A General Result for Deriving Product-Form Solutions in Markovian Models

Andrea Marin
Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino, 155
Venice, Italy
marin@dsi.unive.it

Maria Grazia Vigliotti*
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ, UK
mgv98@doc.ic.ac.uk

ABSTRACT

In this paper we provide a general method to derive product-form solutions for stochastic models. We take inspiration from the Reversed Compound Agent Theorem [14] and we provide a different formulation using labeled automata, a generalization which encompasses a bigger class of product-form solutions, and a new proof based on the solution of the system of global balance equations. We show that our result may have practical applications in the performance evaluation of complex software and hardware architectures and can be the base for the development of new analysis tools or the extension of existing ones.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Modeling techniques

General Terms

Performance

Keywords

Queuing theory, Product form solutions.

1. INTRODUCTION

Performance engineering is about development of efficient computer and communication systems by providing a crucial performance analysis during the design phase. In essence, this requires the definition of a model and the analysis of relevant properties. Creating a model for analysis is quite common to both software engineering and performance analysis communities. The kind of model and the analysis distinguish the two communities.

*Maria G. Vigliotti is supported by the EPSRC grant SPARTACOS (EP /D047587/1)

WOSP/SIPEW'10, January 28–30, 2010, San Jose, California, USA.

It has been established [29] that performance evaluation should be introduced at the early stages of development of software to avoid disproportionate costs of redesign and reimplementation of software. This means finding a suitable modeling formalism that allows both software development and performance analysis. In this direction a lot of work has been done by annotating UML or by directly using process calculi in the design and validation of complex interactive systems [19]. Transition systems, which are labeled automata, are also used in software engineering for development of distributed systems [24]. An automaton is a state machine that formally establishes how a system moves from one state to another. Probabilistic and stochastic automata are also used as basis formalism for model checking [2], which is a fundamental technique in design and verification of complex systems.

In this paper we focus on developing a performance analysis technique that may be applied to a variety of different distributed systems. For this reason, we formulate our analysis in terms of labeled automata, or transition systems. The stochastic models that we study are those whose underlying processes are Continuous Time Markov Chains (CTMCs). In particular, we focus our attention on characterizing the steady-state probability distributions as product-form solutions. Product-form solutions express the steady-state probabilities of a model as the product of the marginal steady-state distributions of the components which the system is composed of. This is particularly useful in the analysis of systems with large state spaces, when computing the steady-state probabilities is quite expensive or unfeasible. Steady-state probabilities are the core of performance analysis as they are needed to obtain significant performance measures, such as the throughput or response time distribution [11, 18]. Response time distribution is crucial in guaranteeing the Quality of Service (QoS) since it determines the distribution of the time customers spend waiting for service. Product-form solutions are very useful in software engineering: in fact, in presence of product-form solutions, different metrics of performance that depend on the steady-state probabilities, can be calculated in a modular way starting from the components of the system [18]. For example in [5, 6] the authors maps a set of UML diagrams into product-form queueing networks, so that they can efficiently perform an exact analysis and derive the desired performance measures.

Product-form solutions have been extensively studied, e.g., [21, 27, 7, 25, 12, 8], for different systems in the theory of

Markovian stochastic models. The holy grail in this field consists in finding general conditions that characterize (most of) the systems that have product-form solution. In this direction a lot of work has been done using Generalized Stochastic Petri Nets (GSPNs) [10, 3, 4] and Performance Evaluation Process Algebra (PEPA) [13, 28, 20, 9]. In this paper we take a different approach [17] and we use a variant of labeled automata. Before presenting the novel results, we informally describe the type of synchronization we deal with in labeled automata. Roughly speaking, two automata synchronize in the sense that some transitions in one of them cause a transition in the other. The synchronized transitions cannot be done by an automaton in an autonomous way. We call active transitions those that govern the synchronization, and passive the others. Our starting point is the Reversed Compound Agent Theorem (RCAT)[14], yet we expand that work in several directions:

- We first notice that in the original formulation there were some limitations [14]. In the example of Section 4 we show that there exist models such that the original structural conditions of RCAT do not hold even if they are in product-form. The logical conclusion is that there exist more general conditions that characterize a larger class of systems in product-form. Specifically, one of the original conditions of RCAT imposes that in each possible state of the automata there exists only *one* synchronizing incoming active action. In this paper, we generalize this condition by considering a finite number of incoming active actions in each state of the components of the system.

- We provide a new, entirely different proof of the theorem (with respect to that formulated for RCAT) based on the global balance equations (GBEs) analysis. The proofs in the original papers [14, 16] and all their subsequents were based on the application of Kolmogorov's criteria. Finally, we think that it is worthwhile pointing out that the usage of the GBEs allows us to give a formulation that does not explicitly need the idea of reversed processes. The latter is not at all intuitive when working with models of real systems.

- We have decided to depart from the original formulation of the theorem in terms of PEPA in favor of the new formulation based on labeled automata. There are several motivations for this choice:

  1. Automata are widely used in software engineering for distributed system. We think that this should make the application of our results easier.
  2. The definition of automata retains the compositionality of process algebra, and allows for modular descriptions of systems.
  3. Since most of the results about product-form concerns stochastic models with an underlying CTMC, we think that using automata is appropriate for both comparing in the same formalism the well known results, and for being able to describe hybrid models, i.e. models that consist of components specified by different formalisms. In particular, the synchronization semantic that we use is suitable for specifying the interactions among queues in a queueing network, or among the transitions in stochastic Petri nets. The only limitation is that a synchronized transition must involve exactly two automata. Future extensions may deal with multiple synchronizations.

In the paper we provide a detailed comparison of our work both with RCAT and the conditions of the Markov implies Markov property ($M \Rightarrow M$). This property, introduced by Muntz in [27] plays a pivotal role in the characterization of the queueing disciplines which lead to a product-form like BCMP. We show that the generalization that we propose is not trivial, and indeed characterizes a larger class of product-form solutions than those identified both by the $M \Rightarrow M$ and RCAT. Informally, we can say that the main theorem that we present defines a unique framework which includes both RCAT and the $M \Rightarrow M$ product-form model classes.

