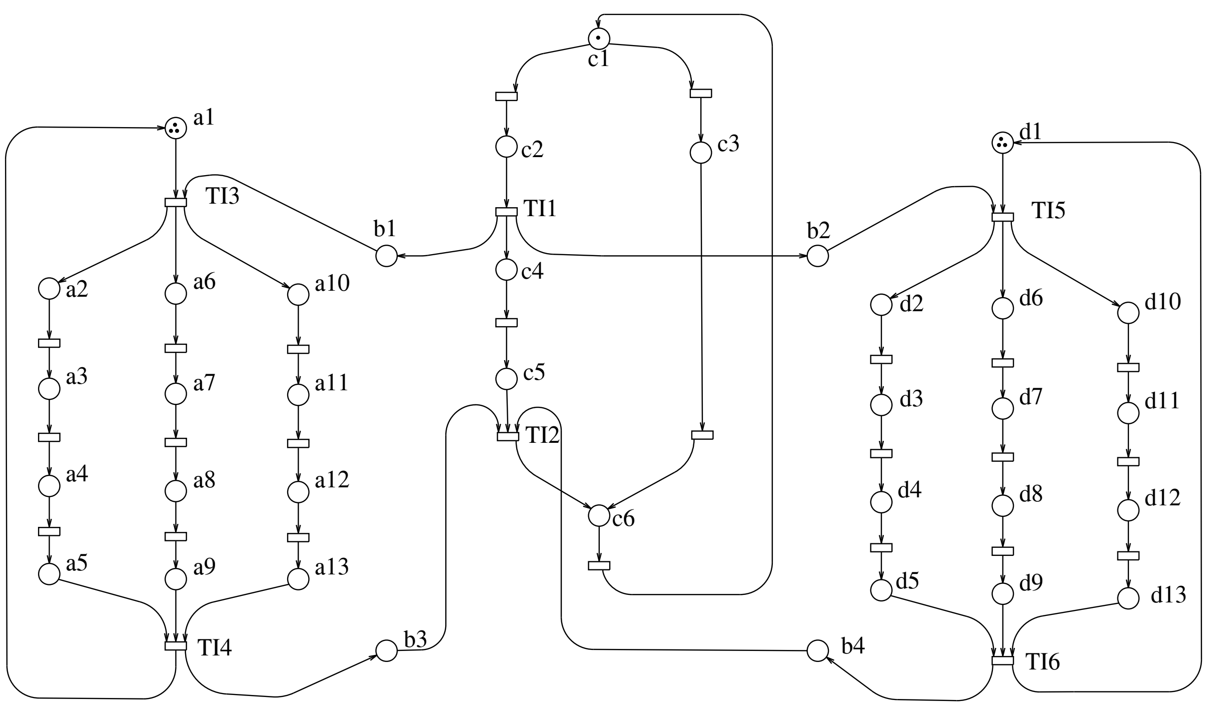

Figure 4: SAM view of an SPN.

3.2 Decomposition according to the SAM view: basic ideas and running example

In this section we informally present the main ideas behind a reduction rule for the internal behaviour of the modules of a SAM. Using the rule, we introduce a decomposition of SAM into a collection of *low level systems* and one *basic skeleton* (along the lines in [8]). In each low level system, *only one of the different modules of the original system is kept while the internal structure of the others is reduced as much as possible.* In the basic skeleton, *the internal structure of all the modules is reduced.* Low level systems and basic skeleton are used for a structured construction of the reachability set of the original model and also for a structured computation of its steady-state probabilities. In the next section, formal definitions and technical details of the reduction process are given.

Let us come back to the example in Figure 3 and consider for it a SAM view that distinguishes two modules, $\mathcal{N}_1$ and $\mathcal{N}_2$, (subnets generated by places labeled with a and c, respectively) and two buffers, $b1$ and $b2$. The first step is to derive in an efficient way an *extended system*, $\mathcal{ES}$, like that depicted in Figure 5. It consists of the original system plus the addition of some *implicit places* ($A14$, $A23$, $C34$, and $C56$) that summarize information of the structure of $\mathcal{N}_1$ and $\mathcal{N}_2$. An implicit place [17] is one whose removal does not affect the behaviour of the system (therefore, behaviour of the original and of the extended systems is the same assuming *interleaving semantics* [17], although the notion of implicit place can be directly extended to cope with a *step semantics* [15]). Since we are considering a Markovian interpretation of PN's and single-server semantics of transitions, the embedded CTMC of a system is preserved if implicit places are added or removed.

Implicit places are computed as follows. First, an *equivalence relation* R is defined over the set of places P_i of each module, partitioning P_i into equivalence classes P_i^j. Two places of a module are related by R if and only if there exists between them a non-directed path including only nodes of that module but interface transitions. In the example, P_1 is partitioned into two equivalence classes, $P_1^1 = \{a1, a4\}$

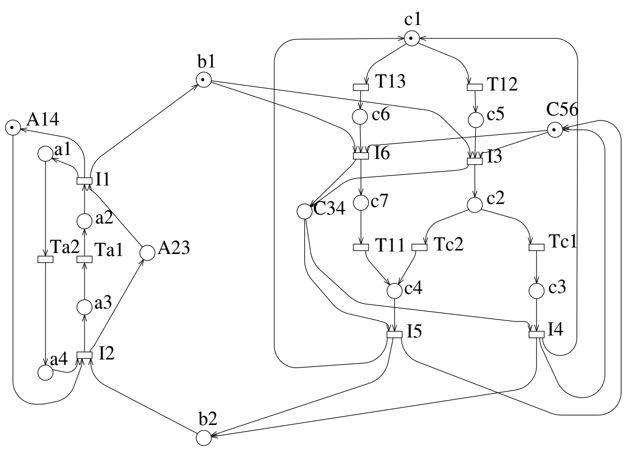

Figure 5: The SAM example model with additional implicit places.

and $P_1^2 = \{a2, a3\}$, while P_2 is partitioned into $P_2^1 = \{c2, c3, c4, c7\}$ and $P_2^2 = \{c1, c5, c6\}$. For each equivalence class P_i^j a set H_i^j (standing for "High-level places") of implicit places is computed, that includes information of the behaviour on $\mathcal{N}_i$. For instance, in the example, $H_2^1 = \{C34\}$ is the set (in this case only one place) of implicit places corresponding to the equivalence class $P_2^1 = \{c2, c3, c4, c7\}$.

Low level system, $\mathcal{LS}_i$ $(i = 1, \ldots, K)$, is derived by reducing all modules $\mathcal{N}_j$, $j \neq i$, to their interface transitions and to the set of implicit H_i^j, while $\mathcal{N}_i$ is fully preserved. The *basic skeleton*, $\mathcal{BS}$, is instead derived by reducing *all* modules to interface transitions and implicit places. The basic skeleton defines the most abstract view of the original system that we are going to consider.

Figure 6 shows the low level systems, $\mathcal{LS}_1$ and $\mathcal{LS}_2$, and the basic skeleton, $\mathcal{BS}$ for the running example. Notice that if $\mathcal{LS}_1$ and $\mathcal{LS}_2$ were synchronized by merging common transitions (interface transitions) and identifying common places (buffers and implicit places), the extended system (with equivalent behaviour to the original system) would be obtained. Notice also that the basic skeleton is the common abstraction between $\mathcal{LS}_1$ and $\mathcal{LS}_2$.

3.3 Decomposition according to the SAM view: technical aspects

Let us first recall the definitions of implicit place, marking structurally implicit place and implying places of it, and later a result that gives a value for its initial marking.

Definition 2 [17] *Let* $\mathcal{S} = \langle P \cup \{p\}, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$ *be a P/T system. Place* p *is* implicit *iff* $\mathrm{L}(\mathcal{S}) = \mathrm{L}(\langle P, T, \mathbf{Pre}[P,T], \mathbf{Post}[P,T], \mathbf{m_0}[P] \rangle)$ *(i.e., deletion of* p *preserves the language of firing sequences).*

In words, p is implicit if it is never the unique one to prevent the enabling of a transition.

Definition 3 [17] *Let* $\mathcal{N}$ *be a net and* p *be a place with incidence vector* $l_p = \mathbf{C}[p, \cdot]$*. The place* p *is a* marking structurally implicit place *(MSIP) in* $\mathcal{N}$ *if there exists* $\mathbf{y} \geq \mathbf{0}$ *such that* $\mathbf{y}[p] = 0$ *and* $l_p = \mathbf{y} \cdot \mathbf{C}$*. The set of places in* $\|\mathbf{y}\|$ *are called* implying places *of* p *(where* $\|\mathbf{y}\|$*, called support of* $\mathbf{y}$*, is the set of non-zero components of* $\mathbf{y}$*).*

The following property provides a sufficient condition for an MSIP to be implicit whose checking requires the solution of a linear programming problem (on real variables), giving rise to a time complexity that is polynomial on the net structure size.

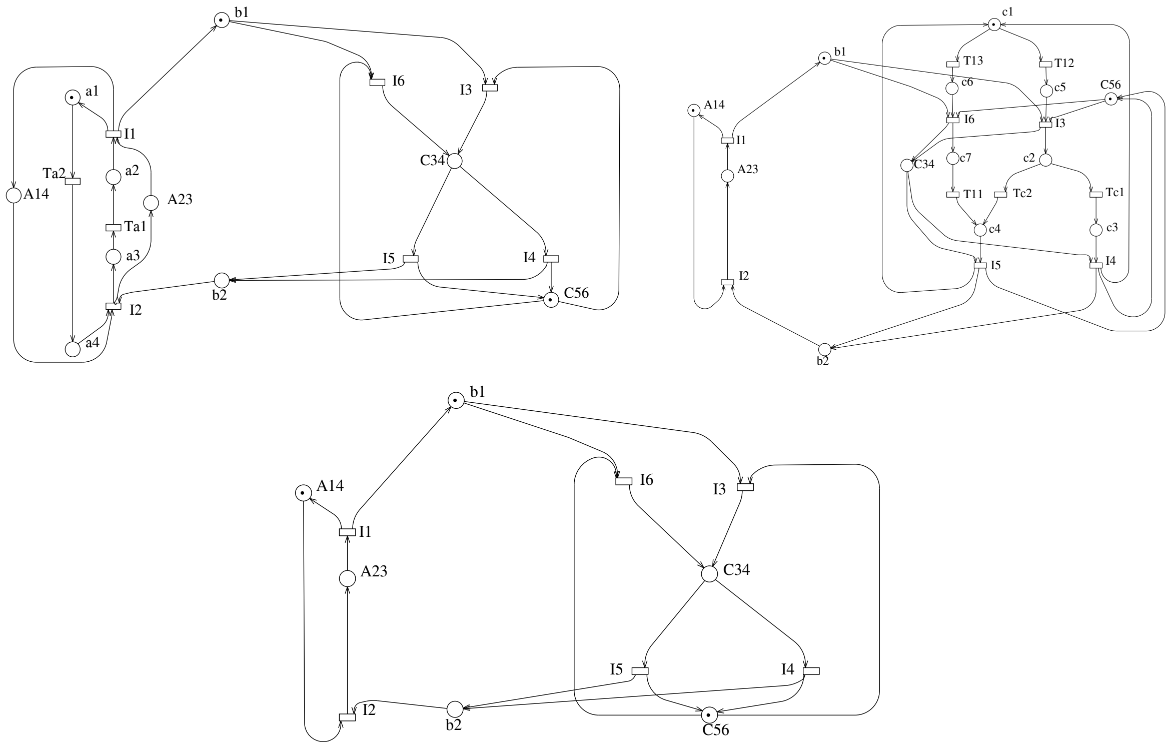

Figure 6: The $\mathcal{LS}_1$, $\mathcal{LS}_2$, and the $\mathcal{BS}$ systems corresponding to the SAM of Fig. 3.

Property 4 [17] *Let $\langle \mathcal{N}, \mathbf{m_0} \rangle$ be a conservative net system and p an MSIP of $\mathcal{N}$. If $\mathbf{m_0}[p]$ is greater than or equal to v then p is implicit, where v is the optimal value of the following linear programming problem:*

$$\begin{array}{rrl} v = & \textit{minimum} & \mathbf{y} \cdot \mathbf{m_0} + \mu \\ & \textit{subject to} & \mathbf{y} \cdot \mathbf{C} = \mathbf{C}[p, \cdot] \\ & & \mathbf{y} \cdot \mathbf{Pre}[\cdot, t] + \mu \geq \mathbf{Pre}[p, t], \forall t \in p^\bullet \\ & & \mathbf{y} \geq \mathbf{0}, \mathbf{y}[p] = 0 \end{array}$$

Now, we formally define the equivalence relation R that induces a partition of the places P_i of each module of a SAM into several equivalence classes P_i^j.

