Aggregation of Stochastic Automata Networks with Replicas

A. Benoit[(1)], L. BRENNER[(2)], P. FERNANDES [(2)] and Brigitte Plateau[(1)]

[(1)] Informatique et Distribution, INRIA Rhones Alpes
51, Av. Jean Kuntzman, 38330 Montbonnot, France
[(2)] PUCRS, Faculdade de Informática, Av. Ipiranga, 6681 90619-900 Porto Alegre, Brazil

Abstract

We present techniques for computing the solution of large Markov chain models whose generators can be represented in the form of a generalized tensor algebra, such as Stochastic Automata Networks (SAN). Many large systems include a number of replications of identical components. This paper exploits replication by aggregating similar components. This leads to a state space reduction, based on lumpability. We define SAN with replicas, and we show how such SAN models can be strongly aggregated, taking functional rates into account. A tensor representation of the matrix of the aggregated Markov chain is proposed, allowing to store this chain in a compact manner and to handle larger models with replicas more efficiently. Examples and numerical results are presented to illustrate the reduction in state space and, consequently, the memory and processing time gains.

Keywords: large Markov chains, Stochastic Automata Networks, generalized tensor algebra, replication, lumpability, strong aggregation, PEPS software tool

1 Introduction

The high complexity of dynamic systems in many areas of application makes them difficult to analyze [26]. Continuous time Markov chains (CTMC) facilitate their performance and even reliability analysis. CTMC are often used as the underlying concept of a high level formalism interpreted by a software package, which generates the state space and the infinitesimal generator of the underlying CTMC, and computes stationary and transient solutions.

The primary difficulty in developing a software tool to handle large-scale Markov chains comes from the explosion in the number of states. Indeed, CTMC modeling real systems are usually huge and sophisticated algorithms are needed to handle them. In order to keep memory requirements manageable, Stochastic Automata Networks (SAN) were introduced [10, 21]. The SAN formalism allows Markov chains models to be described in a memory efficient manner due to their storage based on a tensor representation. A somewhat different approach based on Stochastic Petri Nets allows us to obtain a similar tensor formalism, as shown by Donatelli [8, 9]. Furthermore, analysis techniques for these formalisms have been proposed. Direct solution methods, such as Gaussian elimination, are generally not used because the amount of fill-in that occurs necessitates a prohibitive amount of storage space. Iterative methods, which can take advantage of sparse storage techniques to hold the infinitesimal generator, are more appropriate [11, 21, 26], even though here also, memory requirements can become too large for real life models. In order to analyze large CTMC models with loosely coupled blocks, some iterative aggregation/disaggregation (a/d) methods have been proposed in [26], and an a/d algorithm has also been proposed for SAN in [4].

In fact, it will be necessary to develop techniques to reduce the complexity of the Markov chain that will be analyzed. Fortunately, many large real systems include a considerable large number of identical (replicated) components. Taking such replications of components into account, a reduced Markov chain resulting from strong aggregation [17, 22] can be generated. Previous studies on weak lumpability [2, 19] have also shown how to group identical states, but this kind of aggregation depends on the initial state of the model. Lumpability and equivalence relations have been defined and discussed in [3, 6, 13, 14, 24].

In the previous approaches, lumping on the state space of a Markov chain is described. Some other techniques to exploit replications are based on **hierarchical models** [20]. These techniques generate the reduced Markov chain directly from the model specification. Hierarchical Markovian models are useful for analyzing complex systems, and techniques to generate a reduced Markov chain from the specification of the model have been developed. Some a/d algorithms can also be applied to such models [5]. For Stochastic Activity Networks (a particular class of Stochastic Petri Nets), an algorithm for reducing the state space in the presence of replicated subsystems has been developed [23]. Some similar techniques exist for finite states machines, as shown in [7]. In [16], a symmetric composition for Stochastic Process Algebras is proposed. Its operational semantics is compact and intuitive, and it allows a compact description of systems with replicas.

All these approaches present lumpability conditions for various modeling techniques in order to reduce the state space. The goal of this paper is to present an equivalent technique that can be used to efficiently aggregate **SAN models**. The lumpability conditions, as shown earlier in [17, 22], are described and used in order to demonstrate that a SAN with replicas can be strongly aggregated. Identical automata within the model are detected and they can be automatically grouped and aggregated in order to generate the reduced Markov chain. A similar approach is described by Siegle in [25], but it does not take functional rates, one of the major components of SAN models, into account. In his work [3], Buchholz defines the equivalence relations for Stochastic Automata Networks, and particularly equivalent representations for a Stochastic Automata. Gusak et al. [14, 15] have provided relatively general lumpability conditions for continuous-time and discrete-time SAN; these conditions are checked on automata matrices and they do not require exact replication of components. All these approaches emphasis on equivalence relations, but they do not give any formal definition of replicas for SAN. Moreover they do not give a tensorial expression of the matrix of the aggregated Markov chain.

In the next section, we briefly recall the concept of SAN, and then we formally define SAN with replicas in Section 3. The strong aggregation of such SAN is described in Section 4, and a tensorial expression of the matrix of the aggregated Markov chain is proposed. The final section shows the benefits of aggregation and some numerical results of practical examples.

2 Stochastic Automata Networks (SAN)

Continuous-time Stochastic Automata Networks [10, 11, 21] describe a system as a set of subsystems that interact. Each subsystem is modeled by a stochastic automaton, and some rules between the states of each automaton describe the interactions between subsystems.

Each automaton is composed of states, called *local states*, and transitions among them. Transitions on each automaton are labeled with the list of the events that may trigger it. An event is triggered after a delay which is exponentially distributed and the exponentially distributed variables corresponding to each event are independent. Each event is defined by its name and its rate.

When the occurrence of the same event can lead to different target states, a probability of occurrence is assigned to each possible transition. The label on the transition is *evt*(*prob*), where *evt* is the event name, and *prob* is the probability of occurrence of each possible transition. When only one transition is possible, the probability can be omitted.

There are basically two ways in which stochastic automata interact. Firstly, the rate at which an event may occur can be a *function* of the state of other automata. Such rates are called *functional* rates. Rates that are not functional are said to be *constant* rates. The probabilities of occurrence can also be functional. Secondly, an event may involve more than one automaton: the occurrence of such event triggers transitions in two or more automata at the same time. Such events are called *synchronizing* events, in opposition to events involving only one automaton, called *local* events. As local events, synchronizing events may have constant or functional rates and probabilities.

Consider a SAN model with N automata and E events. It is a N-component Markov chain whose components are not necessarily independent (due to the possible presence of functional rates and synchronizing events). A local state of i-th automaton ($\mathcal{A}^{(i)} \mid i = 1..N$) is denoted $x^{(i)}$ while the complete set of states for this automaton is denoted $S^{(i)}$, and the cardinality of $S^{(i)}$ is denoted by n_i. $\hat{S} = S^{(1)} \times \ldots \times S^{(N)}$ is called the product state space, and its cardinality is equal to $\prod_{i=1}^{N} n_i$. A global state for the model is a vector $x = (x^{(1)}, \ldots, x^{(N)}) \in \hat{S}$. The reachable state space of the model is denoted by S; it is generally smaller than the product state space, most of the time because of synchronizing events and functional rates which prevent some states in $\hat{S}$ from occurring.

An automaton is involved by an event if it has at least one transition labeled by this event. The set of automata involved by an event e is denoted by O_e. The event e can occur if, and only if, all the automata in O_e are in a local state from which one of those transitions can be triggered. When it occurs, all the corresponding transitions are triggered. Notice that for a local event e, O_e is reduced to the automaton involved by this event, and that only one transition occurs.

For $i = 1..N$, the behavior of continuous-time automaton $\mathcal{A}^{(i)}$ is described by a set of square matrices, all of order n_i. We shall denote the set of synchronizing events by $\mathcal{ES}$. Let us denote, for $i = 1..N$, and for $e \in \mathcal{ES}$:

- $Q_l^{(i)}$ the matrix consisting only of the transitions that are local to automaton $\mathcal{A}^{(i)}$;

- $Q_{e+}^{(i)}$ the positive synchronization matrix of $\mathcal{A}^{(i)}$, which represents the occurrence of the synchronizing event e and its rates;
- $Q_{e-}^{(i)}$ the negative synchronization matrix of $\mathcal{A}^{(i)}$, which corresponds to an updating of the diagonal elements for event e; it is defined by $Q_{e+}^{(i)}$ and its elements are not necessarily negative as suggested by the name.

Notice that if $\mathcal{A}^{(i)}$ is not in O_e, then $Q_{e+}^{(i)}$ and $Q_{e-}^{(i)}$ are identity matrices.
Then, it has been shown in [10, 11] that the transition matrix can be expressed as:

$$Q = \bigoplus_{i=1}^{N}{}_{g}\, Q_l^{(i)} + \sum_{e\in\mathcal{ES}} \left(\bigotimes_{i=1}^{N}{}_{g}\, Q_{e+}^{(i)} + \bigotimes_{i=1}^{N}{}_{g}\, Q_{e-}^{(i)} \right) \quad (1)$$

The tensor sum corresponds to the analysis of the local events, while the tensor products correspond to the analysis of the synchronizing events. Notice the use of generalized tensor algebra [10, 21], *i.e.*, the use of operators $\bigoplus_g$ and $\bigotimes_g$, instead of $\oplus$ and $\otimes$.

3 SAN models with replicas

Although large systems often contain identical components, we turn our attention to two different cases: systems where all subsystems are equal, and systems where only some sets of subsystems are equal among themselves. The first case is modeled by a *SAN composed of one replica*, and the second case is modeled by a *SAN with multiple replicas.* In this section we formally define *SAN models composed of one replica* and *SAN models with multiple replicas* with an illustrating example for each case.