Finally, we provide a practical example (see Figure 6) to show that our theoretical work has useful applications in the performance evaluation of distributed systems. Our example, taken from distributed service could not have been dealt by either RCAT or the $M \Rightarrow M$. Yet, the product-form solution follows from the application of Theorem 1 very straightforwardly. Briefly, our system consists of two databases, namely *DB1* and *DB2*. *DB1* serves the requests of *TYPEA*, while *DB2* serves the requests of *TYPEB*. The requests arrive to the databases through a communication line that has two channels. The databases require a sort of synchronisation and it can happen that a transaction in the first database causes a canceling of a transaction in the second database (or vice-versa). The two databases are modeled as G-queues [12], while the channel that receives the requests and sends back the answers is a Multiserver Station with Concurrent Class of Customers (MSCCC) as described in [25].

The rest of the paper is organized as follows: in Section 2 we provide the basic definitions for the Labeled Markov Automata (LMAs) and how the underlying CTMCs are defined; in Section 3 we discuss our theorem and we compare it with RCAT and with the $M \implies M$ property. We also provide a sketch of the proof of the theorem. In Section 4 we use the theoretical results in an example taken from distributed system analysis. We comment on the fact that none of the results before could deal with such an example. Conclusion of our work and suggestion for future work follow.

2. LABELED MARKOV AUTOMATA (LMA)

We assume the reader familiar with probability theory and the basics of CTMCs.

In this section we provide the basic definitions that will allow us to define the cooperation among CTMCs.

Definition 1 (Labeled automaton) *A Markov automaton is a tuple*

$$\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$$

such that:

1. S *is the denumerable set of states (*state space*) with* $s_1, s_2, \ldots s_n, \ldots$ *range over it,*
2. *Act is the set of* action labels *with* $a, b, \ldots$ *range over it,*

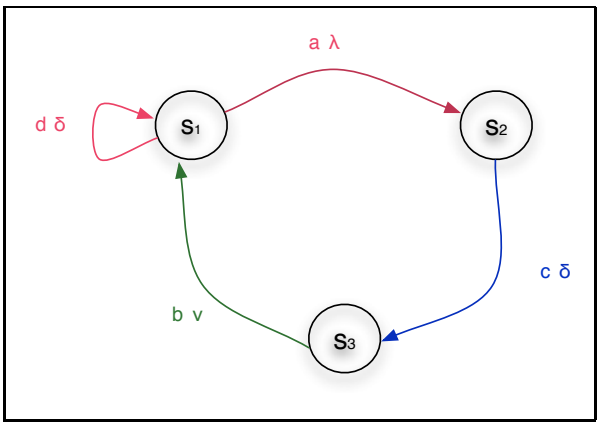

Figure 1: Graphical representation of the Markov automaton $\mathcal{M}_1$

3. $\rightarrow$ *is the transition relation between states defined as follows:*

$$\rightarrow : \mathsf{S} \times Act \times (\mathbb{R}^+ \cup \mathbf{Var}) \times \mathsf{S},$$

where $\mathbb{R}^+$ *is the set of positive real numbers and* **Var** *is the set of variable names such that if* $a \in Act$ *the* $x_a \in \mathbf{Var}$.

For readability, we write $(s_1, a, \lambda, s_1') \in \rightarrow$ *as* $s_1 \xrightarrow{a,\lambda} s_1'$. *We define two sets* $\mathcal{A}(\mathcal{M})$ *and* $\mathcal{P}(\mathcal{M})$, active actions *and* passive actions, *such that for every* $a \in Act$ *if* $s \xrightarrow{a,\lambda} s'$, *with* $\lambda \in \mathbb{R}^+$, *then* $a \in \mathcal{A}(\mathcal{M})$, *and if* $s \xrightarrow{a,x_a} s'$, *with* $x_a \in \mathbf{Var}$, *then* $a \in \mathcal{P}(\mathcal{M})$.

Initially, one can think a LMA as a CTMC were the transitions have been labeled. An easy way to understand the behavior of LMAs is to draw (when possible) the transitions as in the following example.

Example 1 *Let us consider the following labeled automaton that is depicted by Figure 1:*

$$\mathcal{M}_1 = \langle \mathsf{S}_1, Act_1, \rightarrow \rangle$$

where $S_1 = \{s_1, s_2, s_3\}$ *and* $Act_1 = \{a, b, c, d\}$ *and*

$$\rightarrow = \{(s_1, a, \lambda, s_2), (s_2, c, \delta, s_3), (s_3, b, \nu, s_2), (s_1, d, \delta, s_1)\}$$

A more readable notation would be: $s_1 \xrightarrow{a,\lambda} s_2$, $s_1 \xrightarrow{d,\delta} s_1$, $s_2 \xrightarrow{d,\delta} s_3$, $s_3 \xrightarrow{b,\nu} s_1$.

The reader might be puzzled by the use of labels in Markov Automata. Their rôle will become apparent in the following definition of interactive Markov automata, where the labels will help in defining which actions should co-operate and which should not.

Definition 2 (Interacting LMAs) *Let*

$$\mathcal{M}_1 = \langle \mathsf{S}_1, Act_1, \rightarrow_1 \rangle \quad \text{and} \quad \mathcal{M}_2 = \langle \mathsf{S}_2, Act_2, \rightarrow_2 \rangle$$

be two LMAs.