Definition 5 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$ be a SAM view of a P/T system. We denote by* R *the equivalence relation defined on all the places in $P \setminus B$ by: $\langle p, p' \rangle \in \mathrm{R}$ iff there exists an index i such that $p, p' \in P_i$ and there exists a non-directed path Π in $\mathcal{N}_i$ from p to p' such that $\Pi \cap \mathrm{TI} = \emptyset$ (i.e., containing only internal transitions). Let P_i^j, $j = 1, \ldots, r(i)$, be the different equivalence classes defined in P_i by the relation* R *and $T_i^j = {}^\bullet P_i^j \cup P_i^{j\bullet} \setminus \mathrm{TI}$, $j = 1, \ldots, r(i)$, $i = 1, \ldots, K$.*

We shall substitute each P_i^j by a, hopefully smaller, set of implicit places. To this goal we only consider MSIP since the computation of MSIP may be achieved *structurally.* For each equivalence class P_i^j, $j = 1, \ldots, r(i)$, we compute the sets H_i^j with a minimum number of MSIP's to be added to the module $\mathcal{N}_i$ that include the maximum information of the structure of $\mathcal{N}_i$ $(i = 1, \ldots, K)$. Moreover (and obviously) we consider a *finite* subset of the infinite number of possible MSIP's: those places derived from the *minimal* P-semiflows of the subnet generated by P_i^j, as presented in the following algorithm.

The basic idea is to apply a classical algorithm [16] for the computation of all the *minimal (support), positive, left annullers* of $\mathbf{C}[P_i^j, T_i^j]$, and to derive from each of these vectors $\mathbf{y}$ a MSIP $p_{\mathbf{y}}$ with incidence vector $l_{p_{\mathbf{y}}} = \mathbf{y} \cdot \mathbf{C}$, eliminating the repeated instances of an MSIP p that could be generated from different vectors $\mathbf{y}$.

Algorithm 6 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$ be a SAM view of a P/T system. Let* R, *P_i^j, and T_i^j ($j = 1, \ldots, r(i)$, $i = 1, \ldots, K$) be as introduced in Definition 5. The following algorithm defines the sets of MSIP's H_i^j ($j = 1, \ldots, r(i)$, $i = 1, \ldots, K$) and their initial markings.*

```
input   $\mathcal{N}_i$ and one equivalence class $P_i^j$ of those defined in $P_i$ by R.
output  Set of MSIP's $H_i^j$ and their initial markings.
begin
    $Y_i^j := \{\mathbf{y} \mid \mathbf{y}[P_i^j] \cdot \mathbf{C}[P_i^j, T_i^j] = \mathbf{0}, \mathbf{y}[P \setminus P_i^j] = \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{y}$ has minimal support$\}$;
    $H_i^j := \{p_{\mathbf{y}} \mid p_{\mathbf{y}}$ is a place with incidence vector $l_{p_{\mathbf{y}}} = \mathbf{y} \cdot \mathbf{C}\}$;
    for each $p \in H_i^j$ do
        compute its initial marking using Property 4
    end for
end
```

We denote by $H_i = \bigcup_{j=1}^{r(i)} H_i^j$ the set of all MSIP's corresponding to $\mathcal{N}_i$, $i = 1 \ldots K$, and $H = \bigcup_{i=1}^{K} H_i$.

The cardinality of the Y_i^j set may increase *exponentially* on the size of the subnet induced by P_i^j and T_i^j (indeed Y_i^j is the set of all minimal P-semiflows of a subnet), so that the theoretical time complexity of the first step in the algorithm is exponential in the size of the subnet. The same is true for the set H_i^j built in the second step of the algorithm (examples can be built where the cardinality of H_i^j increases exponentially in the size of the subnet). Anyhow, thanks to the *divide and conquer* approach, the exponential increase is in the size of a part (P_i^j and T_i^j) of a module (P_i and T_i), and not in the size of the whole original net.

Two particular cases with *polynomial* complexity, for which the set H_i^j contains only one place are the following: (1) the subnet P_i^j includes neither fork nor join transitions (a particular case is that of DSSP, considered in [9], where modules are state machines), or (2) P_i^j has a *single input* interface transition and a *single output* interface transition. In the first case, a weighted sum of places P_i^j leads to a unique minimal P-semiflow. In the second case, the place in H_i^j can be easily computed (without applying the algorithm) by imposing conservativeness of the subnet resulting after the addition of that place to the subnet defined by P_i^j and T_i^j: the number of minimal P-semiflows can be exponential on the subnet size but all of them would generate the same place.

One interesting question that could be posed is whether it is necessary to consider *all* minimal P-semiflows of the subnet to derive the set H_i^j or if a *base* of minimal P-semiflows would lead to a (smaller) set of MSIP's retaining the same behavioural information (this would made polynomial the time complexity of the Algorithm 6). The answer is no since to consider (only) *a base of minimal P-semiflows can lead to a reduced system that is structurally unbounded*, and an example is the MG in Figure 7. Consider the module composed by places ai, $i = 1, \ldots, 4$, and transition $T7$ with interface transitions $\{T1, T2, T3, T4\}$). The equivalence relation R induces in the module a single equivalence class. A base of minimal P-semiflows for the same module generates three places. A possible selection of these places is represented in Figure 8. The reader can easily check that this system allows the sequence $T1 - T3 - T5$ to be fired an arbitrary number of times before the firing of any other transition (making

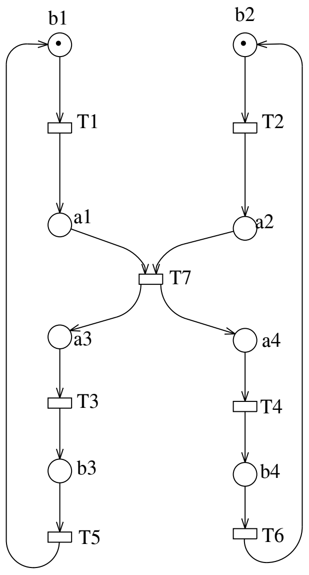

Figure 7: An MG system (bi, $i = 1, \ldots, 4$, are the buffers).

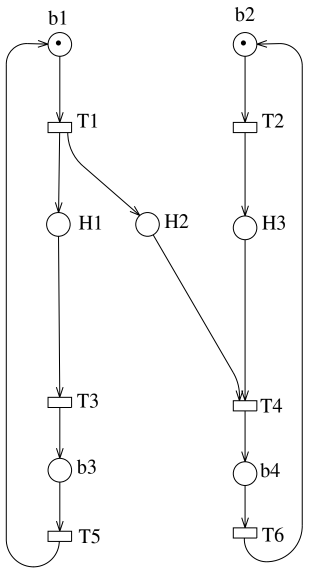

Figure 8: Reduction of module $a1 - a2 - T7 - a3 - a4$ in Figure 7 using only a base of minimal P-semiflows.

place $H2$ unbounded), while this was not the case for the original system. If all minimal P-semiflows were considered instead, an additional place $H4$ connecting $T2$ to $T3$ would be added to the system of Figure 8 and, in this case, the substitution of the subnet $a1 - a2 - T7 - a3 - a4$ by the set of four implicit places preserves exactly the functional observable behaviour of interface transitions.

An *extended system* $\mathcal{ES}$ is built from a system $\mathcal{S}$ by adding the sets of implicit places, H_i^j, computed by Algorithm 6. The reachability graphs of $\mathcal{S}$ and $\mathcal{ES}$ are identical (only the redundant marking of implicit places is added in the state representation of $\mathcal{ES}$).

Definition 7 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0}\rangle$ be a SAM. The extended system $\mathcal{ES}$ of $\mathcal{S}$ is obtained by adding to $\mathcal{S}$ all the places in H_i^j, $j = 1, \ldots, r(i)$, $i = 1, \ldots, K$, defined by Algorithm 6, with the initial marking computed by the same algorithm.*

As an example, the extended system of the SAM of Figure 3 is depicted in Figure 5, where place $C34$ summarizes places $c2, c3, c4$, and $c7$, place $C56$ summarizes places $c1$, $c5$, and $c6$, place $A14$ summarizes places $a1$ and $a4$, and place $A23$ summarizes places $a2$ and $a3$. We use upper cases for implicit places, with an index that is a composition of two of the indices of the places from which they are constructed. The H_i^j are: $H_1^1 = \{A14\}$, $H_1^2 = \{A23\}$, $H_2^1 = \{C34\}$, and $H_2^2 = \{C56\}$.

K different *low level systems* $\mathcal{LS}_i$ can be obtained from the extended system by deleting all places in P_j, $j \neq i$, and transitions in $T_j \setminus \mathrm{TI}$, $j \neq i$ $(i = 1, \ldots, K)$.

Definition 8 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0}\rangle$ be a SAM and $\mathcal{ES}$ its extended system.*

i) The low level system *$\mathcal{LS}_i$ $(i = 1, \ldots, K)$ of $\mathcal{S}$ is obtained from $\mathcal{ES}$ deleting all nodes in $\bigcup_{j\neq i}(P_j \cup (T_j \setminus \mathrm{TI}))$ and their adjacent arcs.*

ii) The basic skeleton *$\mathcal{BS}$ of $\mathcal{S}$ is obtained from $\mathcal{ES}$ deleting all nodes in $\bigcup_j (P_j \cup (T_j \setminus \mathrm{TI}))$ and their adjacent arcs.*

In each $\mathcal{LS}_i$ all modules $\mathcal{N}_j$, $j \neq i$, are reduced to interface transitions and to the implicit places that were added in the extended system, while $\mathcal{N}_i$ is fully preserved. In the basic skeleton all modules are reduced. Figure 6 shows the low level systems $\mathcal{LS}_1$, $\mathcal{LS}_2$, and the basic skeleton $\mathcal{BS}$ of the SAM in Figure 3.

The next property states that conservativeness of the original net is inherited by its low level system and the basic skeleton.

Property 9 *Let $\mathcal{S} = \langle \mathcal{N}, \mathbf{m_0}\rangle$ be a P/T system and $\mathcal{LS}_i$, $i = 1 \ldots, K$, $\mathcal{BS}$ its low level systems and basic skeleton corresponding to a SAM view. If $\mathcal{N}$ is conservative then the nets of $\mathcal{LS}_i$ $(i = 1 \ldots, K)$ and $\mathcal{BS}$ are conservative.*

Proof:
Instead of considering $\mathcal{LS}_i$, we consider the net $\mathcal{N}'$ resulting from $\mathcal{N}$ after the addition of a single set of implicit places H_k^j (for a given $k \neq i$ and a given $j \in \{1, \ldots, r(k)\}$) and the deletion of places P_k^j and transitions T_k^j (and their adjacent arcs). If we prove that $\mathcal{N}'$ is conservative, then conservativeness of the net of $\mathcal{LS}_i$ follows by repeating the same reduction over the rest of places and internal transitions of $\mathcal{N}_k$, with $k \neq i$.

Let us prove that if $\mathcal{N}$ is conservative then $\mathcal{N}'$ is also conservative. It is sufficient to prove that (1) for each P-semiflow of $\mathcal{N}$, $\mathbf{y} \geq \mathbf{0}$ such that $\mathbf{y} \cdot \mathbf{C} = \mathbf{0}$, there exists a P-semiflow of $\mathcal{N}'$ that has the same projection on the preserved places, and (2) places H_k^j in $\mathcal{N}'$ can be covered by a P-semiflow.