3.1 SAN models composed of one replica

Informally, a SAN composed of one replica consists of a set of N identical automata, *i.e.*, the states of each automaton are identical, and the transitions are labeled with identical synchronizing events or replicated local events (for a given transition, the local events have the same rate in each automaton). This implies that the synchronizing events involve all replicated automata. Moreover, we have a replica only if the functions are not changed by a permutation of the parameters. For example, if $N = 2$, for all functional rate f and for all $x^{(1)} \in S^{(1)}$ and $x^{(2)} \in S^{(2)}$, we must have $f(x^{(1)}, x^{(2)}) = f(x^{(2)}, x^{(1)})$ (remember that $S^{(i)}$ is the state space of automaton $\mathcal{A}^{(i)}$). Formalizing the definition of such SAN:

> **Definition 1 -** *A SAN composed of* **one replica** *is a SAN model with N automata, such that, for $i = 1..N$, we have:*
> *- all local matrices $Q_l^{(i)}$ are identical (equal to Q_l);*
> *- for every synchronizing event $e \in \mathcal{ES}$, all matrices $Q_{e+}^{(i)}$ and all matrices $Q_{e-}^{(i)}$ are identical and respectively equal to Q_{e+} and Q_{e-}; only one automaton hold the transition rate f_e of the event (matrices f_eQ_{e+} and f_eQ_{e-});*
> *- for all functional rates f, for all permutations σ of $[1..N]$, and for each global state $x = (x^{(1)}, \ldots, x^{(N)})$, $f(x) = f(\sigma(x))$, where $\sigma(x) = (x^{\sigma(1)}, \ldots, x^{\sigma(N)})$*
> *(the functions are not changed by a permutation of the parameters).*

The concept of a SAN composed of one replica is now illustrated through a small example.

3.2 The basic resource sharing model – RS1

In this model, N_p distinguishable processes share N_r identical units of a common resource. Each of these processes alternates between a *sleeping* state and a resource *using* state. Notice that when $N_r = 1$ this model reduces to the usual mutual exclusion problem and when $N_r = N_p$ all of the processes are independent. Let $\lambda^{(i)}$ be the rate at which process i awakes from the sleeping state wishing to access the resource, and let $\mu^{(i)}$ be the rate at which this same process releases the resource after using.

Each process is modeled by a two-state automaton $\mathcal{A}^{(i)}$, the two states being *sleeping* and *using*. Assuming $\delta(b)$ a function that equals 1 if the expression b is true, otherwise this function equals 0, then let the function f be defined by:

$$f(x) = \delta\left(\left(\sum_{i=1}^{N_p} \delta(x^{(i)} == using) \right) < N_r \right) \quad (2)$$

where $x^{(i)}$ is the local state of automaton $\mathcal{A}^{(i)}$, and $x = (x^{(1)}, \dots, x^{(N_p)})$ is the global state of the SAN. Thus the function f has the value 1 when access to the resource is permitted (there is at least one available resource) and has the value 0 otherwise.

When **all rates are identical** ($\lambda^{(1)} = ... = \lambda^{(N_p)} = \lambda$ and $\mu^{(1)} = ... = \mu^{(N_p)} = \mu$), there are two local events per process:

- $evt_a^{(i)}$ corresponds to "acquiring a resource" by the i-th process and it has a rate λf;
- $evt_r^{(i)}$ corresponds to "releasing a resource" by the i-th process and it has a rate μ.

This model, called *RS1*, is graphically illustrated in Figure 1.

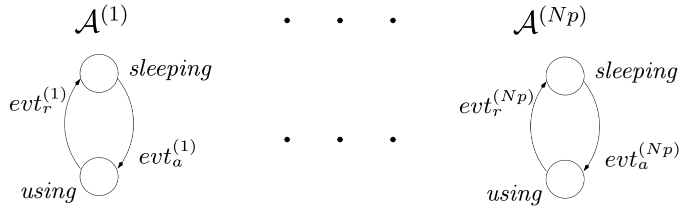

Figure 1: Basic resource sharing model – RS1

Since this model does not have synchronizing events, there is only one matrix per process:

$$Q_l^{(i)} = \begin{pmatrix} -\lambda f & \lambda f \\ \mu & -\mu \end{pmatrix}$$

All local matrices are identical, and the function is not changed by a permutation of the parameters (commutativity of the sum in the definition of f, equation 2). This SAN is therefore a **SAN composed of one replica**, as defined in Definition 1.

3.3 SAN models with multiple replicas

Definition 2 - *Assuming a SAN with N automata, we define a* **partition** *$\mathcal{G}$ such that there are K contiguous subsets of replicated automata ($K \in [1..N]$). Then let k_h be the last automaton index of the h-th subset, called subset h ($h \in [1..K]$). Assuming arbitrarily $k_0 = 0$, we may denote by SI_h the set of indexes of the automata in subset h: $SI_h = (k_{h-1} + 1, \dots, k_h)$ and by $R_h = k_h - k_{h-1}$ the number of automata in the subset h ($R_h = Card(SI_h)$).*

An illustration of this decomposition into K subsets is given in Figure 2.

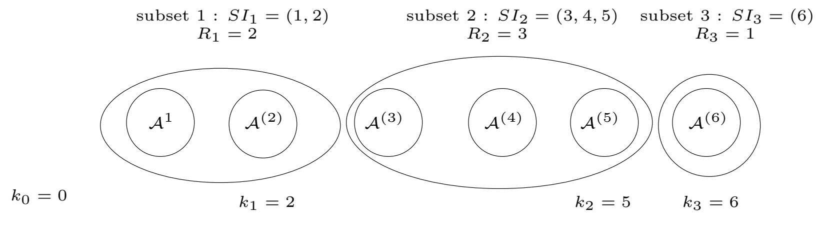

Figure 2: Decomposition into subsets of a SAN with $N = 6$ and $K = 3$

First let us define a set of permutations such that there can be an exchange inside each subset, but not between different subsets:

Definition 3 - *For a given SAN and a partition $\mathcal{G}$ (Definition 2), let $\mathcal{P}$ be the* **set of permutations** *of $[1..N]$ such that*

$$\sigma \in \mathcal{P} \iff \forall h \in [1..K],\ \forall i \in SI_h,\ \sigma(i) \in SI_h$$

For $h \in [1..K]$, let $\mathcal{P}_h$ be the set of permutations inside the corresponding subset (permutations of $[(k_{h-1} + 1)\ ..\ k_h]$). $\sigma \in \mathcal{P}$ can be expressed as a combination *of permutations $\sigma_h \in \mathcal{P}_h$, denoted by $\sigma = (\sigma_1 \ ... \ \sigma_K)$.*

State vector structure due to multiple replicas
Let $x = (x^{(1)}, \dots, x^{(N)})$ be a global state of the SAN.
It can be decomposed into subsets, $x = (x_1 \dots x_K)$ where x_h $(h \in [1..K])$ is a vector such that $x_h = (x^{(k_{h-1}+1)}, \dots, x^{(k_h)})$.
Then, for $\sigma \in \mathcal{P}$, denote $\sigma(x) = (x^{\sigma(1)}, \dots, x^{\sigma(N)})$.
If $\sigma = (\sigma_1 \dots \sigma_K)$ is defined as above,
$\sigma(x) = (\sigma_1(x_1) \dots \sigma_K(x_K))$, and for $h \in [1..K]$, $\sigma_h(x_l) = (x^{\sigma(k_{h-1}+1)}, \dots, x^{\sigma(k_h)})$

Note that the composition of permutations (denoted by $\circ$) is closed within $\mathcal{P}$, that is to say: let $\varphi, \tau \in \mathcal{P}$, then $\sigma = \varphi \circ \tau$ is also in $\mathcal{P}$. It is also obvious that when σ_h goes through $\mathcal{P}_h$, for all $h \in [1..K]$, then σ goes through $\mathcal{P}$. For $K = 1$ (SAN composed of one replica), $\mathcal{P}$ equals the set of permutations of $[1..N]$.

Considering the example in Figure 2 with each automaton having two local states (1 and 2), the global state $x = (\mathbf{2}, \mathbf{1}, 1, 1, 2, \mathbf{1})$ means that $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(4)}$ and $\mathcal{A}^{(6)}$ are in local state 1, while $\mathcal{A}^{(1)}$ and $\mathcal{A}^{(5)}$ are in local state 2. Let σ be the permutation such that $\sigma(x) = (\mathbf{1}, \mathbf{2}, 1, 2, 1, \mathbf{1})$. We have in both global states (x and $\sigma(x)$):

- one automaton from the first subset, two automata from the second subset, and one automaton from the third subset in state 1; and
- one automaton from the first subset, one automaton from the second subset, and no automata from the third subset in state 2.

Having the same number of automata in all local states means that we can exchange global states with a permutation of $\mathcal{P}$ (Definition 3).

A formal definition of SAN models with multiple replicas can be as follows:

Definition 4 - *A SAN with* **multiple replicas** *is a SAN model with N automata and a partition $\mathcal{G}$ (Definition 2), such that, for $h = 1..K$, we have:*
- for all $j \in SI_h$, the local matrices $Q_l^{(j)}$ are identical (equal to $Q_{(l,h)}$);
- for every synchronizing event $e \in \mathcal{ES}$, for almost all $j \in SI_h$, all matrices $Q_{e+}^{(j)}$ and $Q_{e-}^{(j)}$ are identical and respectively equal to $Q_{(e+,h)}$ and $Q_{(e-,h)}$. Only one automaton in the whole SAN holds the transition rate f_e of the event and therefore this automaton has different matrices. When this automaton is in subset h, its matrices are respectively equal to $f_e Q_{(e+,h)}$ and $f_e Q_{(e-,h)}$;
- for all functional rates f, for all permutations $\sigma \in \mathcal{P}$ (Definition 3), and for each global state x, $f(x) = f(\sigma(x))$ (for each subset h, the function is not changed by a permutation of the parameters "state of $\mathcal{A}^{(k_{h-1}+1)}$" to "state of $\mathcal{A}^{(k_h)}$").