The interacting *LMA* $\mathcal{M}_1 \oplus_L \mathcal{M}_2 = \langle \mathsf{S}, Act, \rightarrow, \rangle$ *with* $L \subseteq Act_1 \cap Act_2$ *is a new automata defined as follows:*

1. $\mathsf{S} = \mathsf{S}_1 \times \mathsf{S}_2$.
2. $Act = Act_1 \cup Act_2$.
3. $\rightarrow$ *is the smallest relation defined by the rules below:*

$$\frac{s_1 \xrightarrow{a,\lambda}_1 s_1' \qquad s_2 \xrightarrow{a,x_a}_2 s_2'}{(s_1, s_2) \xrightarrow{a,\lambda} (s_1', s_2')} (a \in L)$$

$$\frac{s_1 \xrightarrow{a,x_a}_1 s_1' \qquad s_2 \xrightarrow{a,\lambda}_2 s_2'}{(s_1, s_2) \xrightarrow{a,\lambda} (s_1', s_2')} (a \in L)$$

$$\frac{s_1 \xrightarrow{a,r}_1 s_1'}{(s_1, s_2) \xrightarrow{a,r} (s_1', s_2)} (a \notin L)$$

$$\frac{s_2 \xrightarrow{a,r}_2 s_2'}{(s_1, s_2) \xrightarrow{a,r} (s_1, s_2')} (a \notin L)$$

We reserve the Greek letters to range over $\mathbb{R}^+$, *the and the Romans* u, q, p *over* $\mathbb{R}^+ \cup \mathbf{Var}$ *and the letter* $x_a, y_a, z_a \ldots$ *or simply* $x, y, z, \ldots$ *to range over* **Var**. *We call the set of labels* L *the cooperation set.*

The definitions above clearly show that LMAs are more than simple labeled CTMCs. First of all, transitions are divided into active and passive. *Active transitions* are those with an associated *delay*, i.e., a rate which is a real number; *Passive transitions* are those whose delays are undefined, i.e.,the rate is a variable. Passive transitions are meaningful to define the cooperation among automata. Cooperation between automata happens only between an active and a passive action, never between two active actions or two passive actions. In the cooperation the unspecified rate moves at the speed of the automaton with the active rate. The meaning of passive transition is directly inspired by PEPA [19], instead of using the symbol $\top$ we use variables for convenience. We will see later that we are assuming that each state in each automaton has *at most one outgoing passive transition* with respect to each label. We do not need to attach weights to the passive actions as in PEPA [19]. Note that our cooperation is more restrictive with respect to that defined in PEPA, yet fully adequate to the purpose of our work here. Generalization of cooperating automata that deals with active-active transitions is possible, but outside the scope of our work. The use of variables at this point is just needed to denote that a passive transition occurs with an unknown rate, since it depends on the transitions of other automata. If an automaton does not contain any passive transition, then the underlying model description is a CTMC. To see this it suffices to associate to each transition $s_1 \xrightarrow{a,\lambda} s_2$ a random variable $X_{a,\lambda}$ such that $\mathbb{P}(X_{a,\lambda} \leq t) = 1 - e^{-\lambda t}$. On this basis we justify the following definitions.

Definition 3 (Open and closed automata) *We distinguish the following classes of automata:*

1. *A LMA* $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$ *is called* open *if there exists a label* $a \in Act$ *and a state* $s \in \mathsf{S}$ *such that a passively enabled in* s, *i.e.,* $\exists s' \in \mathsf{S}$ *such that* $s \xrightarrow{a,x_a} s'$ *and* $s \neq s'$.
2. *A LMA* $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$ *is called* closed *if is not open.*

Given a closed LMA $\mathcal{M}$ all the transitions are carried out according to an exponentially distributed random delay. If

more than one transition is possible from a state s then that with the minimum random delay will occur. Therefore, the state residence time is exponentially distributed and we can define the underlying time-homogenuous CTMC as follows:

- each state of the automaton is a state of the CTMC
- the transition rate $\mathbf{q}(s, s')$ from state s to state s' in the CTMC is given by the sum of the rates of all the labeled transitions of the automaton from s to s', i.e.:

$$\mathbf{q}(s, s') = \sum_{\substack{(a,\lambda): s \xrightarrow{a,\lambda} s' \\ s \neq s'}} \lambda$$

Differently from CTMC, an automaton can perform self-loops, which are not relevant in the underlying CTMC. Therefore, for closed automata we can directly derive the CTMC. In this setting we can also define the instantaneous transition $\mathbf{q}(s, a, s')$ due to activity-type a is defined as:

$$\mathbf{q}(s, a, s') = \sum_{\substack{\lambda: s \xrightarrow{a,\lambda} s' \\ s \neq s'}} \lambda.$$

Clearly, the stochastic process underlying any automaton defined by the interaction of two LMAs is a CTMC.

Finally, we introduce the notion of irreducible LMA.

Definition 4 (Reachability set) *Let $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$ be a LMA.*

1. *A state s' is said to be* reachable in one step *from s if for some $a \in Act$ and $t \in \mathbb{R}^+ \cup \mathbf{Var}$, $s \xrightarrow{a,t} s'$.*
2. *A state s_n is said to be* reachable *from s_1 if for some $a_1, \ldots, a_n \in Act$ and $t_1, \ldots t_n \in \mathbb{R}^+ \cup \mathbf{Var}$ and $s_2, \ldots, s_{n-1} \in \mathsf{S}$ we have $s_1 \xrightarrow{a_1,t_1} s_2 \xrightarrow{a_2,t_2} s_3 \ldots s_{n-1} \xrightarrow{a_n,t_n} s_n$.*

We write $\mathsf{Reach}(s)$ the set of all reachable states from s.

Definition 5 (Irreducible LMA) *A LMA $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$ is* irreducible *if for all $s \in \mathsf{S}$ $\mathsf{Reach}(s) = \mathsf{S}$.*

Once established how to derive the CTMC we can talk directly about the properties of the LMA, meaning that those properties refer to the CTMC. Therefore, we can talk about a stationary closed LMA or the steady-state distribution of the LMA meaning that its CTMC is stationary or it has a steady-state distribution. For steady-state distribution we write $\pi(\mathcal{M})$ meaning the $\vec{\pi}\mathbf{Q} = 0$, where $\mathbf{Q}$ is the generator matrix of the underlying CTMC of $\mathcal{M}$ and $\vec{\pi}$ is the vector of probability distribution of the states in S.