Proof of (1): If $\mathbf{y}[P_k^j] = \mathbf{0}$ we are done; if this is not the case, from $\mathbf{y} \cdot \mathbf{C} = \mathbf{0}$, by re-ordering rows and columns, we get $\mathbf{y}[P_k^j] \cdot \mathbf{C}[P_k^j, T_k^j] = \mathbf{0}$ and $\mathbf{y} \cdot \mathbf{C}[P, T \setminus T_k^j] = \mathbf{0}$. Now, consider place $p_{\mathbf{y}'}$ as defined in Algorithm 6 for $\mathbf{y}' = \left[\mathbf{y}[P_k^j] \mid \mathbf{0}\right]$. Then, it can be easily checked that vector $\left[1, 0, \ldots, 0 \mid \mathbf{y}[P \setminus P_k^j]\right]$, where entry '1' corresponds to the place $p_{\mathbf{y}'}$ and entries '0' correspond to the remaining places generated by Algorithm 6, is a P-semiflow of $\mathcal{N}'$ that has the same projection as $\mathbf{y}$ on the preserved places.
Proof of (2): Consider the net resulting from $\mathcal{N}$ by adding places H_k^j. It is conservative (because $\mathcal{N}$ is conservative and the added places are MSIP's), thus it exists a P-semiflow $\mathbf{y}$ that covers all places in H_k^j. If the support of $\mathbf{y}$ does not include any place from P_k^j, then we are done because $\mathbf{y}$ is also a P-semiflow of $\mathcal{N}'$ and it covers all places in H_k^j. If the support of $\mathbf{y}$ includes some places from P_k^j, then $\mathbf{y}[P_k^j]$ must be such that $\mathbf{y}[P_k^j] \cdot \mathbf{C}[P_k^j, T_k^j] = \mathbf{0}$, i.e., $\mathbf{y}[P_k^j]$ is a P-semiflow of the subnet generated by P_k^j and T_k^j. Then, by construction of the set H_k^j (it contains a place for each minimal P-semiflow of the subnet generated by P_k^j and T_k^j), a P-semiflow for $\mathcal{N}'$ can be derived from $\mathbf{y}$ by deleting the entries corresponding to places P_k^j and incrementing appropriately the entries corresponding to H_k^j.
The same argument is valid for $\mathcal{BS}$, by reducing all modules. ♢

In general, the reduction technique presented here does not remove but it can add new paths between interface transitions (see, for instance, the path from $I6$ to $I4$ that is present in $\mathcal{BS}$ (Figure 6), but that does not exist in the original model of Figure 3) as proved by the next property.

Property 10 *Let* $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0}\rangle$ *be a SAM,* $\mathcal{LS}_i$ *its low level systems (*$i = 1, \ldots, K$*),* $\mathcal{BS}$ *its basic skeleton, and* $\mathrm{L}(\mathcal{S})$ *the language of* $\mathcal{S}$*. Then:*

1. $\mathrm{L}(\mathcal{S})|_{T_i \cup \mathrm{TI}} \subseteq \mathrm{L}(\mathcal{LS}_i)$, *for* $i = 1, \ldots, K$.
2. $\mathrm{L}(\mathcal{S})|_{\mathrm{TI}} \subseteq \mathrm{L}(\mathcal{BS})$.

Proof:
Consider the extended system $\mathcal{ES}$ of $\mathcal{S}$. By definition, all places in H_i^j, $j = 1, \ldots, r(i)$, $i = 1, \ldots, K$, are implicit in $\mathcal{ES}$. Therefore $\mathrm{L}(\mathcal{S}) = \mathrm{L}(\mathcal{ES})$. Consider the system $\mathcal{LS}_i$ for an arbitrary i: obviously, $\mathrm{L}(\mathcal{ES})|_{T_i \cup \mathrm{TI}} \subseteq \mathrm{L}(\mathcal{LS}_i)$, and the property follows. The same argument is valid for the basic skeleton (statement *2*). ♢

As a final comment let us remark that the reduction technique proposed here, when applied to MG, produces a set of components that is the same as the one produced by the algorithm for MG approximation in [8], where Floyd's *all shortest path algorithm* is used to compute H_i^j. However, it is slightly different from the one presented in [6] where the net is split into subnets with a frontier made of transitions, and the $\mathcal{BS}$ model is obtained by substituting paths between pairs of frontier transitions with an implicit place that "summarizes" the behaviour of the path.

4 The structured construction of the reachability set

The decomposition defined in the previous section is now used to build the restricted product space $\mathrm{RPS}(\mathcal{S})$ of a SAM. As a first step the reachability sets of original ($\mathrm{RS}(\mathcal{S})$), extended ($\mathrm{RS}(\mathcal{ES})$), and low level systems ($\mathrm{RS}(\mathcal{LS}_i)$) are partitioned according to the projection of the marking on the places of the basic skeleton.

Definition 11 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$ be a SAM, $\mathcal{ES}$ its extended system, $\mathcal{LS}_i$ its low level systems ($i = 1, \ldots, K$), and $\mathcal{BS}$ its basic skeleton. Then, the following subsets of the reachability sets* $\mathrm{RS}(\mathcal{S})$, $\mathrm{RS}(\mathcal{ES})$, *and* $\mathrm{RS}(\mathcal{LS}_i)$ *are defined for each* $\mathbf{z} \in \mathrm{RS}(\mathcal{BS})$:

$$\begin{aligned}
&\mathrm{RS}_{\mathbf{z}}(\mathcal{ES}) = \{\mathbf{m} \in \mathrm{RS}(\mathcal{ES}) \mid \mathbf{m}|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}\} \\
&\mathrm{RS}_{\mathbf{z}}(\mathcal{S}) = \{\mathbf{m} \in \mathrm{RS}(\mathcal{S}) \mid \exists \mathbf{m}' \in \mathrm{RS}_{\mathbf{z}}(\mathcal{ES}) : \mathbf{m}'|_{P_1 \cup \ldots \cup P_K \cup B} = \mathbf{m}\} \\
&\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i) = \{\mathbf{m}_i \in \mathrm{RS}(\mathcal{LS}_i) \mid \mathbf{m}_i|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}\}
\end{aligned}$$

From the definition, it follows that: $\mathrm{RS}(\mathcal{S}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RS}_{\mathbf{z}}(\mathcal{S})$, $\mathrm{RS}(\mathcal{ES}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RS}_{\mathbf{z}}(\mathcal{ES})$, and $\mathrm{RS}(\mathcal{LS}_i) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)$ ($i = 1, \ldots, K$), where symbol $\uplus$ denotes the disjoint union of sets. Remember that, by Proposition 9, reachability sets of the low level systems and of the basic skeleton are finite.

The following result essentially states that each reachable marking of a SAM can be expressed as a composition of conveniently selected markings of the low level systems.

Theorem 12 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$ be a SAM, $\mathcal{LS}_i$ ($i = 1, \ldots, K$) its low level systems, and $\mathcal{BS}$ its basic skeleton. For each $\mathbf{z} \in \mathrm{RS}(\mathcal{BS})$, let*

$$\mathrm{RPS}_{\mathbf{z}}(\mathcal{S}) = \{\mathbf{z}|_B\} \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_1)|_{P_1} \times \cdots \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_K)|_{P_K}$$

and

$$\mathrm{RPS}(\mathcal{S}) = \bigcup_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RPS}_{\mathbf{z}}(\mathcal{S})$$

then:

$$\mathrm{RS}(\mathcal{S}) \subseteq \mathrm{RPS}(\mathcal{S}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \mathrm{RPS}_{\mathbf{z}}(\mathcal{S}) \tag{1}$$

Moreover:

$$\mathrm{RS}_{\mathbf{z}}(\mathcal{S}) \subseteq \mathrm{RPS}_{\mathbf{z}}(\mathcal{S})$$

Proof:
First, we prove the inclusion.

Let $\mathbf{m} \in \mathrm{RS}_{\mathbf{z}}(\mathcal{S})$, then there exists a sequence σ such that $\mathbf{m_0} \xrightarrow{\sigma} \mathbf{m}$. By Property 10, there exist sequences σ_i with $\sigma_i = \sigma|_{T_i \cup \mathrm{TI}}$ ($i = 1, \ldots, K$) and $\sigma_{\mathcal{BS}}$ with $\sigma_{\mathcal{BS}} = \sigma|_{\mathrm{TI}}$ firable in $\mathcal{LS}_i$ ($i = 1, \ldots, K$) and in $\mathcal{BS}$, respectively: $\mathbf{m_0^{\mathit{i}}} \xrightarrow{\sigma_i} \mathbf{m}_i$ ($i = 1, \ldots, K$) and $\mathbf{z_0} \xrightarrow{\sigma_{\mathcal{BS}}} \mathbf{z}$, where $\mathbf{m_0^{\mathit{i}}}$ ($i = 1, \ldots, K$) and $\mathbf{z_0}$ denote the initial markings of $\mathcal{LS}_i$ ($i = 1, \ldots, K$) and $\mathcal{BS}$, respectively.

Then: $\mathbf{m}_i|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}$, thus $\mathbf{m}_i \in \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)$, $i = 1, \ldots, K$ (since $\mathbf{m_0^{\mathit{i}}}|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z_0}$, $\sigma_i|_{\mathrm{TI}} = \sigma_{\mathcal{BS}}$, and $H_1 \cup \ldots \cup H_K \cup B \subset {}^\bullet\mathrm{TI} \cup \mathrm{TI}^\bullet$).

From analogous arguments, $\mathbf{z}|_B = \mathbf{m}|_B$ (because $\mathbf{z_0}|_B = \mathbf{m_0}|_B$, $\sigma|_{\mathrm{TI}} = \sigma_{\mathcal{BS}}$, and $B \subset {}^\bullet\mathrm{TI} \cup \mathrm{TI}^\bullet$).

Then, $\mathbf{m} = (\mathbf{z}|_B, \mathbf{m}_1|_{P_1}, \ldots, \mathbf{m}_K|_{P_K})$, because $\mathbf{m_0}|_{P_i} = \mathbf{m_0^{\mathit{i}}}|_{P_i}$, $\sigma|_{T_i} = \sigma_i|_{T_i}$, and $P_i \subset {}^\bullet T_i \cup T_i^\bullet$, $i = 1, \ldots, K$.

Now, we prove by contradiction that the union is disjoint.

Assume that there exist two different states $\mathbf{z}, \mathbf{z}' \in \mathrm{RS}(\mathcal{BS})$ such that: $(\mathbf{z}|_B, \mathbf{m}_1|_{P_1}, \ldots, \mathbf{m}_K|_{P_K}) = (\mathbf{z}'|_B, \mathbf{m}'_1|_{P_1}, \ldots, \mathbf{m}'_K|_{P_K})$, with $\mathbf{m}_i \in \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)$, $\mathbf{m}'_i \in \mathrm{RS}_{\mathbf{z}'}(\mathcal{LS}_i)$ ($i = 1, \ldots, K$).

Then: $\mathbf{z}|_B = \mathbf{z}'|_B$ (obvious), $\mathbf{m}_i|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}$ (by definition of $\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)$) and $\mathbf{m}'_i|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}'$ (by definition of $\mathrm{RS}_{\mathbf{z}'}(\mathcal{LS}_i)$).

Since $\mathbf{m}_i|_{P_i} = \mathbf{m}'_i|_{P_i}$ (obvious) and places H_i are implicit in $\mathcal{LS}_i$, $\mathbf{m}_i|_{H_i} = \mathbf{m}'_i|_{H_i}$. Therefore, $\mathbf{z}|_{H_i} = \mathbf{z}'|_{H_i}$, $i = 1, \ldots, K$.

RS of $\mathcal{S}$		RS of $\mathcal{S}$	
$\mathbf{v}_1$	a1, b1, c1	$\mathbf{v}_{14}$	a1, c1, b2
$\mathbf{v}_2$	a1, b1, c6	$\mathbf{v}_{15}$	a4, c3
$\mathbf{v}_3$	a1, b1, c5	$\mathbf{v}_{16}$	a4, c1, b2
$\mathbf{v}_4$	a4, b1, c1	$\mathbf{v}_{17}$	a1, b2, c6
$\mathbf{v}_5$	a1, c7	$\mathbf{v}_{18}$	a1, b2, c5
$\mathbf{v}_6$	a4, b1, c6	$\mathbf{v}_{19}$	a4, b2, c6
$\mathbf{v}_7$	a4, b1, c5	$\mathbf{v}_{20}$	a4, b2, c5
$\mathbf{v}_8$	a1, c2	$\mathbf{v}_{21}$	a3, c1
$\mathbf{v}_9$	a1, c4	$\mathbf{v}_{22}$	a3, c6
$\mathbf{v}_{10}$	a4, c7	$\mathbf{v}_{23}$	a3, c5
$\mathbf{v}_{11}$	a4, c2	$\mathbf{v}_{24}$	a2, c1
$\mathbf{v}_{12}$	a1, c3	$\mathbf{v}_{25}$	a2, c6
$\mathbf{v}_{13}$	a4, c4	$\mathbf{v}_{26}$	a2, c5

Table 1: RS of the SAM of Figure 3.