A SAN is said to be **without replica** if there are N subsets with only one automaton in each subset ($K = N$ and $k_h = h$ for $h = 1..N$). On the other hand, a **SAN composed of one replica** is a SAN for which all automata are in the same subset ($K = 1$), as for the *RS1* example presented in Section 3.2. A **SAN with multiple replicas** is therefore a SAN for which $1 < K < N$.

Note also that synchronizing events may affect several subsets (as well as only one subset). The conditions in the above definitions expresses that automata in the same replica have the same behavior with respect to all synchronizing events.

3.4 The resource sharing model with different rates – RS2

In this model (Figure 3), we consider N_p processes partitioned in K groups with different rates for acquiring and releasing N_r units of a common resource (the N_r units of resource can be used by any process of any group). For each process of each group ($h \in [1..K]$, $i \in SI_h$), we have two local events:

- $evt_{a,h}^{(i)}$ is the local event corresponding to "acquiring a resource" by process i of group h. It has rate $\lambda_h f$ (f has the definition given in Equation 2 for the example *RS1*);
- $evt_{r,h}^{(i)}$ is the local event corresponding to "releasing a resource" by process i of group h. It has rate μ_h.

There are no synchronizing events, so we have only local matrices. For $h = 1..K$ and $i \in SI_h$,

$$Q_{l,h}^{(i)} = \begin{pmatrix} -\lambda_h f & \lambda_h f \\ \mu_h & -\mu_h \end{pmatrix}$$

Inside each subset, the local matrices are identical, and the function f (Equation 2) is not changed by a permutation of the parameters. Following Definition 4, this SAN is therefore a **SAN with multiple replicas**.

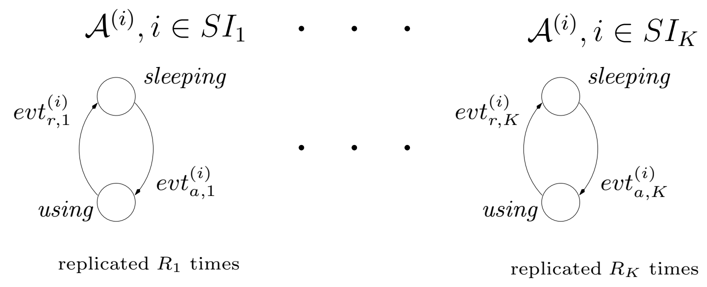

Figure 3: Resource sharing model, different rates – RS2

3.5 How to detect replicas in a SAN model

Since large systems are often described with identical components, it is usual to describe these identical components as replicas during the model specification. So the partition $\mathcal{G}$ of replicated automata is usually given by the user during the model specification phase. In fact, we just have to be sure that the properties of *SAN with multiple replicas* (or *with one replica*) are checked for the user defined model. Therefore we do not have to detect the subsets, because they are given by the model itself.

The current implementation of replica detection in the PEPS software tool [1] is based on the verification of properties for automata informed as identical during the model specification. Specifically, to each set of automata informed as a replica (using brackets []) any functional rates and probabilities are verified to comply with Definition 4. Such verification is quite simple and its computational cost is irrelevant.

4 Aggregation of Replicas

Now that we have introduced the notion of replicas in SAN, we can proceed to model aggregation. In this following, we assume that we have an initial SAN and a decomposition into subsets, as defined in Definition 2. Our goal is to obtain a reduced Markov chain resulting from strong aggregation (the formal definition of strong aggregation will be given in Section 4.2). Firstly, it is shown how aggregation intuitively works on a small example. After that, we prove that a SAN with multiple replicas can be strongly aggregated. Finally, it is shown that the matrix of the aggregated Markov chain can still be written as a tensor expression.

4.1 Aggregation example

We show how aggregation works on the basic resource sharing model *RS1* (Figure 1, Section 3.2), with $N_p = 3$ processes and $N_r = 2$ resources. This SAN is composed of one replica, as shown previously, *i.e.*, $\mathcal{P}$ is the set of permutations of $[1..N_p]$.

All equivalent states of the initial SAN within a permutation of $\mathcal{P}$ (Definition 3) are grouped into one state of the aggregated Markov chain. We can arbitrarily choose one particular state to represent each group of global states, which we call *equivalence classes*. Assuming 1 as the local state *sleeping* and 2 as the local state *using*, the aggregated Markov chain of this example has four equivalence classes represented by the global states (1,1,1), (1,1,2), (1,2,2), and (2,2,2). All the global states of the initial SAN are equivalent to one of these states. We just need to know *how many* processes are using the resources, but it does not matter *which* process(es) is (are) using them. For example, being in global state (1,1,2) or (1,2,1) means that one resource is currently used.

If $C1$ and $C2$ represent two different equivalence classes, the transition rate from state $C1$ to state $C2$ of the aggregated chain is obtained by summing up the transition rates of the initial SAN from one original global state of $C1$ to all the original global states of $C2$[1]. The aggregated Markov chain for *RS1* is represented in Figure 4, and the formal definition of aggregation is presented in the next section.

Notice that state (2,2,2) is not reachable (only two automata can use the resource simultaneously). The state space of the aggregated Markov chain, denoted by $\hat{S}_{agg}$, may have some unreachable states, which can be suppressed, obtaining the state space S_{agg}.

4.2 Strong aggregation of SAN with replicas

In this section we study the conditions of strong aggregation of SAN with replicas. We consider a partition of the state space $\Omega = (\Omega_1, ..., \Omega_\Gamma)$, and recall the definition of strong aggregation [3, 17, 22].

[1]We will see in the next section that the choice of the state of $C1$ in the initial SAN does not affect the result.

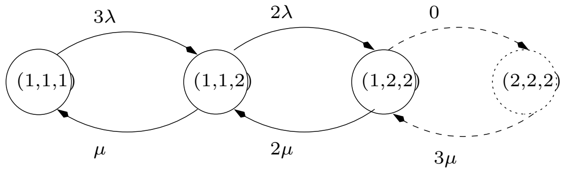

Figure 4: RS1 – $N_p = 3$ and $N_r = 2$ – aggregated Markov chain

Definition 5 (Strong aggregation) - *A Markov chain can be strongly aggregated on the partition Ω if for any initial vector, the aggregated chain (whose states are Ω_γ, for $\gamma \in [1..\Gamma]$) is a Markov chain and the transition rates of this chain do not depend on the initial vector.*

Finally, a condition of strong aggregation is given by the following theorem of Rosenblatt.
A proof of this theorem can be found in [17].

Theorem 1 (Rosenblatt) - *Consider a continuous time Markov chain with state space Ω, and a partition of the state space $\Omega = (\Omega_1, ..., \Omega_\Gamma)$. If, for all $\beta, \gamma \in [1..\Gamma]$, the probability of passing from a state $x \in \Omega_\beta$ to Ω_γ always has the same value for each state x from Ω_β, then the Markov chain can be strongly aggregated on the partition Ω.*

The transition rate from a state Ω_β of the aggregated chain to another state Ω_γ ($\beta, \gamma \in [1..\Gamma]$) is the sum of transition rates in the initial SAN from one state of Ω_β to all the states of Ω_γ. Due to the condition of Rosenblatt, we can arbitrarily choose any of the states of Ω_β, and the result will not be affected.

We have seen the conditions needed to aggregate a Markov chain. We now investigate whether a SAN with multiple replicas can be aggregated, and prove that a SAN with multiple replicas can be strongly aggregated. To do this, we first define the partition of the state space that we want to consider in the theorem of Rosenblatt (Definition 6), then introduce some notation, and finally prove that the condition of Rosenblatt is satisfied for the considered partition (Lemma 1).

Definition 6 - State space partition
Two global states x and y ($x, y \in \hat{S}$) are **equivalent** *if $\exists \sigma \in \mathcal{P} \;\; \sigma(x) = y$.*
For both global states, there is in each subset the same number of automata in a given local state.
The **partition** *that we consider is $\Omega = (\Omega_1, ..., \Omega_\Gamma)$, where each Ω_γ ($\gamma \in [1..\Gamma]$) corresponds to a class of equivalent states.*
All the states in Ω_γ are equivalent, so we can choose for each Ω_γ one particular state $r\gamma \in \Omega_\gamma$, called the **reference** *state, and for all other states $x \in \Omega_\gamma$, we can find a permutation $\tau \in \mathcal{P}$ such that $\tau(x) = r\gamma$.*

Notation:

- If x is a global state of the SAN, $x^{(i)}$ is the local state of $\mathcal{A}^{(i)}$, for $i = 1..N$. Notice that if $\tau \in \mathcal{P}$, then $\tau(x)^{(i)} = x^{(\tau(i))}$.

- If A is a matrix, i and j two indexes, then a_{ij} represents the element of A on row i and column j. It may be functional, and $a_{ij}(x)$ is the function evaluated for the global state x.

- The descriptor Q described in Equation (1) can be decomposed into three parts:
$Q = L + \sum_{e \in \mathcal{ES}} (Pe + Ne)$ where the matrix L corresponds to the local part of the descriptor, Pe to the positive synchronization part for event e, and Ne to the negative synchronization part for event e.

- If x and y are two global states of the SAN, $q_{xy} = l_{xy} + \sum_{e \in \mathcal{ES}} (pe_{xy} + ne_{xy})$ is the rate of passing from state x to state y.