Since a complete analysis of LMAs modeling power is out of the scope of this paper, we introduce a set of restrictions that will simplify the presentation of the theoretical result about the product-form solutions of cooperating LMAs. It should be pointed out that although these restrictions limit the flexibility of LMA modeling, they do not reduce the applicability of the results that will follow. In practice, we require a *well-formed* LMA $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow \rangle$ to satisfy the following properties:

1. given a label $a \in Act$ then all the transitions labeled by a are either active or passive. Hence, we can say that label a is active or passive for the automaton. We call $\mathcal{A}(\mathcal{M})$ and $\mathcal{P}(\mathcal{M})$ the sets of active and passive labels, respectively. Formally:

$$\mathcal{A}(\mathcal{M}) \cap \mathcal{P}(\mathcal{M}) = \emptyset$$

2. if a is a passive label, then for every state s of the automaton there exists only one transition labeled by a outgoing from s. Formally, we define:

$$\forall s \exists s' \in \mathsf{S} \quad \text{such that} \quad s \xrightarrow{a,x_a} s'$$
$$\forall s, s', s'' \in \mathsf{S},\ s \xrightarrow{a,x_a} s' \wedge s \xrightarrow{a,x_a} s'' \implies s' = s''$$

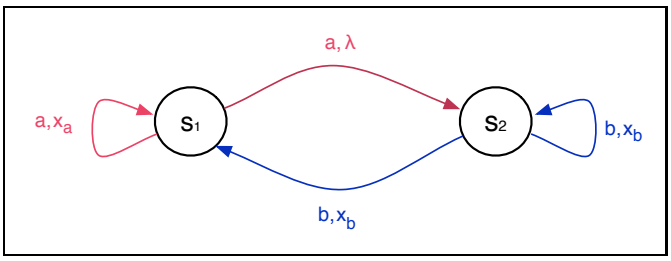

Figure 2: An automaton which is not well formed.

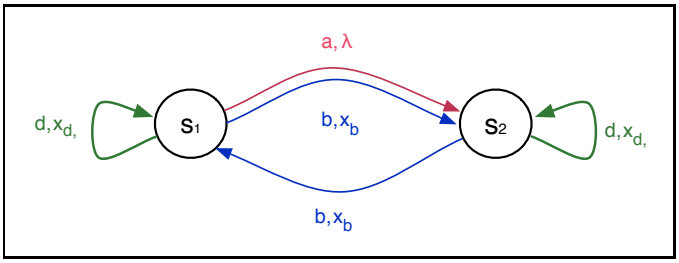

Figure 3: An automaton which is well-formed.

In Figure 2 we show an example of a not well-formed automaton. In fact, a passive and an active transition with label a are enabled in state s_1 which violates Condition (1); in the state s_2 two passive transitions with the same label b are enabled which violates Condition (2). Conversely, the automaton of Figure 3 is well-formed. Indeed, the passive transitions outgoing from state s_2 are differently labeled and Condition (1) holds. Moreover, $\mathcal{P}(\mathcal{M}) = \{b, d\}$ and every state of $\mathcal{M}$ has one outgoing transition labeled by b and one labeled by d.

For the rest of the paper we assume to work with well-formed stochastic automata.

Note that the interaction between two well-formed automata is well-formed, as the following proposition states.

Proposition 1 *Let $\mathcal{M}_1, \mathcal{M}_2$ be two well-formed LMAs. The interactive LMA $\mathcal{M}_1 \oplus_L \mathcal{M}_2$ is also well-formed.*

3. PRODUCT-FORM SOLUTIONS

Let us consider two (well-formed) automata $\mathcal{M}_1$ and $\mathcal{M}_2$. We are interested in those interactions $\mathcal{M}_1 \oplus_L \mathcal{M}_2$ that originate a closed LMA. This happens if $L = \mathcal{P}(\mathcal{M}_1) \cup \mathcal{P}(\mathcal{M}_2)$ and, obviously, $L \subseteq \mathcal{A}(\mathcal{M}_1) \cup \mathcal{A}(\mathcal{M}_2)$.

Before introducing the main theorem we define a last operation on the automata, i.e., the closure with respect to

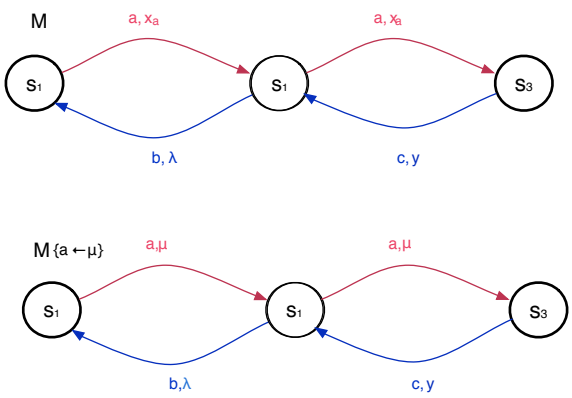

Figure 4: An open automaton where transition a becomes active

a label. Let us consider a well-formed open automaton $\mathcal{M}$ and let $a \in \mathcal{P}(\mathcal{M})$. Then the automaton $\mathcal{M}\{a \leftarrow \lambda\}$ is the automaton $\mathcal{M}$ in which every transition labeled a becomes active and all the rates associated with the transitions labeled with a are set to λ. Note that this means that we associate to each passive transition labeled a the same rate. We show in Figure 4 the effect of $\mathcal{M}\{a \leftarrow \mu\}$. Note that passive transition c in the original automaton $\mathcal{M}$ remains passive, but the transition a becomes active, and every occurrence of the transition has not the same rate μ. Formally, we define:

Definition 6 (Closure of an automaton) *Let*

$$\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow_{\mathcal{M}} \rangle$$

be a LMA and $a \in \mathcal{P}(\mathcal{M})$, then the closure *of $\mathcal{M}$ written $\mathcal{M}\{a \leftarrow \lambda\}$, is defined as follows:*

$$\mathcal{M}\{a \leftarrow \lambda\} = \langle \mathsf{S}, Act, \rightarrow_{\mathcal{M}\{a \leftarrow \lambda\}} \rangle,$$

where

$$\begin{aligned} \rightarrow_{\mathcal{M}\{a \leftarrow \lambda\}} = & \{(s_i, b, r, s_j) : (s_i, b, r, s_j) \in \rightarrow_{\mathcal{M}} \wedge b \neq a\} \\ & \cup \{(s_i, a, \lambda, s_j) : (s_i, a, x_a, s_j) \in \rightarrow_{\mathcal{M}}\} \end{aligned}$$

Several closures may be specified $(\mathcal{M}\{a \leftarrow \lambda_1\})\{b \leftarrow \lambda_2\}$ and a natural question is whether the order of the substitutions affects the outcome.

Proposition 2 *Let $\mathcal{M} = \langle \mathsf{S}, Act, \rightarrow_{\mathcal{M}} \rangle$ be a LMA such that $\mathcal{P}(\mathcal{M}) = \{a_1, a_2\}$. Then for any λ_1, λ_2 it holds that $(\mathcal{M}\{a_1 \leftarrow \lambda_1\})\{a_2 \leftarrow \lambda_2\} = (\mathcal{M}\{a_2 \leftarrow \lambda_2\})\{a_1 \leftarrow \lambda_1\}$*

The proposition above can be generalised to any number of labels. Thus, since the order is not important, we can write then $\mathcal{M}\{a_i \leftarrow \lambda_i : a_i \in \mathcal{P}(\mathcal{M})\}$ corresponds to $(((\mathcal{M}\{a_1 \leftarrow \lambda_1\})\{a_2 \leftarrow \lambda_2\}) \ldots \{a_n \leftarrow \lambda_n\})$ if $\{a_1, a_2, \ldots a_n\} \in \mathcal{P}(\mathcal{M})$.

In the following theorem we use $a_1, a_2, \ldots$ to denote the labels, and $x_1, x_2, \ldots$ to denote the variables (instead of x_{a_1}, $x_{a_2} \ldots$).

Theorem 1 *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two well-formed LMAs that cooperate on a finite set of labels $L = \{a_1, \ldots, a_n\}$, such that $\mathcal{M}_1 \oplus_L \mathcal{M}_2$ is ergodic.*

If there exists the set of rates $\{\lambda_1, \ldots, \lambda_n\}$ which satisfies the following equations:

$$\forall s_k \in \mathsf{S}_1, \forall a_i \in \mathcal{A}(\mathcal{M}_1) \quad \frac{\sum_{s_j \in \mathsf{S}_1} \mathbf{q}(s_j, a_i, s_k)\pi_1(s_j)}{\pi_1(s_k)} = \lambda_i \quad (1)$$

or

$$\forall s_k \in \mathsf{S}_2, \forall a_i \in \mathcal{A}(\mathcal{M}_2) \quad \frac{\sum_{s_j \in \mathsf{S}_2} \mathbf{q}(s_j, a_i, s_k)\pi_2(s_j)}{\pi_2(s_k)} = \lambda_i \quad (2)$$

where π_1 and π_2 are the stationary probability distributions of the closed automata $\mathcal{M}_1^\dagger$ and $\mathcal{M}_2^\dagger$:

$$\begin{aligned} \mathcal{M}_1^\dagger &= \mathcal{M}_1\{a_i \leftarrow \lambda_i : a_i \in \mathcal{P}(\mathcal{M}_1)\} \\ \mathcal{M}_2^\dagger &= \mathcal{M}_2\{a_i \leftarrow \lambda_i : a_i \in \mathcal{P}(\mathcal{M}_2)\} \end{aligned}$$

then the steady-state solution of $\mathcal{M}_1 \oplus_L \mathcal{M}_2$ has the product-form:

$$\pi(\mathcal{M}_1 \oplus_L \mathcal{M}_2) \quad \propto \quad \pi_1(\mathcal{M}_1^\dagger)\pi_2(\mathcal{M}_2^\dagger) \quad (3)$$

where π_i is the steady state distribution of $\mathcal{M}_i$, with $i = 1, 2$.

The solution of the system of equations (1) and (2) can be seen as the solution of the traffic equations of the model. It has been proved in [14] that if we model Jackson queueing networks then these equations become exactly the traffic equations of queueing network, and similarly for G-networks [15]. Finally, note that Equations (1) and (2) basically say that the total flow incoming into a state due to the active transitions labeled by a must be proportional to the stationary probability of that state in the closure of the automaton.

Proof. We present the proof assuming that automata synchronize on label 'a' only. This is only for readability. The proof with any number of synchronizing labels is a simple generalization of the one presented below. Without loss of generality we assume that a is active in $\mathcal{M}_1$ and passive in $\mathcal{M}_2$. The global balance equations (GBEs) for a $r \in \mathsf{S}_{\mathcal{M}_1}$ and $s \in \mathsf{S}_{\mathcal{M}_2^\dagger}$ are:

$$\begin{aligned} &\pi_{\mathcal{M}_1}(r)\Big(\sum_{r' \in \mathsf{S}_{\mathcal{M}_1}} \mathbf{q}_{\mathcal{M}_1}(r, a, r') + \sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_1}(r, b, r')\Big) = \\ &\mathbf{q}_{\mathcal{M}_1}(r', a, r)\pi_{\mathcal{M}_1}(r') + \sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_1}(r', b, r)\pi_{\mathcal{M}_1}(r') \quad (4) \end{aligned}$$

$$\begin{aligned} &\pi_{\mathcal{M}_2^\dagger}(s)\Big(\underbrace{\mathbf{q}_{\mathcal{M}_2^\dagger}(s, a, s')}_{\lambda} + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s, b, s')\Big) = \\ &\sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s', b, s)\pi_{\mathcal{M}_2^\dagger}(s') + \sum_{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger}} \underbrace{\mathbf{q}_{\mathcal{M}_2^\dagger}(s', a, s)}_{\lambda} \pi_{\mathcal{M}_2^\dagger}(s') \end{aligned} \quad (5)$$

where λ is the rate that we substitute into the passive transition of $\mathcal{M}_2$ to make it $\mathcal{M}_2^\dagger$.