Then, $\mathbf{z} = \mathbf{z}'$ and the result follows. $\diamondsuit$

Observe that, by definition, the RPS of a SAM with K modules is the Cartesian product of $K+1$ terms, since the buffers contribution is an isolated term $(\mathbf{z}|_B)$. Actually this term can be removed from the formula by simply taking the buffer contribution out of any $\mathcal{LS}_i$ system, by writing $\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)|_{P_i \cup B}$. Concerning the example, Table 1 lists the reachability set of the SAM of Figure 3, that consists of 26 states, $\mathbf{v}_i$. Table 2 lists instead the reachability sets of $\mathcal{BS}$ $(\mathbf{z}_i)$, $\mathcal{LS}_1$ $(\mathbf{x}_j)$, and $\mathcal{LS}_2$ $(\mathbf{y}_k)$, respectively. For each state of $\mathcal{LS}_1$ ($\mathcal{LS}_2$) we have indicated the partition $\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_1)$ ($\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_2)$) to which the state belongs (third column). The elements of the partition are identified by the corresponding high level marking in $\mathcal{BS}$. As proved above

$$\mathrm{RS}(\mathcal{S}) \subseteq \mathrm{RPS}(\mathcal{S}) = \biguplus_{\mathbf{z} \in \mathrm{RS}(\mathcal{BS})} \{\mathbf{z}|_B\} \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_1)|_{P_1} \times \mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_2)|_{P_2}$$

and in this case we actually have an equality. As an example consider the case of $\mathbf{z}_1$: then the cross product of $\mathrm{RS}_{\mathbf{z}_1}(\mathcal{LS}_1) = \{\mathbf{x}_1, \mathbf{x}_2\}$ (markings of $\mathcal{LS}_1$) and $\mathrm{RS}_{\mathbf{z}_1}(\mathcal{LS}_2) = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\}$ (markings of $\mathcal{LS}_2$) produces the states of RPS: $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_6$, and $\mathbf{v}_7$.

It is important to remember that in general $\mathrm{RPS}(\mathcal{S}) \neq \mathrm{RS}(\mathcal{S})$. The live and bounded PN system of Figure 9 is a case in which the inclusion is strict. Consider its SAM view where bi $(i = 1, \ldots, 4)$ are buffers, interface transitions are labeled with TI, and there are three modules: the first one contains only $TI5$, the second is the subnet generated by places starting with letter a, and the third is generated by places labeled with c. Places $z1, z2$ are the implicit places added to define the extended system. Reachable states corresponding to the high level state [z1, z2] are: [z1, z2, a1, a3, c1, c3], [z1, z2, a1, a3, c2, c4], [z1, z2, a2, a4, c1, c3], [z1, z2, a2, a4, c2, c4], [z1, z2, a1, a4, c1, c4], [z1, z2, a1, a4, c2, c3], [z1, z2, a2, a3, c1, c4], and [z1, z2, a2, a3, c2, c3]. But the cross product of [z1, z2, a1, a3] (reachable in $\mathcal{LS}_1$) and [z1, z2, c1, c4] (reachable in $\mathcal{LS}_2$) generates [z1, z2, a1, a3, c1, c4], that belongs to RPS but is non-reachable.

RS of $\mathcal{BS}$	
$\mathbf{z}_1$	A14, C56, b1
$\mathbf{z}_2$	A14, C34
$\mathbf{z}_3$	A14, C56, b2
$\mathbf{z}_4$	A23, C56

RS of $\mathcal{LS}_1$		
$\mathbf{x}_1$	a1, b1, C56, A14	$\mathbf{z}_1$
$\mathbf{x}_2$	a4, b1, C56, A14	$\mathbf{z}_1$
$\mathbf{x}_3$	a1, C34, A14	$\mathbf{z}_2$
$\mathbf{x}_4$	a4, C34, A14	$\mathbf{z}_2$
$\mathbf{x}_5$	a1, b2, C56, A14	$\mathbf{z}_3$
$\mathbf{x}_6$	a4, b2, C56, A14	$\mathbf{z}_3$
$\mathbf{x}_7$	a3, C56, A23	$\mathbf{z}_4$
$\mathbf{x}_8$	a2, C56, A23	$\mathbf{z}_4$

RS of $\mathcal{LS}_2$		
$\mathbf{y}_1$	A14, b1, c1, C56	$\mathbf{z}_1$
$\mathbf{y}_2$	A14, b1, c6, C56	$\mathbf{z}_1$
$\mathbf{y}_3$	A14, b1, c5, C56	$\mathbf{z}_1$
$\mathbf{y}_4$	A14, c7, C34	$\mathbf{z}_2$
$\mathbf{y}_5$	A14, c2, C34	$\mathbf{z}_2$
$\mathbf{y}_6$	A14, c4, C34	$\mathbf{z}_2$
$\mathbf{y}_7$	A14, c3, C34	$\mathbf{z}_2$
$\mathbf{y}_8$	A14, b2, c1, C56	$\mathbf{z}_3$
$\mathbf{y}_9$	A14, b2, c6, C56	$\mathbf{z}_3$
$\mathbf{y}_{10}$	A14, b2, c5, C56	$\mathbf{z}_3$
$\mathbf{y}_{11}$	A23, c1, C56	$\mathbf{z}_4$
$\mathbf{y}_{12}$	A23, c6, C56	$\mathbf{z}_4$
$\mathbf{y}_{13}$	A23, c5, C56	$\mathbf{z}_4$

Table 2: RS's of the SAM of Figures 6.

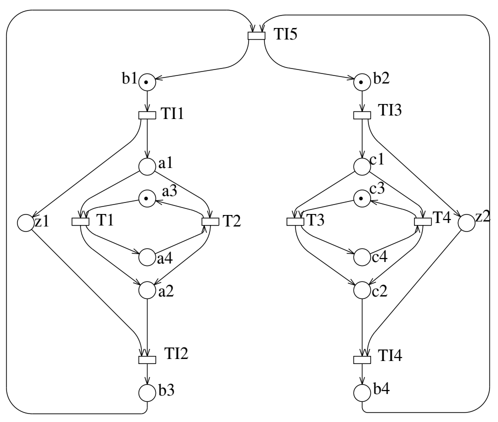

Figure 9: A SAM with RPS $\neq$ RS.

5 The structured solution of SAM

5.1 Construction of the infinitesimal generator

Given an infinitesimal generator, a rate matrix can be obtained by simply disregarding diagonal elements. The use of the rate matrix $\mathbf{R}$ instead of the infinitesimal generator $\mathbf{Q}$ allows for a simpler tensorial expression, as pointed out in numerous papers [26, 2], at the cost of either computing the diagonal elements on the fly, or of explicitly storing the diagonal. In this section we define the rate matrix of a stochastic SAM in terms of matrices derived from the rate matrix of the $\mathcal{LS}_i$ systems.

Let $\mathbf{Q}$ be the infinitesimal generator of a stochastic SAM. We can rewrite $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{R} - \mathbf{\Delta} \tag{2}$$

where $\mathbf{\Delta}$ is a diagonal matrix and $\mathbf{\Delta}[i,i] = \sum_{k \neq i} \mathbf{Q}[i,k]$. The same definition holds for the $\mathcal{LS}_i$ components:

$$\mathbf{Q}_i = \mathbf{R}_i - \mathbf{\Delta}_i$$

If states are ordered according to the high level state $\mathbf{z}$, then matrices $\mathbf{Q}$ and $\mathbf{R}$ (respectively, $\mathbf{Q}_i$ and $\mathbf{R}_i$) can be described in terms of blocks $(\mathbf{z},\mathbf{z}')$, of size $|\mathrm{RS}_{\mathbf{z}}(\mathcal{S})| \cdot |\mathrm{RS}_{\mathbf{z}'}(\mathcal{S})|$ (respectively, $|\mathrm{RS}_{\mathbf{z}}(\mathcal{LS}_i)| \cdot |\mathrm{RS}_{\mathbf{z}'}(\mathcal{LS}_i)|$). We shall indicate them with $\mathbf{Q}(\mathbf{z},\mathbf{z}')$ and $\mathbf{R}(\mathbf{z},\mathbf{z}')$ ($\mathbf{Q}_i(\mathbf{z},\mathbf{z}')$ and $\mathbf{R}_i(\mathbf{z},\mathbf{z}')$, respectively).

Diagonal blocks $\mathbf{R}_i(\mathbf{z},\mathbf{z})$ have non null entries that are due *only* to the firing of transitions in $T_i \setminus \mathrm{TI}$ (internal behaviour), while blocks $\mathbf{R}_i(\mathbf{z},\mathbf{z}')$ with $\mathbf{z} \neq \mathbf{z}'$ have non null entries due *only* to the firing of transitions in TI.

Let $\mathrm{TI}_{\mathbf{z},\mathbf{z}'}$ with $\mathbf{z} \neq \mathbf{z}'$, be the set of transitions $t \in \mathrm{TI}$ such that $\mathbf{z} \xrightarrow{t} \mathbf{z}'$ in the basic skeleton $\mathcal{BS}$. From a matrix $\mathbf{R}_i(\mathbf{z},\mathbf{z}')$, with $\mathbf{z} \neq \mathbf{z}'$ we can build additional matrices $\mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')$, for each $t \in \mathrm{TI}_{\mathbf{z},\mathbf{z}'}$, according to the following definition:

$$\mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')[\mathbf{m},\mathbf{m}'] = \begin{cases} 1 & \text{if } \mathbf{m} \xrightarrow{t} \mathbf{m}' \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{m}$ and $\mathbf{m}'$ are two of the states with a high level view equal to $\mathbf{z}$ and $\mathbf{z}'$ respectively: $\mathbf{m}|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}$, and $\mathbf{m}'|_{H_1 \cup \ldots \cup H_K \cup B} = \mathbf{z}'$.

Matrices $\mathbf{G}(\mathbf{z},\mathbf{z}')$ of size $|\mathrm{RPS}_{\mathbf{z}}(\mathcal{S})| \cdot |\mathrm{RPS}_{\mathbf{z}'}(\mathcal{S})|$ can then be defined as:

$$\begin{aligned} \mathbf{G}(\mathbf{z},\mathbf{z}) &= \textstyle\bigoplus_{i=1}^{K} \mathbf{R}_i(\mathbf{z},\mathbf{z}) \\ \mathbf{G}(\mathbf{z},\mathbf{z}') &= \textstyle\sum_{t \in \mathrm{TI}_{\mathbf{z},\mathbf{z}'}} w(t) \bigotimes_{i=1}^{K} \mathbf{K}_i(t)(\mathbf{z},\mathbf{z}') \end{aligned} \tag{3}$$

The following theorem states that a stochastic SAM can be solved using the $\mathbf{G}$ matrix defined by the $\mathbf{G}(\mathbf{z},\mathbf{z})$ and $\mathbf{G}(\mathbf{z},\mathbf{z}')$ blocks of equation (3).

Theorem 13 *Let $\mathcal{S} = \langle P_1 \cup \ldots \cup P_K \cup B, T_1 \cup \ldots \cup T_K, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0}, w \rangle$ be an SPN with a SAM view, $\mathbf{Q}$ its infinitesimal generator, $\mathcal{LS}_i$ its low level systems ($i = 1, \ldots, K$), and $\mathcal{BS}$ its basic skeleton. Let $\mathbf{R}$ be the matrix defined by equation (2), and $\mathbf{G}$ the one defined by equations (3). Then:*

1. *$\forall \mathbf{z}$ and $\mathbf{z}' \in \mathrm{RS}(\mathcal{BS})$: $\mathbf{R}(\mathbf{z},\mathbf{z}')$ is a submatrix of $\mathbf{G}(\mathbf{z},\mathbf{z}')$.*

2. *$\forall \mathbf{m} \in \mathrm{RS}(\mathcal{S})$ and $\forall \mathbf{m}' \in \mathrm{RPS}(\mathcal{S}) \setminus \mathrm{RS}(\mathcal{S})$: $\mathbf{G}[\mathbf{m},\mathbf{m}'] = 0$.*

Proof:
Since $\mathcal{S}$ and $\mathcal{ES}$ have the same behaviour and the same set of states, for notational convenience we use $\mathcal{ES}$ instead of $\mathcal{S}$. If $\mathbf{m}$ is a state of $\mathrm{RS}(\mathcal{ES})$ then, by Theorem 12, $\mathbf{m}$ can be rewritten in terms of the buffers state and of the $\mathcal{LS}_i$ states $\mathbf{m}_i$ as

$$\mathbf{m} = \mathbf{z}|_B, \mathbf{m}_1|_{P_1 \cup H_1}, \ldots, \mathbf{m}_K|_{P_K \cup H_K}$$

For the rest of the proof we shall rewrite a generic marking $\mathbf{m}$ as

$$\mathbf{m} = \mathbf{l}_1, \ldots, \mathbf{l}_K, \mathbf{b}, \mathbf{H}$$

where $\mathbf{l}_i = \mathbf{m}|_{P_i}$, $\mathbf{b} = \mathbf{z}|_B$, and $\mathbf{H} = \mathbf{z}|_{H_1 \cup \ldots \cup H_K}$.