- If x is a global state of the SAN and $\gamma \in [1..\Gamma]$, then $q_{x\Omega_\gamma}$ is the cumulative rate of passing from state x to one of the states of Ω_γ: $\quad q_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} q_{xy}$.
This can also be defined for the matrix L, and for Pe and Ne (where $e \in \mathcal{ES}$):
$l_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} l_{xy}, \quad pe_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} pe_{xy}, \quad ne_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} ne_{xy}$.
Notice that all states in Ω_γ are equivalent within a permutation of $\mathcal{P}$, so we have
$l_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} l_{xy} = \sum_{\sigma \in \mathcal{P}} l_{x\sigma(r\gamma)}$. The same can be written for pe and ne.

Lemma 1 - *The Rosenblatt condition (Theorem 1) is satisfied for the SAN on the partition $\Omega = (\Omega_1, ..., \Omega_\Gamma)$ (Definition 6). In other words, for all $\beta, \gamma \in [1..\Gamma]$, the probability of passing from a state $x \in \Omega_\beta$ to Ω_γ always has the same value for each state x in Ω_β: $\forall x \in \Omega_\beta \;\; q_{x\Omega_\gamma} = q_{r\beta\Omega_\gamma}$ (where $r\beta \in \Omega_\beta$ is the reference state of Ω_β).*

The proof of this Lemma is in the appendix. With the application of the Theorem 1 (Rosenblatt), the Markov chain underlying the SAN can therefore be strongly aggregated on the partition Ω.

The aggregated Markov chain is defined on the set of equivalence classes defined by the permutations of $\mathcal{P}$ (Definition 6). These equivalence classes are built from the product state space of the initial SAN which may contain unreachable states. As a consequence, some of these equivalence classes may also be unreachable. Let us denote by $\hat{S}_{agg}$ this state space of equivalence classes (product state space of the aggregated Markov chain).

4.3 Tensor expression of the matrix of the aggregated Markov chain

This section aims at showing how we can express the matrix of the aggregated Markov chain as a tensor expression.

Recall that the descriptor Q of the initial SAN described in Equation (1) can be decomposed into three parts: $Q = L + \sum_{e \in \mathcal{ES}} (P_e + N_e)$. L is a tensor sum; all P_e and N_e are tensor products.

Let $\tilde{x}$ and $\tilde{y}$ be two global states of the aggregated state space (equivalence classes), and $x \in \tilde{x}$ be any of the global states of the equivalence class (it is a global state of the original state space). The generator of the aggregated Markov chain, denoted by $\tilde{Q}$, is defined by $\tilde{q}_{\tilde{x}\tilde{y}} = \sum_{y \in \tilde{y}} q_{xy} = \sum_{\sigma \in \mathcal{P}} q_{x\sigma(y)}$

Moreover, with the definition of Q, $q_{xy} = l_{xy} + \sum_{e \in \mathcal{ES}} (pe_{xy} + ne_{xy})$, and finally

$$\tilde{q}_{\tilde{x}\tilde{y}} = \sum_{\sigma \in \mathcal{P}} l_{x\sigma(y)} + \sum_{e \in \mathcal{ES}} \left(\sum_{\sigma \in \mathcal{P}} pe_{x\sigma(y)} + \sum_{\sigma \in \mathcal{P}} ne_{x\sigma(y)} \right)$$

Therefore, the matrix $\tilde{Q}$ can be expressed as a sum of matrices: $\tilde{Q} = \tilde{L} + \sum_{e \in \mathcal{ES}} (\tilde{P}_e + \tilde{N}_e)$, where, for each global state $\tilde{x}$ and $\tilde{y}$, and for $a = l$, $a = pe$ or $a = ne$ ($e \in \mathcal{ES}$),
$\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{\sigma \in \mathcal{P}} a_{x\sigma(y)}$

Now we consider a *lumped* version of each matrix A (A can be any of the matrix L, P_e and N_e, where $e \in \mathcal{ES}$), denoted by $\hat{A}$. This *lumped* matrix (defined in the following) is a tensor expression, obtained by the aggregation of each subset of replicated automaton. The equivalent SAN obtained is not defined there, we work only on the matrix expression of this SAN.

We want to prove that $\tilde{Q} = \hat{L} + \sum_{e \in \mathcal{ES}} (\hat{P}_e + \hat{N}_e)$, where $\tilde{Q}$ is the matrix of the aggregated Markov chain. So, we will have a tensor expression of $\tilde{Q}$.

To prove this, we work on each term a, where a can be l, pe or ne ($e \in \mathcal{ES}$), and we show that $\tilde{a}_{\tilde{x}\tilde{y}} = \hat{a}_{\tilde{x}\tilde{y}}$.

We consider first the tensor products P_e and N_e, for $e \in \mathcal{ES}$. Then we explore the case of the tensor sum L.

- Let A be one of the **tensor products** P_e or N_e, $e \in \mathcal{ES}$: $A = \bigotimes_{g\, i=1}^{N} Q^{(i)}$.

Let x and y be two global states. With the definition of generalized tensor product, we have $a_{xy} = \prod_{i=1}^{N} q^{(i)}_{x^{(i)}y^{(i)}}(x)$. So $\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{\sigma \in \mathcal{P}} a_{x\sigma(y)} = \sum_{\sigma \in \mathcal{P}} \prod_{i=1}^{N} q^{(i)}_{x^{(i)}\sigma(y^{(i)})}(x)$

Recall that σ can be expressed as a combination $\sigma = (\sigma_1 \dots \sigma_K)$ (Definition 3). So we have

$$\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{\sigma_1 \in \mathcal{P}_1} \dots \sum_{\sigma_K \in \mathcal{P}_K} \prod_{i=1}^{N} q^{(i)}_{x^{(i)}\sigma(y^{(i)})}(x)$$

If we decompose the product into subsets, we can replace in the product term, σ by the corresponding σ_h, where $h \in [1..K]$, and factorize the independent terms:

$$\tilde{a}_{\tilde{x}\tilde{y}} = \prod_{h=1}^{K} \left(\sum_{\sigma_h \in \mathcal{P}_h} \prod_{i \in SI_h} q^{(i)}_{x^{(i)}\sigma_h(y^{(i)})}(x) \right) \tag{3}$$

Then let us define the lumped matrix $\hat{A}$.

Definition 7 - *Let us consider a SAN with N automata and a partition $\mathcal{G}$ (Definition 2). Then, let A be one of the tensor products of the generator of the SAN, $A = \bigotimes_{g\, i=1}^{N} Q^{(i)}$.*

The **lumped matrix** *$\hat{A}$ is defined as a tensor product of K matrices:*

$$\hat{A} = \bigotimes_{g\, h \in [1..K]} \hat{A}^h,$$

where $\hat{A}^h$, $h \in [1..K]$, is defined by

$$\hat{a}^h_{\tilde{x}_h\tilde{y}_h}(\tilde{x}) = \sum_{\sigma_h\in\mathcal{P}_h}\prod_{i\in SI_h} q^{(i)}_{x^{(i)}\sigma_h(y^{(i)})}(\tilde{x})$$

Recall that $\tilde{x} = (\tilde{x}_1 \ ... \ \tilde{x}_K) = (\tilde{x}^{(1)}, ..., \tilde{x}^{(N)})$.

Then we have

$$\hat{a}_{\tilde{x}\tilde{y}} = \prod_{h=1}^{K} \hat{a}^h_{\tilde{x}_h\tilde{y}_h}(\tilde{x}) = \prod_{h=1}^{K}\sum_{\sigma_h\in\mathcal{P}_h}\prod_{i\in SI_h} q^{(i)}_{x^{(i)}\sigma_h(y^{(i)})}(\tilde{x}) \tag{4}$$

Due to the properties of the functions, when we evaluate them with all $x \in \tilde{x}$, we always have the same result, so formula (3) and (4) are equal, which prove the result $\tilde{a}_{\tilde{x}\tilde{y}} = \hat{a}_{\tilde{x}\tilde{y}}$.

• Now, let A be the **tensor sum** L: $A = \bigoplus_{i=1}^{N}{}_g \, Q^{(i)}$.

Let x and y be two global states. With the definition of generalized tensor sum, we have $a_{xy} = \sum_{i=1}^{N} q^{(i)}_{x^{(i)}y^{(i)}}(x)\Delta_{(x^{(i)},y^{(i)})}$, where[2] $\Delta_{(x^{(i)},y^{(i)})} = \prod_{j=1..N, j\neq i}\delta_{x^{(j)}y^{(j)}}$.

So $\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{\sigma\in\mathcal{P}} a_{x\sigma(y)} = \sum_{\sigma\in\mathcal{P}}\sum_{i=1}^{N} q^{(i)}_{x^{(i)}\sigma(y^{(i)})}(x)\Delta_{(x^{(i)},\sigma(y^{(i)}))}$

Notice that the only $\sigma = (\sigma_1 \ ... \ \sigma_K)$ giving a nonzero result for $\Delta_{(x^{(i)},\sigma(y^{(i)}))}$ are such that only one of the σ_h, $h = 1..K$, is not the identity (denoted by id).

Let $\bar{\sigma}_h = (id \ ... \ \sigma_h \ ...id)$ be such a permutation. Then we have:

$$\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{h=1}^{K}\left(\sum_{\sigma_h\in\mathcal{P}_h}\sum_{i=1}^{N} q^{(i)}_{x^{(i)}\bar{\sigma}_h(y^{(i)})}(x)\Delta_{(x^{(i)},\bar{\sigma}_h(y^{(i)}))}\right)$$

$\Delta_{(x^{(i)},\bar{\sigma}_h(y^{(i)}))}$ can be non null only for $i \in SI_h$, so finally,

$$\tilde{a}_{\tilde{x}\tilde{y}} = \sum_{h=1}^{K}\left(\sum_{\sigma_h\in\mathcal{P}_h}\sum_{i\in SI_h} q^{(i)}_{x^{(i)}\bar{\sigma}_h(y^{(i)})}(x)\Delta_{(x^{(i)},\bar{\sigma}_h(y^{(i)}))}\right) \tag{5}$$

Then let us define the lumped matrix $\hat{A}$.