The GBEs for the joint state space $(r,s) \in \mathsf{S}_{\mathcal{M}_1} \times \mathsf{S}_{\mathcal{M}_2}$ are:

$$\pi\big((r,s)\big)\Big(\sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}((r,s),b,(r',s)) + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2} \\ b \neq a}} \mathbf{q}((r,s),b,(r,s')) + \sum_{(r',s') \in \mathsf{S}_{\mathcal{M}_1} \times \mathsf{S}_{\mathcal{M}_2}} \mathbf{q}((r,s),a,(r',s'))\Big) =$$
$$\sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}((r',s),b,(r,s))\pi\big((r',s)\big) + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2} \\ b \neq a}} \mathbf{q}((r,s'),b,(r,s))\pi\big((r,s')\big) + \sum_{(r',s') \in \mathsf{S}_{\mathcal{M}_1} \times \mathsf{S}_{\mathcal{M}_2}} \mathbf{q}((r',s'),a,(r,s_i))\pi\big((r',s')\big) \quad (6)$$

We substitute product form and we rewrite the rates in each terms with the rates of each automaton taking into account self loops:

$$\pi_{\mathcal{M}_1}(r)\pi_{\mathcal{M}_2^\dagger}(s)\Big(\sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_1}(r,b,r') + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s,b,s') + \sum_{(r,s)^{a,\mathbf{q}\xrightarrow{(r,a,r')}(r',s')} } \mathbf{q}_{\mathcal{M}_1}(r,a,r') + \sum_{(r,s)^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s')} } \mathbf{q}_{\mathcal{M}_1}(r,a,r)\Big)$$
$$= \sum_{\substack{r' \in \mathsf{S}_{\mathcal{M}_1} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_1}(r',b,r)\pi_{\mathcal{M}_1}(r')\pi_{\mathcal{M}_2^\dagger}(s) + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s',b,s)\pi_{\mathcal{M}_1}(r)\pi_{\mathcal{M}_2^\dagger}(s') + \sum_{(r',s')^{a,\mathbf{q}\xrightarrow{(r',a,r)}(r,s)} } \mathbf{q}_{\mathcal{M}_1}(r',a,r)\pi_{\mathcal{M}_1}(r')\pi_{\mathcal{M}_2^\dagger}(s') + \sum_{(r,s')^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s)} } \mathbf{q}_{\mathcal{M}_1}(r,a,r)\pi_{\mathcal{M}_1}(r)\pi_{\mathcal{M}_2^\dagger}(s')$$

After a few algebraic manipulations substituting the right part of equation (4) we obtain:

$$\mathbf{q}_{\mathcal{M}_1}(r',a,r)\frac{\pi_{\mathcal{M}_1}(r')}{\pi_{\mathcal{M}_1}(r)} + \sum_{(r,s)^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s')} } \mathbf{q}_{\mathcal{M}_1}(r,a,r) + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s,b,s') = \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s',b,s)\frac{\pi_{\mathcal{M}_2^\dagger}(s')}{\pi_{\mathcal{M}_2^\dagger}(s)} + \sum_{(r',s')^{a,\mathbf{q}\xrightarrow{(r',a,r)}(r,s)} } \mathbf{q}_{\mathcal{M}_1}(r',a,r)\frac{\pi_{\mathcal{M}_1}(r')\pi_{\mathcal{M}_2^\dagger}(s')}{\pi_{\mathcal{M}_2^\dagger}(s)\pi_{\mathcal{M}_1}(r)} + \sum_{(r,s')^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s)} } \mathbf{q}_{\mathcal{M}_1}(r,a,r)\frac{\pi_{\mathcal{M}_2^\dagger}(s')}{\pi_{\mathcal{M}_2^\dagger}(s)}$$

By observing that λ is the reversed rate and with further algebraic manipulations we obtain the following.

$$\pi_{\mathcal{M}_2^\dagger}(s)(\mathbf{q}_{\mathcal{M}_2^\dagger}(s,a,s') + \sum_{(r,s)^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s')} } \mathbf{q}_{\mathcal{M}_1}(r,a,r) + \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s,b,s')) = \sum_{\substack{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger} \\ b \neq a}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s',b,s)\pi_{\mathcal{M}_2^\dagger}(s') + \sum_{s' \in \mathsf{S}_{\mathcal{M}_2^\dagger}} \mathbf{q}_{\mathcal{M}_2^\dagger}(s',a,s)\pi_{\mathcal{M}_2^\dagger}(s') + \sum_{(r,s')^{a,\mathbf{q}\xrightarrow{(r,a,r)}(r,s)} } \mathbf{q}_{\mathcal{M}_1}(r,a,r)\pi_{\mathcal{M}_2^\dagger}(s')$$

This latter can be derived from (5).

□

Note that in the proof above self-loops were considered. While self-loops do not change the behavior of the Markov Chain, i.e., they have no effect on the global balance equations of a single automaton, they are important in the definition of the structure of the interacting LMA.

Note that Theorem 1 implies that every state of an automaton must have at least one incoming transition for each active label.