We first prove that:

$$\forall\ \mathbf{m}, \mathbf{m}' \in \mathrm{RS}(\mathcal{ES}): \quad \mathbf{R}[\mathbf{m}, \mathbf{m}'] = \mu \implies \mathbf{G}[\mathbf{m}, \mathbf{m}'] = \mu$$

(since from $\mathbf{m}$ and $\mathbf{m}'$ the high level states $\mathbf{z}$ and $\mathbf{z}'$ are uniquely determined, we usually omit the specification of $\mathbf{z}$ and $\mathbf{z}'$).
Case $\mathbf{z} = \mathbf{z}'$: if the high level view is the same, then the change of state from $\mathbf{m}$ to $\mathbf{m}'$ can only be due to internal transitions (i.e., not belonging to TI), thus belonging to a single module $\mathcal{N}_i$. But, by definition of $\bigoplus$, $\mathbf{G}(\mathbf{z}, \mathbf{z})$ expresses the independent composition of the stochastic processes represented by the $\mathbf{R}_i(\mathbf{z}, \mathbf{z})$, which is exactly the behaviour of $\mathcal{LS}_i$ system due to transitions local to $\mathcal{N}_i$ (transitions belonging to $T_i \setminus \mathrm{TI}$).
Case $\mathbf{z} \neq \mathbf{z}'$: if the high level view is different, the change of state from $\mathbf{m}$ to $\mathbf{m}'$ can only be due to a transition in TI. Let t be such transition and assume, for the time being, that there is only one, so that $\mu = w(t)$. If t is a transition in the interface, then its enabling depends on the marking of the buffer places, and on the places of a *single* module.

If t is enabled in $\mathbf{m}$, then t is enabled in *each* $(\mathbf{l}_i, \mathbf{b}, \mathbf{H})$ state of the K $\mathcal{LS}_i$ systems, and in each $\mathcal{LS}_i$ system produces a change of state $\mathbf{l}_i, \mathbf{b}, \mathbf{H} \xrightarrow{t} \mathbf{l}'_i, \mathbf{b}', \mathbf{H}$. By definition of $\mathbf{K}_i(t)(\mathbf{z}, \mathbf{z}')$, we have that $\forall i, \mathbf{K}_i(t)(\mathbf{z}, \mathbf{z}')[\mathbf{m}_i, \mathbf{b}, \mathbf{H}] = 1$, and by definition of $\bigotimes$ there is a 1 in the corresponding entry of $\bigotimes_{i=1}^{K} \mathbf{K}_i(t)(\mathbf{z}, \mathbf{z}')$, and therefore a value of $w(t)$ in the $\mathbf{G}(\mathbf{z}, \mathbf{z}')$ matrix.

In the expression of $\mathbf{G}(\mathbf{z}, \mathbf{z}')$ the case in which more than one transition realizes the same change of state is accounted for by the summation over all transitions in $\mathrm{TI}_{\mathbf{z},\mathbf{z}'}$.

We now prove the second part of the theorem, that can be rewritten in terms of $\mathcal{ES}$ as:

$$\forall \mathbf{m} \in \mathrm{RS}(\mathcal{ES}), \forall \mathbf{m}' \in \mathrm{RPS}(\mathcal{S}) \setminus \mathrm{RS}(\mathcal{ES}) : \mathbf{G}[\mathbf{m}, \mathbf{m}'] = 0$$

As in the previous case, from $\mathbf{m}$ and $\mathbf{m}'$ the high level states $\mathbf{z}$ and $\mathbf{z}'$ are uniquely determined, and we consider for $\mathbf{m}$ and $\mathbf{m}'$ the same decomposition in terms of modules, buffers, and implicit places as before.

The proof is by contradiction assuming that $\mathbf{G}[\mathbf{m}, \mathbf{m}'] \neq 0$. It can be rewritten as: $\mathbf{G}[(\mathbf{l}_1, \ldots, \mathbf{l}_K, \mathbf{b}, \mathbf{H}), (\mathbf{l}'_1, \ldots, \mathbf{l}'_K, \mathbf{b}', \mathbf{H}')] \neq 0$.
Case $\mathbf{z} = \mathbf{z}'$: by definition of $\bigoplus$, $\mathbf{G}[\mathbf{m}, \mathbf{m}'] \neq 0$ implies that there exists exactly one index i such that $\mathbf{R}_i(\mathbf{z}, \mathbf{z})[(\mathbf{l}_i, \mathbf{b}, \mathbf{H}), (\mathbf{l}'_i, \mathbf{b}, \mathbf{H})] \neq 0$, and this change of state can be due only to the $\mathbf{l}_i$ portion of the state (since the high level view $\mathbf{z}$ does not change). Therefore, there must be a transition $t \in T_i \setminus \mathrm{TI}$ that is enabled in state $(\mathbf{l}_i, \mathbf{b}, \mathbf{H})$ of $\mathcal{LS}_i$, but then t is enabled also in state $\mathbf{m} = (\mathbf{l}_1, \ldots, \mathbf{l}_K, \mathbf{b}, \mathbf{H})$, and its firing produces $\mathbf{m}' = (\mathbf{l}_1, \ldots, \mathbf{l}'_i, \ldots, \mathbf{l}_K, \mathbf{b}, \mathbf{H})$, thus making $\mathbf{m}'$ is reachable in $\mathcal{ES}$, which contradicts the hypothesis.

Case $\mathbf{z} \neq \mathbf{z}'$: by definition of $\bigotimes$, $\mathbf{G}[\mathbf{m},\mathbf{m}'] \neq 0$ implies that there exists a $t \in \mathrm{TI}$ such that, for all indices i, $\mathbf{K}_i(t)[(\mathbf{l}_i,\mathbf{b},\mathbf{H}),(\mathbf{l}'_i,\mathbf{b}',\mathbf{H}')] = 1$, therefore t is enabled in state $\mathbf{m}$ of $\mathcal{ES}$, and its firing produces state $(\mathbf{l}'_1,\ldots,\mathbf{l}'_K,\mathbf{b}',\mathbf{H}')$, thus making $\mathbf{m}'$ reachable in $\mathcal{ES}$, and therefore in $\mathcal{S}$, which contradicts the hypothesis. $\diamondsuit$

As a consequence of the theorem the steady state distribution of a stochastic SAM can be computed using the $\mathbf{G}$ matrix given in equation (3). Indeed, as for *Superposed GSPN* [21], if we apply an iterative solution method for $\boldsymbol{\pi} \cdot \mathbf{G} = \mathbf{0}$, and if the initial probability vector assigns a non-null probability only to reachable states (for example by assigning a value of 1 to the initial marking), then by the second item of the above theorem a non-null probability is never assigned to a non reachable state.

Coming back to the example in Figure 5, we can order states in $\mathrm{RS}(\mathcal{S})$ according to their projection over $\mathcal{BS}$, so that $\mathbf{R}$ can be written in block structured form as:

$$\mathbf{R} = \left(\begin{array}{c|c|c|c}
\mathbf{R}(\mathbf{z}_1,\mathbf{z}_1) & \mathbf{R}(\mathbf{z}_1,\mathbf{z}_2) & \mathbf{R}(\mathbf{z}_1,\mathbf{z}_3) & \mathbf{R}(\mathbf{z}_1,\mathbf{z}_4) \\ \hline
\mathbf{R}(\mathbf{z}_2,\mathbf{z}_1) & \mathbf{R}(\mathbf{z}_2,\mathbf{z}_2) & \mathbf{R}(\mathbf{z}_2,\mathbf{z}_3) & \mathbf{R}(\mathbf{z}_2,\mathbf{z}_4) \\ \hline
\mathbf{R}(\mathbf{z}_3,\mathbf{z}_1) & \mathbf{R}(\mathbf{z}_3,\mathbf{z}_2) & \mathbf{R}(\mathbf{z}_3,\mathbf{z}_3) & \mathbf{R}(\mathbf{z}_3,\mathbf{z}_4) \\ \hline
\mathbf{R}(\mathbf{z}_4,\mathbf{z}_1) & \mathbf{R}(\mathbf{z}_4,\mathbf{z}_2) & \mathbf{R}(\mathbf{z}_4,\mathbf{z}_3) & \mathbf{R}(\mathbf{z}_4,\mathbf{z}_4)
\end{array}\right)$$

By definition of SAM only interface transitions contribute to $\mathbf{R}(\mathbf{z}_i,\mathbf{z}_j)$, when $i \neq j$, while only internal transitions contribute to $\mathbf{R}(\mathbf{z}_i,\mathbf{z}_i)$. Let us consider in more detail blocks $\mathbf{R}(\mathbf{z}_1,\mathbf{z}_1)$ and $\mathbf{R}(\mathbf{z}_1,\mathbf{z}_2)$: we explicitly write in the matrix the state identifier for rows and columns as pairs of states of $\mathcal{LS}_1$ and $\mathcal{LS}_2$. For example the second row corresponds to state $\mathbf{v}_2$ of $\mathrm{RS}(\mathcal{S})$, which is obtained as composition of state $\mathbf{x}_1$ of $\mathcal{LS}_1$ and state $\mathbf{y}_2$ of $\mathcal{LS}_2$.

$\mathbf{R}(\mathbf{z}_1,\mathbf{z}_1) =$

$$\begin{array}{r|cccccc}
 & \mathbf{x}_1,\mathbf{y}_1 & \mathbf{x}_1,\mathbf{y}_2 & \mathbf{x}_1,\mathbf{y}_3 & \mathbf{x}_2,\mathbf{y}_1 & \mathbf{x}_2,\mathbf{y}_2 & \mathbf{x}_2,\mathbf{y}_3 \\ \hline
\mathbf{x}_1,\mathbf{y}_1 & & w(T_{13}) & w(T_{12}) & w(Ta_2) & & \\
\mathbf{x}_1,\mathbf{y}_2 & & & & & w(Ta_2) & \\
\mathbf{x}_1,\mathbf{y}_3 & & & & & & w(Ta_2) \\
\mathbf{x}_2,\mathbf{y}_1 & & & & & w(T_{13}) & w(T_{12}) \\
\mathbf{x}_2,\mathbf{y}_2 & & & & & & \\
\mathbf{x}_2,\mathbf{y}_3 & & & & & &
\end{array}$$

$\mathbf{R}(\mathbf{z}_1,\mathbf{z}_2) =$

$$\begin{array}{r|cccccccc}
 & \mathbf{x}_3,\mathbf{y}_4 & \mathbf{x}_3,\mathbf{y}_5 & \mathbf{x}_3,\mathbf{y}_6 & \mathbf{x}_3,\mathbf{y}_7 & \mathbf{x}_4,\mathbf{y}_4 & \mathbf{x}_4,\mathbf{y}_5 & \mathbf{x}_4,\mathbf{y}_6 & \mathbf{x}_4,\mathbf{y}_7 \\ \hline
\mathbf{x}_1,\mathbf{y}_1 & & & & & & & & \\
\mathbf{x}_1,\mathbf{y}_2 & w(I_6) & & & & & & & \\
\mathbf{x}_1,\mathbf{y}_3 & & w(I_3) & & & & & & \\
\mathbf{x}_2,\mathbf{y}_1 & & & & & & & & \\
\mathbf{x}_2,\mathbf{y}_2 & & & & & w(I_6) & & & \\
\mathbf{x}_2,\mathbf{y}_3 & & & & & & w(I_3) & &
\end{array}$$

The only non null element of $\mathbf{R}_1(\mathbf{z}_1,\mathbf{z}_1)$ is $\mathbf{R}_1(\mathbf{z}_1,\mathbf{z}_1)[\mathbf{x}_1,\mathbf{x}_2] = w(Ta_2)$, while the only non null elements of $\mathbf{R}_2(\mathbf{z}_1,\mathbf{z}_1)$ are: $\mathbf{R}_2(\mathbf{z}_1,\mathbf{z}_1)[\mathbf{y}_1,\mathbf{y}_2] = w(T_{13})$ and $\mathbf{R}_2(\mathbf{z}_1,\mathbf{z}_1)[\mathbf{y}_1,\mathbf{y}_3] = w(T_{12})$.