Definition 8 - *Let us consider a SAN with N automata and a partition $\mathcal{G}$ (Definition 2). Then, let A be the tensor sum of the generator of the SAN, $A = \bigoplus_{i=1}^{N}{}_g \, Q^{(i)}$.*

The **lumped matrix** *$\hat{A}$ is defined as a tensor sum of K matrices:*

$$\hat{A} = \bigoplus_{h\in[1..K]}{}_g \, \hat{A}^h,$$

where $\hat{A}^h$, $h \in [1..K]$, is defined by

$$\hat{a}^h_{\tilde{x}_h\tilde{y}_h}(\tilde{x}) = \sum_{\sigma_h\in\mathcal{P}_h}\sum_{i\in SI_h} q^{(i)}_{x^{(i)}\sigma_h(y^{(i)})}(\tilde{x})\Delta^h_{(x^{(i)},\sigma_h(y_{(i)}))}$$

and $\Delta^h_{(x^{(i)},y^{(i)})} = \prod_{j\in SI_h, j\neq i}\delta_{x^{(j)}y^{(j)}}$.

Then we have

$$\hat{a}_{\tilde{x}\tilde{y}} = \sum_{h=1}^{K}\hat{a}^h_{\tilde{x}_h\tilde{y}_h}(\tilde{x}) = \sum_{h=1}^{K}\left(\sum_{\sigma_h\in\mathcal{P}_h}\sum_{i\in SI_h} q^{(i)}_{x^{(i)}\sigma_h(y^{(i)})}(\tilde{x})\Delta^h_{(x^{(i)},\sigma_h(y_{(i)}))}\right)\prod_{k\in[1..K],k\neq h}\delta_{x_k y_k} \tag{6}$$

In equation (6), the product $\prod_{k\in[1..K],k\neq h}\delta_{x_k y_k}$ multiplied by the $\Delta^h_{(x^{(i)},\sigma_h(y_{(i)}))}$ is equivalent to the $\Delta_{(x^{(i)},\bar{\sigma}_h(y^{(i)}))}$ defined for equation (5).

The two formula are therefore identical, and $\tilde{a}_{\tilde{x}\tilde{y}} = \hat{a}_{\tilde{x}\tilde{y}}$.

Tensor expression of $\tilde{Q}$

We have proved that $\tilde{Q} = \hat{L} + \sum_{e\in\mathcal{ES}}(\hat{P}_e + \hat{N}_e)$, and with the expression of the different lumped matrix, we have

[2] $\delta_{uv} = 1$ if $u = v$, and 0 otherwise

$$\tilde{Q} = \bigoplus_{g \atop h\in[1..K]} \hat{L}^h + \sum_{e\in\mathcal{ES}} \left(\bigotimes_{g \atop h\in[1..K]} \hat{P_e}^h + \bigotimes_{g \atop h\in[1..K]} \hat{N_e}^h \right)$$

and all the matrices $\hat{A}^h$ have been defined above (Definitions 7 and 8).

Thus, we obtain a compact representation of the aggregated Markov chain, as if each subset has been aggregated in an independent way.

5 The benefits of aggregation

We have seen that we can strongly aggregate a SAN with multiple replicas, but we still need to see what benefits it brings in order to justify this aggregation. We present at first some theoretical results, then we show the benefits of aggregation on practical examples.

5.1 Theoretical results

Consider a **SAN composed of one replica** with N automata, and let M be the number of states per automaton (it is the same for all automata because of the replication). The total number of states of the SAN is then M^N. We wish to calculate the number of states of the reduced Markov chain. Notice that we do not take unreachable states into account, and some states of the aggregated Markov chain can be removed (as seen in the examples).

For a given global state of the initial SAN, let us denote by na_m ($m \in [1..M]$) the number of automata that are in state m. The number of states of the aggregated Markov chain is therefore bounded by the number of integer solutions of the equation $\sum_{m=1}^{M} na_m = N$, which is [18]

$$\binom{N+M-1}{M-1} = \frac{(N+M-1)!}{N!(M-1)!}$$

Aggregation is beneficial only when there are several automata. For example, if we aggregate a SAN with $N = 2$ and $M = 5$, we have initially $5^2 = 25$ states and we obtain 15 states after aggregation. For large values of N, it becomes much more interesting. For example, when $N = 100$ and $M = 5$, we obtain an order of 4.6 million states after aggregation, instead of the $5^{100} \approx 10^{70}$ initial states. If we have N automata with 2 states each, we have initially 2^N states, and only $N + 1$ after aggregation.

Aggregation is useful for large SAN models, and it can be observed that in these cases, the state space reduction is significant. We have drawn some curves that show the state space reduction: percentage of the aggregated product state space size compared to the original product state space size, function of N, for different values of M (Figure 5).

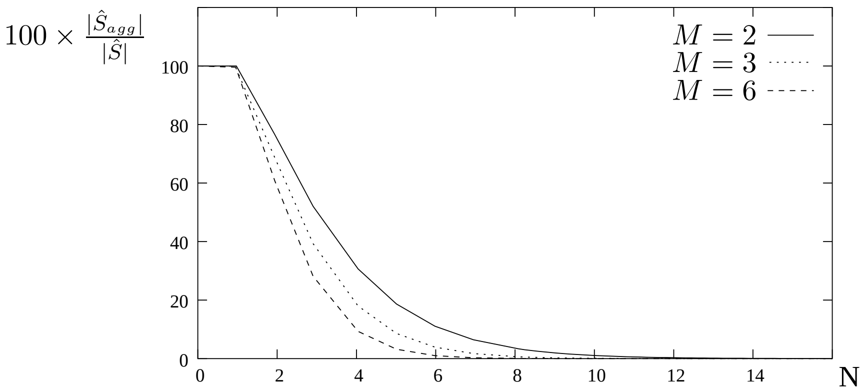

Figure 5: State space reduction

For $N > 10$, the size of the aggregated state space is negligible compared to the size of the original product state space, and the reduction is more significant for larger values of M.

For a **SAN with multiple replicas**, we have for each subset $h \in [1..K]$ $\frac{(R_h+M_h-1)!}{R_h!(M_h-1)!}$ aggregated states (M_h is the number of states of the automata of the subset h, and R_h the number of replicas), what makes a total of

$\prod_{h=1}^{K} \frac{(R_h+M_h-1)!}{R_h!(M_h-1)!}$ states. For example, if we have 2 subsets of R automaton, each having 2 states, and K other subsets containing only one automaton with M states each, we have initially $2^R \times 2^R \times M^K$ states, and only $(R+1) \times (R+1) \times M^K$ states after aggregation.

5.2 Numerical results

This section aims at showing the practical benefits of aggregation through a few examples. At first, we introduce some examples, and then we summarize the results.

RS1 and RS2

These models have already been described in Sections 3.2 and 3.4.

Figure 6 shows the aggregated Markov chain for the *RS1* model[3]. State i ($i = 1..N_p$) corresponds to the state *i processes are using a resource.* The part with dotted lines correspond to the unreachable states, it can be suppressed from the Markov chain.

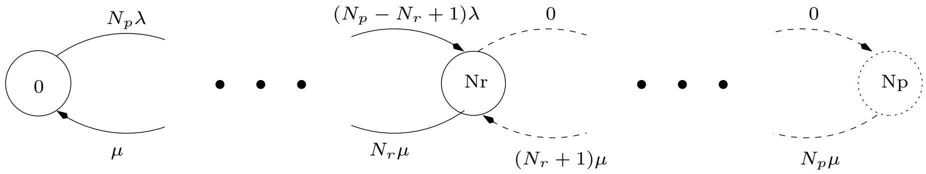

Figure 6: RS1 – aggregated Markov chain

The size of the SAN before aggregation is $|\hat{S}| = 2^{N_p}$ and $|S| = \sum_{i=0}^{N_r} \binom{N_p}{i}$.
The size of the aggregated Markov chain is $|\hat{S}_{agg}| = N_p + 1$, and when we suppress the unreachable states, $|S_{agg}| = N_r + 1$.

For the *RS2*, the aggregated state $(i_1, ..., i_K)$ means that i_h processes of the group h ($h = 1..K$) are using a resource. Some numerical results will be provided, showing the reduction of the state space in some special cases.

Resource sharing model with failure – RS3

This model is similar to *RS1*, except that the system may fail. Each of the N_p processes has an additional state *fail*, and can go to this state via a synchronizing transition from both states *sleeping* and *using*. The event evt_{fail} (rate λ_f) corresponds to a failure of the system. The occurrence of the event evt_{rep} (rate λ_r) means that the system has been repaired, and then all the processes are back in state *sleeping*.
An additional automaton represents the state of the system, it can be *failure* or *active*. Transitions from one state to another occur with evt_{fail} and evt_{rep}. The other events are the same as those of *RS1* (Section 3.2). This model, which we shall call *RS3*, is graphically illustrated in Figure 7. It is a SAN with two replicas, the first one consisting in automata $\mathcal{A}^{(i)}$, $i = 1..N_p$ and the second one consisting only in automaton $\mathcal{A}^{(N_p+1)}$.