Corollary 1 *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two well-formed LMAs that cooperate on a finite set of labels $L = \{a_1, \ldots, a_n\}$ such that they satisfy the condition in Theorem 1. If $a_i \in \mathcal{A}(\mathcal{M}_j)$ then for all $s \in \mathsf{S}_j$ there exists a s' such that $s' \xrightarrow{a_i,\mu} s$ with $\mu > 0$ and $a_i \in L$ and $j = 1, 2$.*

3.1 Comparison of Theorem 1 with other results on product-form solutions

In this part of the section we briefly illustrate the differences between the result stated by Theorem 1 an other results on product-form. We show that, as far as we know, the novelty of Theorem 1 is *not* just the reformulation of a well-known result in terms of LMAs. In particular, we wish to point out the differences with a couple of general results about product-form models, i.e., RCAT [14] and the $M \Rightarrow M$ property [27]. We show in terms of simple examples that Theorem 1 is more general than both these results.

Comparison with $M \Rightarrow M$.

The $M \Rightarrow M$ property has been introduced almost at the same time of the BCMP theorem definition [7]. The main result proved in [27] states that a multi-class queueing network whose stations fulfill the $M \Rightarrow M$ property have a product-form solution and a linear system of traffic equations. A work-conserving, multi-class queueing station without priority and with the one-step behavior (for a definition of these concepts see, e.g., [22]), fulfills the $M \Rightarrow M$ property if under class-independent Poisson arrivals it exhibits class-independent Poisson departures. Obviously, since the state is work-conserving by hypothesis, the departure rate for a customer class is equal to the arrival rate for that class. We now briefly illustrate the necessary and sufficient condidition to decide the $M \Rightarrow M$ property for a multi-class queueing station. Let Γ be the state space, $c = 1, \ldots, C$ be the customer classes and λ_c^* be the arrival rate for class c.

We assume that the station has a stationary solution $\pi(s_i)$ with $s_i \in \Gamma$. The condition for the $M \Rightarrow M$ property is that:

$$\forall c, \forall s_i \in \Gamma \quad \sum_{s_j \in \mathcal{L}^{+c}(s_i)} \pi(s_j) q(s_j, s_i) = \pi(s_i) \lambda_c^*, \quad (7)$$

where $\mathcal{L}^{+c}(s_i)$ is the set of all the states in Γ with a customer of class c more than those that are present in s_i and $q(s_j, s_i)$ is the transition rate from state s_j to state s_i. If we model such a queueing station in our framework, we define an automaton whose state space is the same state space of the queueing station. We may label each active transition of the automaton corresponding to the departure of a class c customer by d_c. Note that, since the station is work-conserving, if state s_j has one more class c customer than state s_i, and s_i is reachable form s_j, then the transition(s) from s_j to s_i must be labeled by d_c. It is now trivial to observe that Equation (7) is Condition (1) where $\lambda_c = \lambda_c^*$ for $c = 1, \ldots, C$. Therefore, in the framework we are presenting we are able to deal with the product-form of all the set of stations that satisfy the $M \Rightarrow M$ property that includes the BCMP stations (that may be multi-class and with Coxian service time distribution) and more recent ones (e.g., [1, 25])

Comparison with RCAT.

In this paragraph we aim to point out the main difference between Theorem 1 and RCAT.

- RCAT is formulated in terms of PEPA and it is based on the analysis of the reversed process of the interacting agents.
- We provide a novel and elegant proof of our theorem using a general form of balance equations.
- The notion of active and passive actions is identical to the usage of active and passive transitions provided in the current framework. Our notion of well-formed automaton reflects a structural condition of RCAT, i.e., for each passive label a, exactly one passive transition outgoes from every state. Moreover, RCAT requires that:

$$\forall s_k \in \mathsf{S}_1, \forall a_i \in \mathcal{A}(\mathcal{M}_1) \quad \frac{\mathbf{q}(s_j, a_i, s_k)\pi_1(s_j)}{\pi_1(s_k)} = \lambda \quad (8)$$

  where s_j is the *only* state in $\mathcal{M}_1^\dagger$ with an active transition labeled by a_i going into that state s_k. If we observe that λ_i is the reversed rate of $\mathbf{q}(s_j, a_i, s_k)$ [23], then we can see that RCAT requires that the reversed rate of any active action a has to be constant. By contrast, in our theorem we require the *sum* of the reversed rates of the active transitions to be constant in each state rather that the reversed rate of each active transition. This is a generalization of the condition of RCAT. The distinction presented here is not trivial. In fact, in what follows we show an example in which Theorem 1 holds while the original RCAT does not.

Example 2 *In this example we study the interaction between the automata depicted in Figure 5 (A) and (B). In particular, we show that the automaton generated by the composition is in product-form by Theorem 1 but not by RCAT. Let us call the automaton of Figure 5 (A) $\mathcal{M}_1$ and that of Figure 5 (B) $\mathcal{M}_2$. Since $\mathcal{M}_1$ has no passive transitions, i.e., it is closed, we immediately have $\mathcal{M}_1 = \mathcal{M}_1^\dagger$. Hence, we can immediately verify Condition (1) knowing that $\pi(1) = 1/5$ and $\pi(2) = 4/5$:*

$$\begin{aligned} &\textit{state } 1: && \pi_1(2)\mathbf{q}(2, a, 1) = \pi_1(1)\lambda \\ &\textit{state } 2: && \pi_1(1)\mathbf{q}(1, a, 2) + \pi_1(2)\mathbf{q}(2, a, 2) = \pi_1(2)\lambda. \end{aligned}$$

Condition 1 is verified for $\lambda = 1$. Now we can derive the automaton $\mathcal{M}_2^\dagger = \mathcal{M}_2\{a \leftarrow 1\}$, which has the following stationary distribution: $\pi_2(A) = 2/3$ and $\pi_2(B) = 1/3$. By Theorem 1 if (i, j) with $i = 1, 2$ and $j = A, B$ is a state of the automata $\mathcal{M}_1^\dagger \oplus_{\{a\}} \mathcal{M}_2^\dagger$ then $\pi(i, j) \propto \pi_1(i)\pi_2(j)$.