The change of state from $\mathbf{z}_1$ to $\mathbf{z}_2$ can be due only to transition I_3 and I_6, we therefore build matrices $\mathbf{K}_1(I_3)(\mathbf{z}_1,\mathbf{z}_2)$, $\mathbf{K}_2(I_3)(\mathbf{z}_1,\mathbf{z}_2)$, $\mathbf{K}_1(I_6)(\mathbf{z}_1,\mathbf{z}_2)$, and $\mathbf{K}_2(I_6)(\mathbf{z}_1,\mathbf{z}_2)$. $\mathbf{K}_1(I_3)(\mathbf{z}_1,\mathbf{z}_2)$ and $\mathbf{K}_1(I_6)(\mathbf{z}_1,\mathbf{z}_2)$ are identity matrices. $\mathbf{K}_2(I_3)(\mathbf{z}_1,\mathbf{z}_2)[\mathbf{y}_3,\mathbf{y}_5] = 1$ and all other elements of $\mathbf{K}_2(I_3)(\mathbf{z}_1,\mathbf{z}_2)$ are null. $\mathbf{K}_2(I_6)(\mathbf{z}_1,\mathbf{z}_2)[\mathbf{y}_2,\mathbf{y}_4] = 1$ and all other elements of $\mathbf{K}_2(I_6)(\mathbf{z}_1,\mathbf{z}_2)$ are null.

According to equations (3) we get:

$$\begin{aligned}
&\mathbf{G}(\mathbf{z}_1,\mathbf{z}_1) = \mathbf{R}_1(\mathbf{z}_1,\mathbf{z}_1) \oplus \mathbf{R}_2(\mathbf{z}_1,\mathbf{z}_1)\\
&\mathbf{G}(\mathbf{z}_1,\mathbf{z}_2) = w(I_3)(\mathbf{K}_1(I_3)(\mathbf{z}_1,\mathbf{z}_2) \otimes \mathbf{K}_2(I_3)(\mathbf{z}_1,\mathbf{z}_2))+\\
&\qquad w(I_6)(\mathbf{K}_1(I_6)(\mathbf{z}_1,\mathbf{z}_2) \otimes \mathbf{K}_2(I_6)(\mathbf{z}_1,\mathbf{z}_2))
\end{aligned}$$

Since RPS($\mathcal{S}$) = RS($\mathcal{S}$), then $\mathbf{G}(\mathbf{z}_1,\mathbf{z}_1) = \mathbf{R}(\mathbf{z}_1,\mathbf{z}_1)$ and $\mathbf{G}(\mathbf{z}_1,\mathbf{z}_2) = \mathbf{R}(\mathbf{z}_1,\mathbf{z}_2)$.

5.2 Ergodicity

When the rate matrix expression is used to compute the steady-state probability distribution of markings, the associated CTMC should be *marking ergodic* in order for the computation to make sense. In the case of bounded PN systems (boundedness ensures finiteness of the CTMC), ergodicity of the marking process is equivalent to the existence of a unique *ergodic class* in the associated CTMC; in other words, the RG of the PN system must have a *home state*: that is to say only one *sink* (non-transient) *strongly connected component* with more than one state. Therefore, ergodicity testing can be achieved by implementing a computation of strongly connected components of the RG and a test for the existence of a single sink with several nodes among these components.

In [26], a depth-first search exploration of the RG has been proposed that uses the structured representation of a supermatrix of the infinitesimal generator of SGSPN's to avoid storing the graph matrix of the whole RG. In a similar way, *Tarjan's algorithm* [22] for the computation of the strongly connected components of a directed graph (based also in a depth-first traversal of the graph) can be implemented making use of the Kronecker expression of the blocks of the (supermatrix of the) rate matrix defined by equation (3). The computational cost of this algorithm is of the order of the iterations needed for the iterative solution of $\boldsymbol{\pi} \cdot \mathbf{G} = \mathbf{0}$, therefore is not significant with respect to the complete solution cost.

The particular case of DSSP is specially interesting since ergodicity can be checked in a very efficient (alternative) way using only the incidence matrix of the net and the initial marking: this is a check that can, and should, be done before starting the whole solution process. The testing procedure presented in detail in [9], consists of checking first structural boundedness, then checking a characterization for structural liveness and structural boundedness (rank theorem [33]), and finally checking deadlock-freeness. If all these conditions are satisfied, then the system is bounded, live, and it has a home state, thus, the associated CTMC is ergodic.

5.3 Hints about complexity

What is the complexity of the proposed approach with respect to the explicit generation and storage of the infinitesimal generator? There are clear limit cases: for example if all transitions are interface transitions (the system is tightly coupled), then $\mathcal{S} = \mathcal{BS} = \mathcal{LS}_i$, and it makes no sense to apply this method.

The computational cost to solve a SAM is the sum of the cost to build the RG, the cost to build the expression of the infinitesimal generator of the associated CTMC, and the cost of solving the characteristic equation $\boldsymbol{\pi} \cdot \mathbf{Q} = \mathbf{0}$. The proposed method has instead a cost that is due to: the construction of the $K+1$ components, the construction of the RG_i of each component, the construction of the $\mathbf{R}_i(\mathbf{z},\mathbf{z}')$ and $\mathbf{K}_i(t)(\mathbf{z},\mathbf{z}')$ matrices (that may include a re-ordering of the states in the reachability sets), the solution of the characteristic equation $\boldsymbol{\pi} \cdot \mathbf{G} = \mathbf{0}$, when $\mathbf{G}$ is expressed as in equation (3). It is clear that the advantages/disadvantages of the method depend on the relative size of the reachability graphs of $\mathcal{S}$, $\mathcal{BS}$, and $\mathcal{LS}_i$.

The storage cost of the classical solution method is due to the storage of vector $\boldsymbol{\pi}$ of size $|\mathrm{RS}(\mathcal{S})|$, and of matrix $\mathbf{Q}$. Usually $\mathbf{Q}$ is stored in sparse form, so that, disregarding the diagonal, its occupation is of the same order as the number of arcs in $\mathrm{RG}(\mathcal{S})$. The storage cost of the proposed approach is instead that of a vector $\boldsymbol{\pi}$ of size $|\mathrm{RPS}(\mathcal{S})|$, and of a number of matrices, all stored in sparse form: the total number of non-null elements, disregarding the diagonal, is of the same order as the sum of the number of arcs in the K reachability graphs $\mathrm{RG}_i(\mathcal{LS}_i)$. If a bit vector of size $|\mathrm{RPS}(\mathcal{S})|$ can be stored, it is actually possible to use a vector $\boldsymbol{\pi}$ of size $|\mathrm{RS}(\mathcal{S})|$ [26], as explained in Section 2.2.

In summary, the difference between the number of arcs in $\mathrm{RG}(\mathcal{S})$ and the sum of the number of arcs in the K $\mathrm{RG}_i(\mathcal{LS}_i)$ is what makes the method applicable in cases in which a direct solution is not possible, due to the lack of memory to store $\mathbf{Q}$.

6 Examples

We present three examples of the application of the technique for SAM. For each example we show the sizes of the reachability graphs of the basic skeleton and of the low level systems, and the size of RPS and RS. These results have been validated against the results of GreatSPN [10], when feasible, and with the results of SupGSPN, a tool for the analysis of SGSPN system ideated and implemented by Kemper [27]. We should point out that SupGSPN was ideated for the solution, using a flat approach, of SGSPN, but it can indeed be applied to any type of SPN for which a partition of the set of places is given, if it is possible to generate a finite state space for the components identified by the partition. Of course we should not expect that SupGSPN performs at his best here: it was used only to show that it is worth to have a two level technique for SAM. We have run SupGSPN with the option that uses the computation of P-invariants to bound the places, so as to be able to produce a finite state space also for components that will not produce a finite state space when considered in isolation.

As shall be shown by the examples, the SAM approach performs better when the $\mathcal{BS}$ is really an abstraction of the real system (that is to say when a non trivial number of states are mapped into the same macro state), and it may be convenient to have more small components than few big ones. Quite the opposite, as it may be expected, is true for the tool based on SGSPN, since a large number of modules tends to increase the difference between RS and PS.

For the technique presented in this paper we have implemented only the generation of the state space, according to the formula in equation (1). In our prototype implementation the subsystems are generated manually (which is actually straightforward with a graphical tool like GreatSPN). Then, the solution program builds the tangible reachability graphs of the $\mathcal{BS}$ and $\mathcal{LS}_i$ systems using GreatSPN (modified so that the buffer places and implicit places are first in the state definition and are in the same order in the different low level systems as in the basic skeleton). The states of each component are sorted in lexicographical order using the sort utility of Unix, and the global state space is built at the cost of an ordered merge of the state spaces.

In all tables reported the size of RS is computed either directly, using GreatSPN, or using the bit vector technique of SupGSPN.

	$\mathbf{m}(\mathsf{c1})$=1	$\mathbf{m}(\mathsf{c1}) = 2$	$\mathbf{m}(\mathsf{c1}) = 3$
$\mathcal{BS}$	10	46	146
$\mathcal{LS}_1$	199	6.985	108.945
$\mathcal{LS}_2$	199	6.985	108.945
$\mathcal{LS}_3$	22	190	1.032
RPS	8.716	3.872.341	431.270.717
RS	8.716	3.872.341	no memory
PS(case A)	11.426.400	no memory	no memory
PS(case B)	—	198.994.880	no memory

Table 3: SAM technique: State spaces computed for the model of Figure 4.

The first example is shown in Figure 4, and we consider a SAM view composed of three modules, interconnected through 4 buffers. Places b1 to b4 are buffers, while tags starting with a, c, d identify the places of the first, second and third module respectively. All modules have a single input interface transition and a single output interface transition, thus it is straightforward to compute implicit places, since there is a single implicit place per module, as explained in Section 3.3. Interface transitions are easily identified, since their names start with TI; only the labels of interface transitions are shown in the figure.

From this SAM we have built three low level systems $\mathcal{LS}_1$, $\mathcal{LS}_2$, $\mathcal{LS}_3$, and one basic skeleton $\mathcal{BS}$. Table 3 shows the sizes of the state space of these subsystems, as well as the size of RPS and RS of the complete system, for an initial marking with three tokens in a1 and d1, and corresponding implicit places, and of $1, 2$, and 3 tokens in place c1.

A straightforward comparison with SupGSPN may not be very significant, since the division of the SAM into modules may not be the best one for SGSPN. We have tried two different decompositions with SupGSPN: three components identified by places starting with a or b for the first component, c for the second, and d for the third (case **A**), and two components identified by places starting with a or b or d for the first component, and c for the second (case **B**).

The size of the product state space is also shown in Table 3, where — means that the experiment was not performed. For the **A** case it was not possible to increase to 2 the number of tokens in c1, since after the generation of the state spaces of the three components, of size 1.701, 5.161, and 5.161, the tool stops, which is not surprising considering that, according to the size of the components, the cardinality of PS is about $45 \ * 10^9$.

The decomposition of case **B** is indeed more favourable, since we were able to solve also the $\mathbf{m}(\mathsf{c1}) = 2$ model.

The second example is shown in Figure 10, and we consider a SAM view composed of three modules, interconnected through 5 buffers. Places b1 to b5 are buffers, while places whose tag starts with a, c, d identify the first, second and third modules respectively. Places that start with IP are the implicit places computed by Algorithm 6 (Section 3.3). Due to the strong interconnections between the components and the buffer places, and to the high number of interface transitions (12), the algorithm produces 14 implicit places, moreover 8 of these places are exact replica of places of the net, therefore we should not expect to have very "abstract" macro states, and, consequently, the advantage of the structured approach in this case is expected to be rather limited.