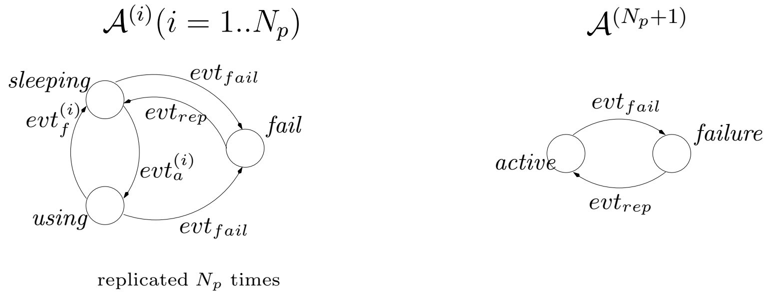

Figure 7: Resource Sharing Model with failure – RS3

Figure 8 shows the aggregated Markov chain (without the unreachable states). State i ($i = 1..N_r$) corresponds to the state *i processes are using a resource*, and state *fail* corresponds to the failure of the system.

[3]In fact, Figure 6 is a generalization of the model in Figure 4.

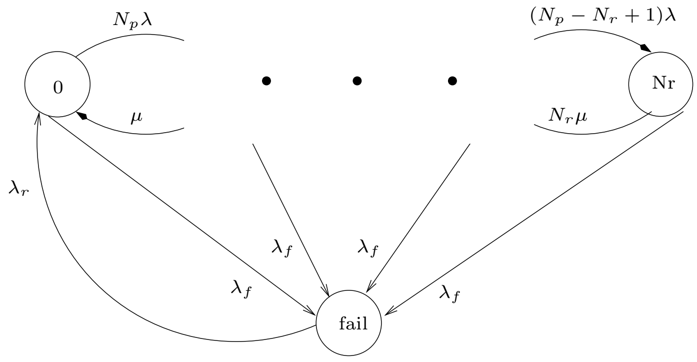

Figure 8: RS3 – aggregated Markov chain

The size of the SAN before aggregation is $|\hat{S}| = 2 * 3^{N_p}$ and $|S| = 1 + \sum_{i=0}^{N_r} \binom{N_p}{i}$ (one more state than for *RS1*, corresponding to the global state *the system is in failure*).
The size of the aggregated Markov chain is $|\hat{S}_{agg}| = (N_p + 2)(N_p + 1)$, and when we suppress the unreachable states, $|S_{agg}| = N_r + 2$.

On/Off Sources model – OS

This fourth model presents a case in which s on/off sources feed a limited capacity (C requests) queue. The SAN for this model is represented by $s + 1$ automata. Each source is represented by a two-states automaton (local states *on* and *off*), and an additional automaton with $C + 1$ states. The arrival rate of the queue is a function of the state of the sources automata, and the service rate is a constant value μ (rate of the event evt_s). We consider two groups of sources, the first with s_1 sources (automata 1 to s_1), and rate λ_1, and the second with $s_2 = s - s_1$ sources (automata $s_1 + 1$ to s), and rate λ_2. These source rates shall be added to the arrival rate when the source is in the state *on*. Then the rate of the event of arrival in the queue evt_t is

$$\lambda_1 \sum_{i=1}^{s_1} \delta\left(st(\mathcal{A}^{(i)} == on)\right) + \lambda_2 \sum_{i=s_1+1}^{s} \delta\left(st(\mathcal{A}^{(i)} == on)\right)$$

Figure 9 illustrates this SAN model. The events $evt_a^{(i)}$ and $evt_f^{(i)}$ have constant rates, depending only of the type of source (group 1 or group 2).

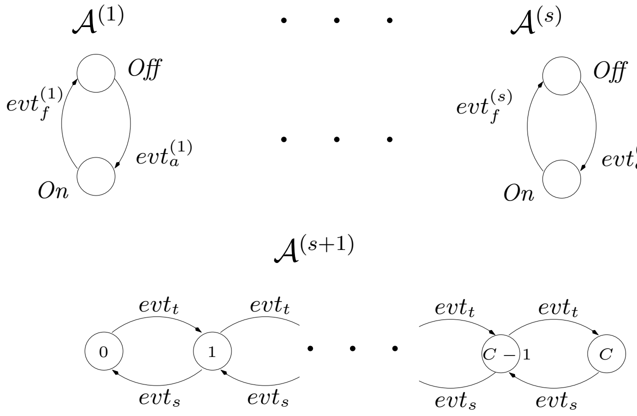

Figure 9: On/Off Sources model – OS

This model has three subsets ($K = 3$), and the first and second subsets can be aggregated. The first subset aggregation results in an automaton with $s_1 + 1$ states, and the second subset aggregation results in an automaton with $s_2 + 1$ states. The reduced state space is then $|\hat{S}_{agg}| = (s_1 + 1) \times (s_2 + 1) \times (C + 1)$, instead of the original state space $|\hat{S}| = 2^s \times (C + 1)$.

Cluster with Bus Communication – CB

This fifth model represents a cluster with N_p processing nodes with a simple behavior:

- each node i alternates from *idle* ($Id^{(i)}$) to *processing* ($Pr^{(i)}$) state;
- being in the *idle* state a node can pass to *transmission* ($Tx^{(i)}$) state if no other node is already transmitting; or it can pass to *reception* ($Rx^{(i)}$) state if there is another node transmitting;
- after transmitting or receiving each node returns to *idle* state regardless of the state of the other nodes[4].

Such model could be seen as a simple description of UDP protocol over a shared bus, or other communication protocol in which there is no need of commitment between send/receive connections. Despite the application of such model, our interest in this paper is the description of a SAN with N_p automata with four states each. This replicated automata model is represented in Figure 10. All events in this SAN are local, but events $l_3^{(i)}$ and $l_5^{(i)}$ must be defined with functional rates (the other events have constant rates), respectively:

- $\left(\sum_{i=1}^{N_p} \delta(st(\mathcal{A}^{(i)} == Tx^{(i)}))\right) = 1$ in order to grant access to reception state if there is one node transmitting; and
- $\left(\sum_{i=1}^{N_p} \delta(st(\mathcal{A}^{(i)} == Tx^{(i)}))\right) = 0$ in order to grant access to transmission state if no node is already transmitting.

When all nodes have the same rates this model is a SAN composed of one replica. Only the global states in which only one node, at most, is transmitting are reachable, *i.e.*:

$$reachability = \left(\sum_{i=1}^{N_p} \delta(st(\mathcal{A}^{(i)} == Tx^{(i)}))\right) \leq 1$$

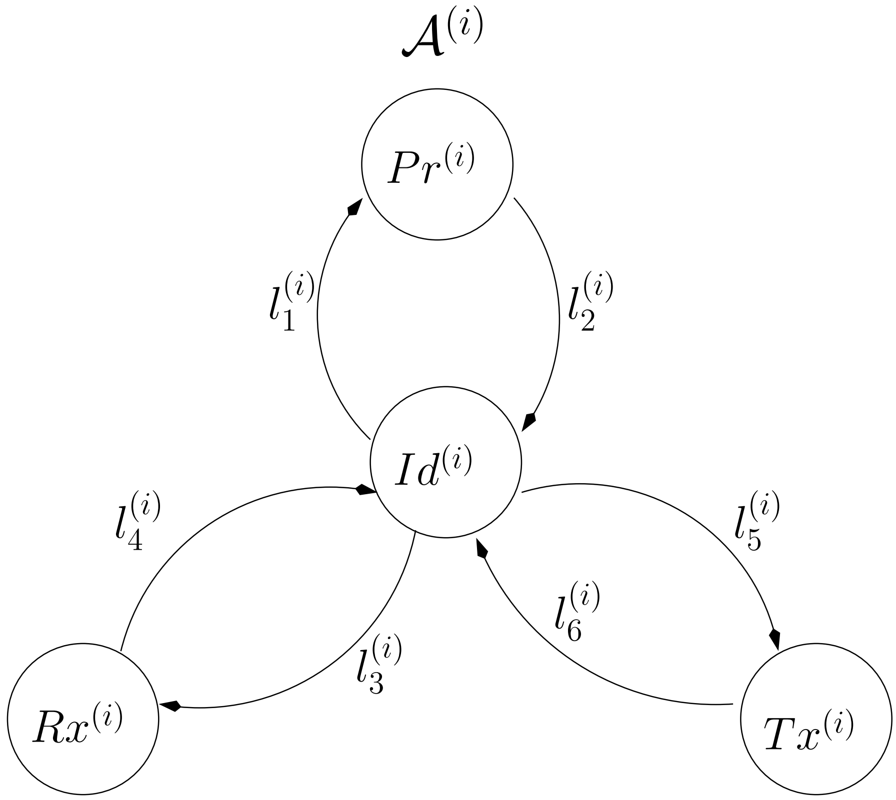

Figure 10: Cluster with Bus Communication replicated automaton – CB

[4]According to such behavior it is not necessary to synchronize nor the beginning, neither the end of transmissions.

Numerical Results

We present here some numerical results obtained with the software package PEPS [1] aggregating and solving the presented examples. The aggregation of replicas was automatically performed by PEPS using the model specification information (see Section 3.5), and the lumped SAN models were stored in the tensor format (see Section 4.3). The computer used to run the examples was an IBM-PC running Linux OS (Mandrake distribution, version 8.0), with 1.5 Gbytes of RAM and with 2.0 GHz Pentium IV processor. The indicated processing times do not take system time into account, *i.e.*, they refer only to the user time spent to perform one iteration[5], in order to compute the steady-state probabilities[6]. A time of 0 *sec.* means that the time is negligible (less than 10^{-4} *sec.*). An indication $-$ means that PEPS was unable to solve the model (too many states). It is important to notice that only the processing times to compute the solution were indicated, since it is not necessary to disaggregate the solution vector on to the unaggregated state space. In fact, the computation of the result functions performed by PEPS (integration over the probability vector) works over the lumped state space.

Table 1: Numerical results

Model	initial SAN			aggregated Markov chain		
	$\|\hat{S}\|$	$\|S\|$	Time (sec.)	$\|\hat{S}_{agg}\|$	$\|S_{agg}\|$	Time (sec.)
RS1 $N_p = 20,\ N_r = 10$	$1,048,576$	$616,666$	15.7	21	11	0.000
RS1 $N_p = 10,000,\ N_r = 8,000$	$2^{10,000}$	-	-	$10,001$	$8,001$	0.004
RS2 $K = 2,\ N_r = 10$ $R_1 = 5,\ R_2 = 15$	$1,048,576$	$616,666$	15.7	96	51	0.000
RS3 $N_p = 20,\ N_r = 10$	$6,973,568,802$	$616,667$	-	462	12	0.000
OS $s_1 = 12,\ s_2 = 4,\ K_q = 1,000$	$65,601,536$	$65,601,536$	101.5	$65,065$	$65,065$	0.032
OS $s_1 = 8,\ s_2 = 8,\ K_q = 1,000$	$65,601,536$	$65,601,536$	101.5	$81,081$	$81,081$	0.034
CB $N_p = 12$	$16,777,216$	$2,657,205$	72.6	455	169	0.000
CB $N_p = 15$	$1,073,741,824$	$86,093,442$	-	816	256	0.000

The results in Table 1 show that aggregation always produce a significant state space reduction. This is true even when the original product state space is equal to the original reachable state space (model *OS*). The cardinality of the subsets, as in the two *OS* models presented, has little effect when compared to the gains achieved by the aggregation technique. The time required to compute performance indexes for this model becomes quite negligible. Taking replicas into account can even make possible to solve problems that were too large before, as the proposed *RS3* model, the *RS1* model with $N_p = 10,000$, and the *CB* model with $N_p = 15$. In fact, the last *OS* with a little bit more than 65 million states is the largest model directly solved by PEPS until now. Larger models can be solved, but with the current computational power, and specially the current memory limitations, there is no much room to handle larger models, unless some aggregation technique, like ours, is employed.

6 Conclusions

In this paper, we have defined SAN models with replicas, and have exposed a technique to aggregate such models. We proved a theorem showing how strong aggregation can be performed, and how the matrix of the aggregated

[5]The number of iterations may vary according to the chosen numerical input parameters. However, the time to perform one iteration is not affected by the choice of numerical parameters. The solution method used to compute results was power iteration, but the other methods available in PEPS (Arnoldi and GMRES) have a quite similar time costs per iteration that depends on the cost of one vector-description multiplication [12].

[6]The technique presented in this paper is probably also useful for transient analysis. However, our interest here is limited to stationary solution, since this is the only solution formally defined for SAN models and implemented in the PEPS software tool.

Markov chain can be expressed as a tensor expression. The theoretical benefits of such an aggregation are also presented and verified by the numerical achievements of the implementation in the PEPS software tool.

It can be shown that a SAN with replicas is also exactly lumpable over the same partition [2]. The proof is very similar to the proof of Lemma 1 (Section 4). This property (exactly lumpable) allows to compute the probabilities of the initial model knowing the probabilities of the aggregated system.

Notice that the subsets of replicated automata have to be defined at the high level specification. This is often a fake problem since the replicas are usually known when modeling a particular system.

The implementation of the automatic generation of the aggregated Markov Chain in PEPS proved the efficiency of the proposed technique. The implemented algorithm generates the tensor expression of the matrix very quickly (much faster than the time spent by one single iteration). The resulting tensor expression is usually much more compact, so we can compute the numerical solution with an impressive reduction of CPU costs.

In this paper, we work only on the matrices describing such SAN to prove that the matrix of the aggregated Markov chain can be expressed as a tensor expression. We could as future work define formally an equivalent SAN resulting from the aggregation of each subset.

Finally, we plan to work on applications, *e.g.*, communication protocols, to experiment those new techniques on real and large systems with a significant number of replicas. Many models in communication could not be solved by SAN due to the limitation of the product state space size, but we do believe that the proposed technique can boost the use of SAN to such practical cases.

The authors wish to **thank** Cyril Guilloud for his preliminary work on SAN aggregation.

References

[1] A. Benoit, L. Brenner, P. Fernandes, B. Plateau and W. J. Stewart. The PEPS software tool. In P. Kemper, W. H. Sanders (eds.) *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, LNCS*, Urbana-Champaign, Illinois, USA, 2003.

[2] P. Buchholz. Exact and ordinary lumpability in finite Markov chains. *Journal of Applied Probability*, 31:59–75, 1994.

[3] P. Buchholz. Equivalence relations for stochastic automata networks. In *Proc. 2nd int. workshop on the numerical solution of Markov chains, Kluwer ed. Boston, Masachussetts*, 1995.

[4] P. Buchholz. An aggregation-disaggregation algorithm for stochastic automata networks. *Probability in the Engineering and Informational Sciences*, 11:229–253, 1997.

[5] P. Buchholz. An adaptative aggregation-disaggregation algorithm for hierarchical Markovian models. *European Journal of Operational Research*, 116:545–564, 1999.

[6] P. Buchholz. Exact Performance Equivalence - An Equivalence relation for stochastic automata. *Theoretical Computer Science 215(1/2)*, pages 239–261, 1999.

[7] J. Campos, M. Silva and S. Donatelli. Structured solution of stochastic DSSP systems. In *Proc. 7th Int. Workshop on Petri Nets and Performance Models (PNPM'97), Saint Malo, France, IEEE Comp. Soc. Press*, pages 91–100, June 1997.

[8] S. Donatelli. Superposed Stochastic Automata: a class of stochastic Petri nets with parallel solution and distributed state space. *Performance Evaluation*, 18(1):21–36, 1993.

[9] S. Donatelli. Superposed Generalized Stochastic Petri nets: definition and efficient solution. In R. Valette (ed.) *15th International Conference on Application and Theory of Petri Nets, LNCS 815*, Zaragoza, Spain, pages 258–277, 1994.

[10] P. Fernandes. *Mthodes Numriques pour la Solution de Systmes Markoviens Grand Espace d'États.* PhD thesis, Institut National Polytechnique de Grenoble, France, 1998.

[11] P. Fernandes, B. Plateau and W.J. Stewart. Efficient descriptor-vector multiplications in stochastic automata networks. *JACM*, 45(3):381–414, 1998.

[12] P. Fernandes, B. Plateau and W.J. Stewart. Optimizing tensor product computations in stochastic automata networks. *RAIRO, Operations Research*, 32(3):325–351, 1998.

[13] O. Gusak, T. Dayar and J-M. Fourneau. Stochastic automata networks and near complete decomposability. *SIAM Journal on Matrix analysis and Applications*, 23:581–599, 2001.

[14] O. Gusak, T. Dayar and J-M. Fourneau. Lumpable continuous-time stochastic automata networks. *European Journal of Operational Research*, 148:436–451, 2003.

[15] O. Gusak, T. Dayar and J-M. Fourneau. Iterative disaggregation for a class of lumpable discrete-time stochastic automata networks. *Performance Evaluation*, 53:43–69, 2003.

[16] H. Hermanns and M. Ribaudo. Exploiting symmetries in stochastic process algebras. In *Proc. of 12th European Simulation Multiconference*, Manchester (UK), 1998.

[17] J.G. Kemeny and J.L. Snell. *Finite Markov Chains*. Van Nostrand Reinhold, New York, 1960

[18] D.E. Knuth. *The art of computer programming, Volume 1 (Fundamental Algorithms).* Addison-Wesley, 1967.

[19] J. Ledoux. Weak lumpability of finite Markov chains and positive invariance of cones. *Technical report*, Rapport de recherche INRIA n. 2801, France, 1996.

[20] A.S. Miner, G. Ciardo and S. Donatelli. Using the exact state space of a Markov model to compute approximate stationary measures. In *Measurement and Modeling of Computer Systems*, pages 207–216, 2000.

[21] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In *Proc. ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems*, Austin, Texas, Aug 1985.

[22] M. Rosenblatt. Functions of a Markov process that are Markovian. *Journal of Math. and Mech.*, 8(4): 585–596, 1959.

[23] W. H. Sanders and J. F. Meyer. Reduced base model construction methods for stochastic activity networks. *IEEE Journal on Selected Areas in Communications*, 9(1):25–36, 1991.

[24] M. Siegle. Structured Markovian performance modelling with automatic symmetry exploitation. In *Proc. 11th Conf. Computer performance Evaluation : Modelling Techniques and Tools, LNCS 794*, 1994.

[25] M. Siegle. Using structured modelling for efficient performance prediction of parallel systems. In *Parallel Computing: Trends and Applications (G.R.Joubert et al., eds.)*, pages 453–460, NorthHolland, 1994.

[26] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press: Princeton, 1994.

APPENDIX: proof of Lemma 1

To prove Lemma 1, we decompose the problem into two parts, corresponding respectively to the local part of the descriptor (Lemma 2) and the synchronizing part (Lemma 3).

> **Lemma 2:** *For all $\beta, \gamma \in [1..\Gamma]$, the probability of passing from a state $x \in \Omega_\beta$ to Ω_γ with a local transition always has the same value for each state x in Ω_β: $\forall x \in \Omega_\beta \;\; l_{x\Omega_\gamma} = l_{r\beta\Omega_\gamma}$.*

• PROOF OF LEMMA 2

Let $\beta, \gamma \in [1..\Gamma]$, $x \in \Omega_\beta$ and let $\tau \in \mathcal{P}$ be the permutation such that $x = \tau(r\beta)$.
With the definition of a generalized tensor sum ([10, 26]), for $y \in \Omega_\gamma$, we have
$l_{xy} = \sum_{i=1}^{N} \left[q^{(i)}_{l\; x^{(i)} y^{(i)}}(x) \prod_{j=1, j\neq i}^{N} \delta_{x^{(j)} y^{(j)}} \right]$, where $Q^{(i)}_l$ is the local transition matrix of automaton $\mathcal{A}^{(i)}$, and $\delta_{uv} = 1$ if $u = v$, and 0 otherwise.