For this SAM four components are built: $\mathcal{LS}_1$ (places with tag starting with a), $\mathcal{LS}_2$ (tags starting

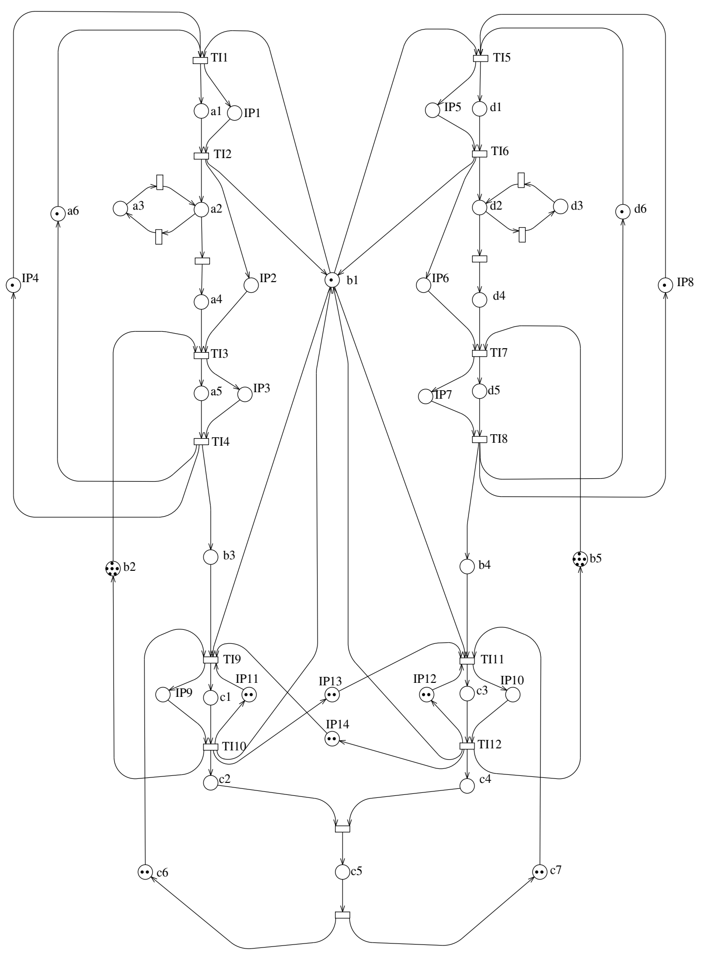

Figure 10: The second example.

with c), $\mathcal{LS}_3$ (tags starting with d), and one basic skeleton $\mathcal{BS}$. Table 4 shows the sizes of the state space of these subsystems, as well as the size of RPS and RS of the complete system, for eight different configurations of the initial marking. The first four columns consider initial markings with a single token in the first and third component, $\mathbf{m}(\mathsf{b1}) = 1$, $\mathbf{m}(\mathsf{b2}) = \mathbf{m}(\mathsf{b5}) = 2$, and a varying number of tokens in the second component, as shown in the table; the fifth column has a single token in the first and third component, six tokens in the second component, and $\mathbf{m}(\mathsf{b1}) = 1$, $\mathbf{m}(\mathsf{b2}) = \mathbf{m}(\mathsf{b5}) = 3$; the sixth column only differs from the previous one for having $\mathbf{m}(\mathsf{b1}) = 2$; the seventh column represents a configuration in which the first and third components have two tokens each, the second has six, and $\mathbf{m}(\mathsf{b1}) = 2$, $\mathbf{m}(\mathsf{b2}) = \mathbf{m}(\mathsf{b5}) = 3$; the last column is for the case of one token in the first and third component, two in the second, and $\mathbf{m}(\mathsf{b2}) = \mathbf{m}(\mathsf{b5}) = 6$.

Using SupGSPN with only two components (we have put buffer b1, the first, and third component together), for the same initial marking as in the fourth column of Table 4, we find a size for the two components of 72 and 36.855 and PS = 2.653.560, while the size of RS is 67.424. We also run the case reported in the last column of Table 4, getting 72 states for the first component, 84.035 for the second, and a PS of 6.050.520, against an RS of 38.624. It is interesting to note that, by adding to the net the two implicit places IP3 and IP7, and considering them as part of the second component (thus

	$\mathbf{m}$(c)=1	$\mathbf{m}$(c)=2	$\mathbf{m}$(c)=3	$\mathbf{m}$(c)=6	$\mathbf{m}$(b2) = 3 $\mathbf{m}$(b5)= 3	$\mathbf{m}$(b1) = 2	$\mathbf{m}$(a) = 2 $\mathbf{m}$(d)= 2	$\mathbf{m}$(b2) = 6 $\mathbf{m}$(b5)= 6
$\mathcal{BS}$	496	880	1.264	2.416	4.829	8.511	34.505	6.120
$\mathcal{LS}_1$	818	1.458	2.098	4.018	7.949	13.659	84.621	9.922
$\mathcal{LS}_2$	720	2.208	4.960	24.640	48.972	80.946	335.032	14.960
$\mathcal{LS}_3$	818	1.458	2.098	4.018	7.949	13.659	84.621	9.922
RPS	1.904	5.936	13.440	67.424	131.404	205.862	1.972.243	38.624
RS	1.904	5.936	13.440	67.424	131.404	205.862	1.972.243	38.624

Table 4: SAM technique: State spaces computed for the model of Figure 10.

	$\mathbf{m}$(c1)=1	$\mathbf{m}$(c1) = 2	$\mathbf{m}$(c1) = 3	$\mathbf{m}$(c1) = 4	$\mathbf{m}$(c1) = 5
$\mathcal{BS}$	46	371	1.596	4.917	12.298
$\mathcal{LS}_1$	166	1.531	6.956	22.085	56.306
$\mathcal{LS}_2$	94	1.345	9.720	47.790	182.412
$\mathcal{LS}_3$	323	2.423	10.018	30.106	74.029
$\mathcal{LS}_4$	127	1.352	6.546	21.531	56.137
$\mathcal{LS}_5$	166	1.531	6.956	22.085	56.306
RPS	25.926	536.277	4.557.376	24.613.188	99.960.756
RS	25.926	536.277	4.557.376	24.613.188	no memory

Table 5: SAM technique: State spaces computed for the model of Figure 11.

making the second component a strongly connected graph), we have got only 5.259 states in the second component, for a PS of 378.648; moreover the time of the computation decreased of almost one order of magnitude.

The third example is depicted in Figure 11, were we have 5 components, again identified by the first letter of the tag of the places. The results are presented in Table 5. The "no memory" of the last column is due to GreatSPN running out of memory for this initial marking. The computation of the reachable states took some a few minutes of user time for the four tokens case, and half an hour for five tokens. Considering the size of $\mathcal{LS}_2$ in the last column, we did not experiment with a larger number of tokens.

On this example we have also tried a different decomposition, by considering the subnets a, c, and f as a single module, with b2 and b5 as buffers. For this decomposition we only need two implicit places, one from TI5 to TI9, and the other from TI1 to TI2. The size of the $\mathcal{BS}$, $\mathcal{LS}_1$ and $\mathcal{LS}_2$ state spaces are shown in Table 6. Observe that, due to the large size of $\mathcal{LS}_1$, we could not solve the system for $\mathbf{m}$(c1) = 5.

We have also run SupGSPN on this example, with the same two decompositions as before. The results are reported in Tables 7 (five components case), and 8 (two components case).

The meaning of "no memory" on Table 8 is that, for $\mathbf{m}$(c1) = 4, the size of PS is too large, so that not even a bit-vector of that size could be allocated.

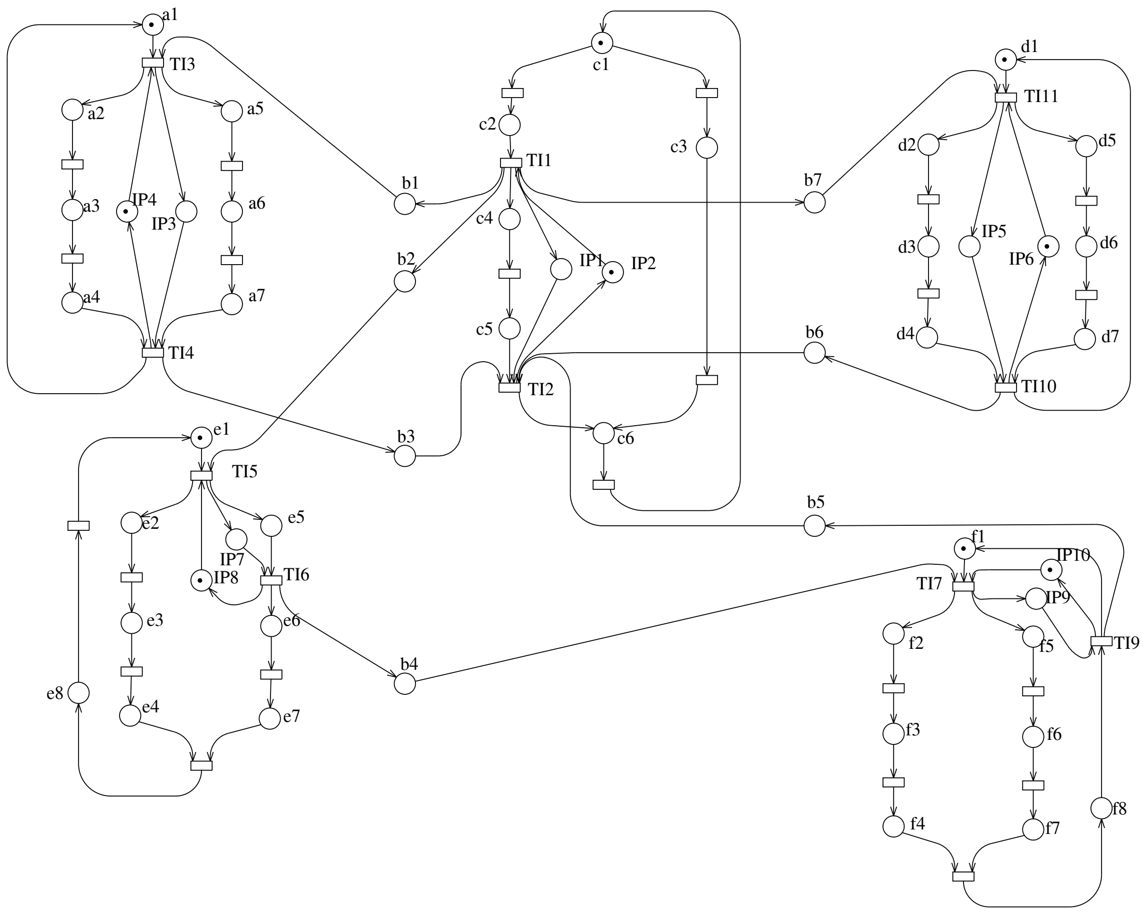

Figure 11: The third example.

	$\mathbf{m}(\mathsf{c1})$=1	$\mathbf{m}(\mathsf{c1}) = 2$	$\mathbf{m}(\mathsf{c1}) = 3$	$\mathbf{m}(\mathsf{c1}) = 4$
$\mathcal{BS}$	4	10	20	35
$\mathcal{LS}_1$	730	10.852	77.472	373.770
$\mathcal{LS}_2$	115	442	1.110	2.240
RPS	25.926	536.277	4.557.376	24.613.188
RS	25.926	536.277	4.557.376	24.613.188

Table 6: SAM technique: State spaces computed for the model of Figure 11 (decomposition into two components).

	$\mathbf{m}(\mathsf{c1})=1$	$\mathbf{m}(\mathsf{c1}) = 2$
GSPN1	10	10
GSPN2	768	45.927
GSPN3	11	11
GSPN4	11	11
GSPN5	10	10
PS	9.292.800	555.716.700
RS	25.926	536.277

Table 7: State spaces computed for the model of Figure 11, using SupGSPN, with a decomposition into five components.

	$\mathbf{m}(\mathsf{c1})=1$	$\mathbf{m}(\mathsf{c1}) = 3$	$\mathbf{m}(\mathsf{c1}) = 4$
GSPN1	246	11.576	41.886
GSPN2	968	7.744	15.125
PS	25.926	4.557.376	633.525.750
RS	25.926	4.557.376	no memory

Table 8: State spaces computed for the model of Figure 11, using SupGSPN, with a decomposition into two components.

7 Conclusions

A technique for computing exact performance indices of P/T systems (observed as structured with a SAM view) has been presented. It uses *linear algebra*-based structure theory for producing net decompositions and *tensor algebra* to express and solve the stochastic part. The decomposition phase builds a set of components: *low level systems* (in which everything is abstracted except one of the modules) and an abstract representation of the full system, called the *basic skeleton*. Basic skeleton and low level systems provide a two levels description of the model, leading to a technique that reduces the "spurious" markings that can appear in the product space PS (Cartesian product of the state spaces of the components), through the construction of the restricted product space RPS (union of Cartesian products of state subspaces).