In order to simplify the notation, let us define $\Delta_{x^{(i)} y^{(i)}} = \prod_{j=1, j\neq i}^{N} \delta_{x^{(j)} y^{(j)}}$.

Then we have

$$l_{x\Omega_\gamma} = \sum_{y \in \Omega_\gamma} l_{xy} = \sum_{\sigma \in \mathcal{P}} l_{x\sigma(r\gamma)} = \sum_{\sigma \in \mathcal{P}} \sum_{i=1}^{N} q^{(i)}_{l\; x^{(i)} r\gamma^{(\sigma(i))}}(x) \;\; \Delta_{x^{(i)} r\gamma^{(\sigma(i))}}$$

Moreover, we have $x = \tau(r\beta)$, so

$$l_{x\Omega_\gamma} = \sum_{\sigma \in \mathcal{P}} \sum_{i=1}^{N} q^{(i)}_{l\; r\beta^{(\tau(i))}\; r\gamma^{(\sigma(i))}}(\tau(r\beta)) \;\; \Delta_{r\beta^{(\tau(i))}\; r\gamma^{(\sigma(i))}}$$

Now decompose the equation for each subset of the SAN. Recall that the matrices of a subset $h \in [1..K]$ are all identical to $Q_{(l,h)}$. Moreover, for $\sigma \in \mathcal{P}$, let $\varphi \in \mathcal{P}$ be the permutation $\varphi = \sigma \circ \tau^{-1}$. Then $\sigma = \varphi \circ \tau$ and when σ goes through $\mathcal{P}$, φ does the same (in a different order). Because of the commutativity of the sum, we can replace $\sum_{\sigma \in \mathcal{P}}$ with $\sum_{\varphi \in \mathcal{P}}$. Then we have:

$$l_{x\Omega_\gamma} = \sum_{\varphi \in \mathcal{P}} \sum_{h=1}^{K} \sum_{i \in SI_h} q_{(l,h)\; r\beta^{(\tau(i))}\; r\gamma^{(\varphi\circ\tau(i))}}(\tau(r\beta)) \;\; \Delta_{r\beta^{(\tau(i))}\; r\gamma^{(\varphi\circ\tau(i))}}$$

The functions are not changed by a permutation of $\mathcal{P}$, and $\tau \in \mathcal{P}$, so we can replace $\tau(r\beta)$ by $r\beta$ in the above equation. Moreover, for each subset h, when i goes through SI_h, $\tau(i)$ does the same but in a different order (it is a permutation inside the subset). Because of the commutativity of a sum, we can change the order and replace $\tau(i)$ by i in the equation:

$$l_{x\Omega_\gamma} = \sum_{\varphi \in \mathcal{P}} \sum_{h=1}^{K} \sum_{i \in SI_h} q_{(l,h)\; r\beta^{(i)}\; r\gamma^{(\varphi(i))}}(r\beta) \;\; \Delta_{r\beta^{(i)}\; r\gamma^{(\varphi(i))}} = \sum_{\varphi \in \mathcal{P}} \sum_{i=1}^{N} q^{(i)}_{l\; r\beta^{(i)}\; r\gamma^{(\varphi(i))}}(r\beta) \;\; \Delta_{r\beta^{(i)}\; r\gamma^{(\varphi(i))}}$$

Finally, $l_{x\Omega_\gamma} = \sum_{\varphi \in \mathcal{P}} l_{r\beta\; \varphi(r\gamma)} = l_{r\beta\; \Omega_\gamma}$.

□

Lemma 3: *For all $\beta, \gamma \in [1..\Gamma]$, the probability of passing from a state $x \in \Omega_\beta$ to Ω_γ with a synchronizing transition always has the same value for each state x in Ω_β:* $\forall x \in \Omega_\beta \quad \sum_{e\in\mathcal{ES}} \left(pe_{x\Omega_\gamma} + ne_{x\Omega_\gamma}\right) = \sum_{e\in\mathcal{ES}} \left(pe_{r\beta\Omega_\gamma} + ne_{r\beta\Omega_\gamma}\right)$.

• Proof of Lemma 3

Let $e \in \mathcal{ES}$, $\beta, \gamma \in [1..\Gamma]$, $x \in \Omega_\beta$ and let $\tau \in \mathcal{P}$ be the permutation such that $x = \tau(r\beta)$. We prove first that $pe_{x\Omega_\gamma} = pe_{r\beta\Omega_\gamma}$. The proof for ne is similar.

With the definition of a generalized tensor product ([10, 26]), for $y \in \Omega_\gamma$, we have $pe_{xy} = \prod_{i=1}^{N} q^{(i)}_{e+\ x^{(i)} y^{(i)}}(x)$, where $Q^{(i)}_{e+}$ is the positive synchronization transition matrix of automaton $\mathcal{A}^{(i)}$.

Moreover, $x = \tau(r\beta)$, so we have

$$pe_{x\Omega_\gamma} = \sum_{y\in\Omega_\gamma} pe_{xy} = \sum_{\sigma\in\mathcal{P}} pe_{x\sigma(r\gamma)} = \sum_{\sigma\in\mathcal{P}} \prod_{i=1}^{N} q^{(i)}_{e+\ r\beta^{(\tau(i))}\ r\gamma^{(\sigma(i))}}(\tau(r\beta))$$

Now decompose the equation for each subset of the SAN. Recall that almost all the matrices are identical inside a subset $g \in [1..K]$; we denote them by $Q_{(e+,g)}$. There is only one particular matrix in one of the subsets equal to $f_e Q_{(e+,g)}$, where f_e is the transition rate of the event (definition of SAN models with replicas).

For $\sigma \in \mathcal{P}$, let $\varphi \in \mathcal{P}$ be the permutation $\varphi = \sigma \circ \tau^{-1}$. Then $\sigma = \varphi \circ \tau$ and when σ goes through $\mathcal{P}$, φ does the same (in a different order). Because of the commutativity of the sum, we can replace $\sum_{\sigma\in\mathcal{P}}$ with $\sum_{\varphi\in\mathcal{P}}$. Then we have:

$$pe_{x\Omega_\gamma} = \sum_{\varphi\in\mathcal{P}} \prod_{h=1}^{K} \left(f_e(\tau(r\beta)) \prod_{i\in SI_h} q_{(e+,h)\ r\beta^{(\tau(i))}\ r\gamma^{(\varphi\circ\tau(i))}}(\tau(r\beta)) \right)$$

The functions are not changed by a permutation of $\mathcal{P}$, and $\tau \in \mathcal{P}$, so we can replace $\tau(r\beta)$ by $r\beta$ in the above equation. Moreover, for each subset h, when i goes through SI_h, $\tau(i)$ does the same but in a different order (it is a permutation inside the subset). Because of the commutativity of a product, we can change the order and replace $\tau(i)$ by i in the equation:

$$pe_{x\Omega_\gamma} = \sum_{\varphi\in\mathcal{P}} \prod_{h=1}^{K} \left(f_e(r\beta) \prod_{i\in SI_h} q_{(e+,h)\ r\beta^{(i)}\ r\gamma^{(\varphi(i))}}(r\beta) \right) = \sum_{\varphi\in\mathcal{P}} \prod_{i=1}^{N} q^{(i)}_{e+\ r\beta^{(i)}\ r\gamma^{(\varphi(i))}}(r\beta)$$

Finally, $pe_{x\Omega_\gamma} = \sum_{\varphi\in\mathcal{P}} pe_{r\beta\ \varphi(r\gamma)} = pe_{r\beta\ \Omega_\gamma}$.

In a similar way, we can prove that $ne_{x\Omega_\gamma} = ne_{r\beta\ \Omega_\gamma}$. This is true for all events $e \in \mathcal{ES}$, so we finally have $\sum_{e\in\mathcal{ES}} \left(pe_{x\Omega_\gamma} + ne_{x\Omega_\gamma}\right) = \sum_{e\in\mathcal{ES}} \left(pe_{r\beta\Omega_\gamma} + ne_{r\beta\Omega_\gamma}\right)$

$\square$

• Proof of Lemma 1

Let $\beta, \gamma \in [1..\Gamma]$, $x \in \Omega_\beta$ and let $\tau \in \mathcal{P}$ be the permutation such that $x = \tau(r\beta)$.
With the decomposition of Q, we have

$q_{x\Omega_\gamma} = \sum_{y\in\Omega_\gamma} q_{xy} = \sum_{y\in\Omega_\gamma} \left(l_{xy} + \sum_{e\in\mathcal{ES}}(pe_{xy} + ne_{xy})\right)$
$= \sum_{y\in\Omega_\gamma} l_{xy} + \sum_{e\in\mathcal{ES}} \left(\sum_{y\in\Omega_\gamma} pe_{xy} + \sum_{y\in\Omega_\gamma} ne_{xy}\right) = l_{x\Omega_\gamma} + \sum_{e\in\mathcal{ES}} \left(pe_{x\Omega_\gamma} + ne_{x\Omega_\gamma}\right)$

With the application of Lemma 2 and Lemma 3, we finally have
$q_{x\Omega_\gamma} = l_{r\beta\Omega_\gamma} + \sum_{e\in\mathcal{ES}} \left(pe_{r\beta\Omega_\gamma} + ne_{r\beta\Omega_\gamma}\right) = q_{r\beta\Omega_\gamma}$.

$\square$