In a way, we can consider the technique based on the basic skeleton as a method to use high level information to cut from PS($\mathcal{S}$) states that are not reachable, and consequently, from the $\mathbf{G}$ matrix, rows and columns that correspond to non-reachable states. The price we have to pay is a more complex expression for the supermatrix of the infinitesimal generator, and therefore a more complex storage scheme for the matrices, but the order of complexity of the solution does not change.

The technique does not guarantee that RPS = RS, but for live and bounded nets we have encountered a large number of cases in which RPS = RS. Moreover, for the flat (i.e., single level) solution method the ratio |PS|/|RS| tends to increase as we increase the number of modules, what limits in practice the efficiency of the divide and conquer strategy underlying the tensorial approach. For the two levels method, the decomposition in a larger number of modules appears to be a better strategy, assuming that the modules are anyhow "big enough" to exhibit some kind of local behaviour. In practice, and as a rule of thumb for our approach, the system should be decomposed in several (easily tractable) modules with the same order of magnitude for their respective state spaces. Of course, if |RS| is so large that no steady-state probability vector can be allocated, then even this approach remains infeasible.

Some numerical examples have illustrated the technique and gave informal insight on the computational behaviour of the approach.

Single and two levels methods are orthogonal to net classes or views; indeed we have shown on a small example (section 2.3) how the two levels method can be applied to SGSPN. On the other side, a flat solution for SAM can be trivially derived by considering as components of the solution only the set of low level systems $\mathcal{LS}_i$, synchronized over interface transitions.

The reader should notice that we have considered stochastic PN's without *immediate* transitions: a possible way to consider the full GSPN class is to extend the approach in [14] (that preserves vanishing states in the solution process), to the two levels view. Another technique could be that of removing immediate transitions with the technique in [12], but this reduction process can change the input and output arcs of buffers, so that for the resulting SPN it may be necessary to define a different SAM view.

In this paper we only considered single server transitions, but marking-dependent policies can be considered, as explained in [5, 14]. A certain attention should be paid when defining $\mathcal{ES}$, and, consequently, the low level systems: the addition of implicit places may change the enabling degree of transitions, and multiple and infinite server transitions in the original model may have to be translated into marking dependent ones in the $\mathcal{ES}$ system.

Acknowledgments

We would like to thank Laura Recalde for the many clarifying discussions, Peter Kemper for providing us his prototype implementation of SupGSPN, and the anonymous referees for the helpful comments.

A Tensor operators

Kronecker operators are defined in terms of rectangular matrices, but for the sake of this presentation we use square matrices for the Kronecker sum, and rectangular ones for the Kronecker product.

In the following we shall consider matrices on real values.

Definition 14 *Let* $\mathbf{A}$ *be a* $n \times m$ *matrix, and* $\mathbf{B}$ *be a* $p \times q$ *one;* $\mathbf{C}$ *is the tensor (Kronecker) product of* $\mathbf{A}$ *and* $\mathbf{B}$ *and we write* $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ *iff* $\mathbf{C}$ *is a* $n \cdot p \times m \cdot q$ *matrix defined by:*

$$\mathbf{C} = \{c_{\bar{\imath}\bar{\jmath}} : c_{\bar{\imath}\bar{\jmath}} = a_{i_1 j_1} b_{i_2 j_2} \text{ with } \bar{\imath} = (i_1, i_2), \bar{\jmath} = (j_1, j_2)\}$$

As a simple example consider the tensor product of a 2×2 matrix, with a 2×3. We have

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix}$$

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} = \left(\begin{array}{ccc|ccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \end{array}\right)$$

In case of square matrices $\mathbf{A}$ and $\mathbf{B}$ can be interpreted as the matrices of transition probabilities of two discrete time Markov chains, it is immediate to recognize (see Davio in [18]) that $\mathbf{C}$ is the transition probabilities matrix of the process obtained as independent composition of the two original processes.

Let us now define the Kronecker (or tensor) sum of two square matrices

Definition 15 *Let* $\mathbf{A}$ *be a* $n \times n$ *matrix, and* $\mathbf{B}$ *be a* $p \times p$ *one;* $\mathbf{D}$ *is the tensor (Kronecker) sum of* $\mathbf{A}$ *and* $\mathbf{B}$ *and we write* $\mathbf{D} = \mathbf{A} \oplus \mathbf{B}$ *iff* $\mathbf{D}$ *is a* $n \cdot p \times n \cdot p$ *matrix defined by:*

$$\mathbf{D} = \mathbf{A} \oplus \mathbf{B} = \mathbf{A} \otimes \mathbf{Id}_p + \mathbf{Id}_n \otimes \mathbf{B}$$

where $\mathbf{Id}_k$ *is the* $k \times k$ *identity matrix.*

Let us consider again the two matrices $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}$ is the same as before, and $\mathbf{B}$ is the same matrix as before, but for the last column, which is missing. The computation of their tensor (Kronecker) sum is:

$$\mathbf{A} \otimes \mathbf{Id}_2 = \left(\begin{array}{cc|cc} a_{11} & 0 & a_{12} & 0 \\ 0 & a_{11} & 0 & a_{12} \\ \hline a_{21} & 0 & a_{22} & 0 \\ 0 & a_{21} & 0 & a_{22} \end{array}\right)$$

$$\mathbf{Id}_2 \otimes \mathbf{B} = \left(\begin{array}{cc|cc} b_{11} & b_{12} & 0 & 0 \\ b_{21} & b_{22} & 0 & 0 \\ \hline 0 & 0 & b_{11} & b_{12} \\ 0 & 0 & b_{21} & b_{22} \end{array}\right)$$

$$\mathbf{D} = \left(\begin{array}{cc|cc} a_{11}+b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11}+b_{22} & 0 & a_{12} \\ \hline a_{21} & 0 & a_{22}+b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{22}+b_{22} \end{array}\right)$$

Again if we consider $\mathbf{A}$ and $\mathbf{B}$ as the infinitesimal generator of two continuous time Markovian processes, then $\mathbf{D}$ is the infinitesimal generator of the process obtained by independent composition of the two original one. It can be observed that in $\mathbf{D}$ all transition rates among states that differ by more than a single component are set equal to zero, as it is the case when we are in a continuous-time environment.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* J. Wiley, 1995.

[2] P. Buchholz. A hierarchical view of GCSPN's and its impact on qualitative and quantitative analysis. *Journal of Parallel and Distributed Computing*, 15(3):207–224, July 1992.

[3] P. Buchholz. Numerical solution methods based on structured descriptions of Markovian models. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation. Modeling Techniques and Tools*, pages 251–267. Elsevier, 1992.

[4] P. Buchholz. A class of hierarchical queueing networks and their analysis. *Queueing Systems*, 15:59–80, 1994.

[5] P. Buchholz, G. Ciardo, S Donatelli, and P. Kemper. Complexity of Kronecker operations on sparse matrices with applications to the solution of Markov models. Icase report 97-66, Institute for Computer Applications in Science and Engeneering, Hampton, VA, 1997.

[6] P. Buchholz and P. Kemper. Numerical analyisis of stochastic marked graphs. In *Proc. 6th Intern. Workshop on Petri Nets and Performance Models*, pages 32–41, Durham, NC, USA, October 1995. IEEE-CS Press.

[7] J. Campos, G. Chiola, and M. Silva. Properties and performance bounds for closed free choice synchronized monoclass queueing networks. *IEEE Transactions on Automatic Control*, 36(12):1368–1382, December 1991.

[8] J. Campos, J. M. Colom, H. Jungnitz, and M. Silva. Approximate throughput computation of stochastic marked graphs. *IEEE Transactions on Software Engineering*, 20(7):526–535, July 1994.

[9] J. Campos, S. Donatelli, and M. Silva. Structured solution of stochastic DSSP systems. In *Proc. of the 7th Intern. Workshop on Petri Nets and Performance Models*, pages 91–100, Saint Malo, France, June 1997. IEEE-CS Press.

[10] G. Chiola. A graphical Petri net tool for performance analysis. In *Proceedings of the 3rd International Workshop on Modeling Techniques and Performance Evaluation*, Paris, France, March 1987. AFCET.

[11] G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Transactions on Software Engineering*, 19(2):89–107, February 1993.

[12] G. Chiola, S. Donatelli, and G. Franceschinis. GSPNs versus SPNs: What is the actual role of immediate transitions? In *Proc. of the 4th Intern. Workshop on Petri Nets and Performance Models*, pages 20–31, Melbourne, Australia, December 1991. IEEE-CS Press.

[13] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42(11), November 1993.

[14] G. Ciardo and M. Tilgner. On the use of Kronecker operators for the solution of generalized stochastic Petri nets. Icase report 96-35, Institute for Computer Applications in Science and Engeneering, Hampton, VA, 1996.

[15] J. M. Colom. *Análisis Estructural de Redes de Petri, Programación Lineal y Geometría Convexa.* PhD thesis, Departamento de Ingeniería Eléctrica e Informática, Universidad de Zaragoza, Spain, June 1989. Research Report. GISI-RR-89-11. In Spanish.

[16] J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. A comparative study of algorithms for computation of minimal p-semiflows. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 79–112. Springer-Verlag, Berlin, 1991.

[17] J. M. Colom and M. Silva. Improving the linearly based characterization of P/T nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 113–145. Springer-Verlag, Berlin, 1991.

[18] M. Davio. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, 30(2):116–125, 1981.

[19] F. DiCesare, G. Harhalakis, J. M. Proth, M. Silva, and F.B. Vernadat, editors. *Practice of Petri Nets in Manufacturing.* Chapman & Hall, London, 1993.

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

[21] S. Donatelli. Superposed generalized stochastic Petri nets: Definition and efficient solution. In R. Valette, editor, *Proc. of the* 15^{th} *Intern. Conference on Applications and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 258–277. Springer-Verlag, Berlin Heidelberg, 1994.

[22] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, London, 1985.

[23] S. Haddad and P. Moreaux. Evaluation of high level Petri nets by means of aggregation and decomposition. In *6-th International Conference on Petri Nets and Performance Models - PNPM95*, pages 11–20. IEEE Computer Society, 1995.

[24] S. Haddad and P. Moreaux. Asynchronous composition of high level Petri nets: A quantitative approach. In J. Billington and W. Reisig, editors, *Proc. of the* 17^{th} *Intern. Conference on Applications and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pages 192–211. Springer-Verlag, Berlin, 1996.

[25] W. Henderson and P.G. Taylor. Aggregation methods in exact performance analysis of stochastic Petri nets. In *Proc.* 3^{rd} *Intern. Workshop on Petri Nets and Performance Models*, pages 12–18, Kyoto, Japan, December 1989. IEEE-CS Press.

[26] P. Kemper. Numerical analyisis of superposed GSPN. *IEEE Transactions on Software Engineering*, 22(4):615–628, September 1996.

[27] P. Kemper. SupGSPN Version 1.0. An analysis engine for superposed GSPNs. Technical report, Universität Dortmund, 1997.

[28] P. Kemper. Transient analysis of superposed GSPNs. In *7-th International Conference on Petri Nets and Performance Models - PNPM97*, pages 101–110. IEEE Computer Society, 1997.

[29] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9):913–917, September 1982.

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

[31] C. J. Pérez-Jiménez, J. Campos, and M. Silva. State machine reduction for the approximate performance evaluation of manufacturing systems modelled with cooperating sequential processes. In *Proc. of the 1996 IEEE Intern. Conf. on Robotics and Automation*, pages 1159–1165, Minneapolis, Minnesota, USA, April 1996.

[32] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. 1985 SIGMETRICS Conference*, pages 147–154, Austin, TX, USA, August 1985. ACM.

[33] L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: The case of Deterministic Systems of Sequential Processes. In J. Desel, editor, *Structures in Concurrency Theory, Berlin 1995*, pages 279–293. Springer, 1995.

[34] L. Recalde, E. Teruel, and M. Silva. Modeling and analysis of sequential processes that cooperate through buffers. *IEEE Trans. on Robotics and Automation*, 14(2):267–277, 1998.

[35] M. Sereno and G. Balbo. Mean value analysis of stochastic Petri nets. *Performance Evaluation*, 29(1):35–62, 1997.

[36] M. W. Shields. *An Introduction to Automata Theory.* Blackwell Scientific Publications, 1987.

[37] M. Silva. Introducing Petri nets. In DiCesare et al. [19], chapter 1.