It is worth noting that in this case RCAT cannot be applied as the reversed rate of the active transitions with label a are not *constant: $2 \xrightarrow{a,7/8} 2$ thus by Equation (8) $\frac{(4/5)(7/8)}{4/5} = 7/8$ but $1 \xrightarrow{a,1/2} 2$ thus $\frac{(1/2)(1/5)}{4/5} = 1/8$.*

4. EXAMPLE

In this section we illustrate an example of a software and hardware architecture that can be studied by the theoretical results provided in this paper. The aim of this example is showing a model whose product-form is not trivial, i.e., it cannot be straightforwardly studied with the existing tools. Indeed, the model consists of a component which can be seen as non-BCMP queueing station that satisfies the $M \Rightarrow M$ property, and other components that are G-queues. As a consequence, neither the tools developed for the analysis of BCMP queueing networks, neither those developed for the analysis of the G-networks can be applied and our approach shows its peculiarity.

System description

Let us consider a system that consists of two databases, namely *DB1* and *DB2*. *DB1* serves the requests of *TYPEA*, while *DB2* serves the requests of *TYPEB*. The requests arrive to the databases through a communication line that has two channels. Each of them works just in one direction at a given time, i.e., from the databases to outside or vice-versa. A request (both of *TYPEA* and *TYPEB*) arrives to the communication line and waits in queue for an available channel according to a FCFS discipline. Once the transmission starts we assume that it takes a random time to be completed. The databases serve the requests in a random time. The databases require a sort of synchronization and it can happen that a transaction in the first database causes a canceling of a transaction in the second database (or vice-versa). A further assumption is that we require the requests of the same type to be processed in the order they arrive to the communication line. By now, this simply implies that even if the communication line has an available channel, it is not allowed to transmit two requests of the same type, because it could happen that the transmission of the most recent request takes less time than the transmission of the oldest. When the databases finish the processing of the requests, the answer is sent back through the same communication line. Figure 6 illustrates an informal sketch of the system.

Modeling assumptions

In order to be able to model the system and obtain an exact solution we need these further assumptions:

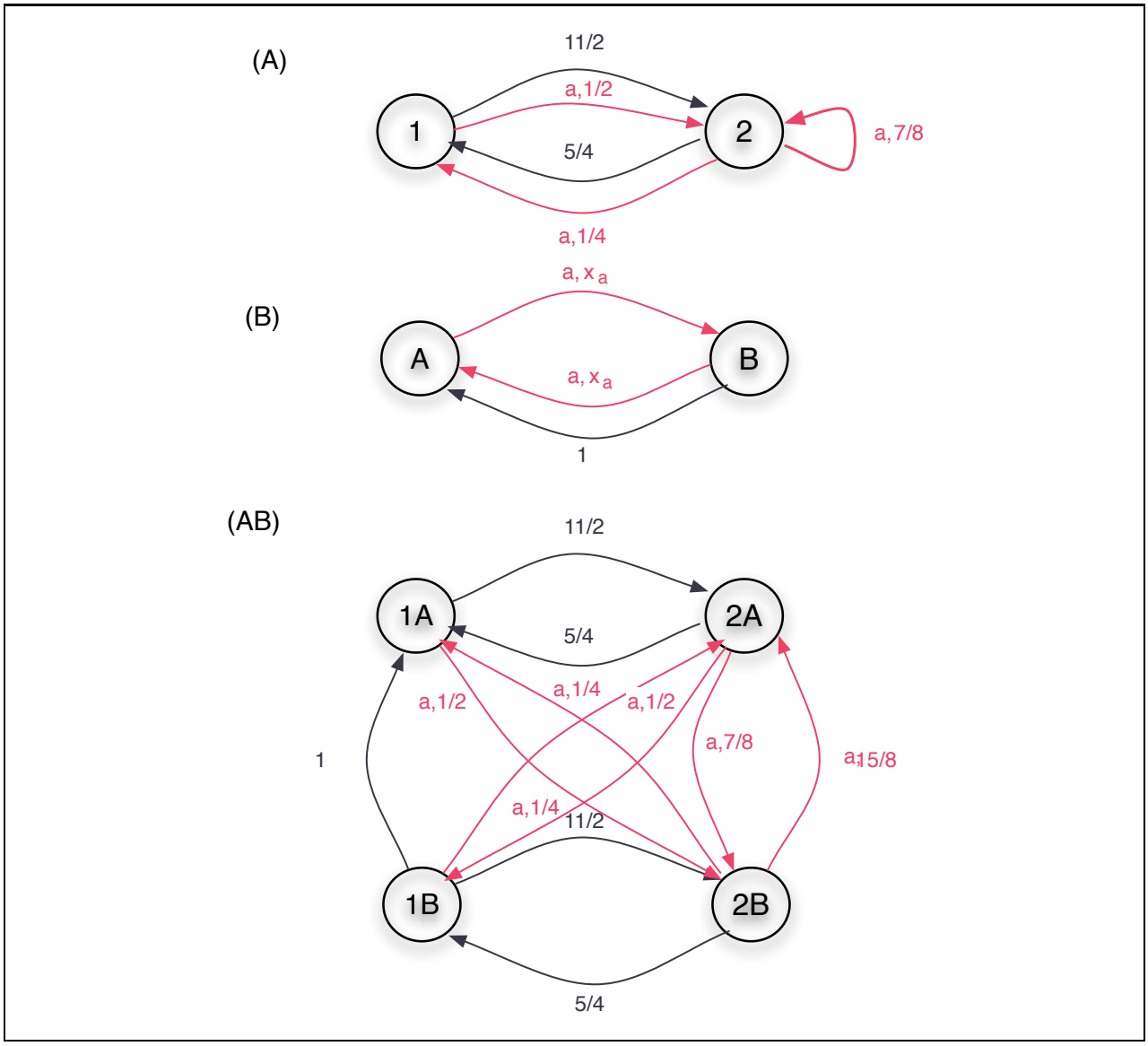

Figure 5: Graphical representation of the Markov automaton $\mathcal{M}_1$

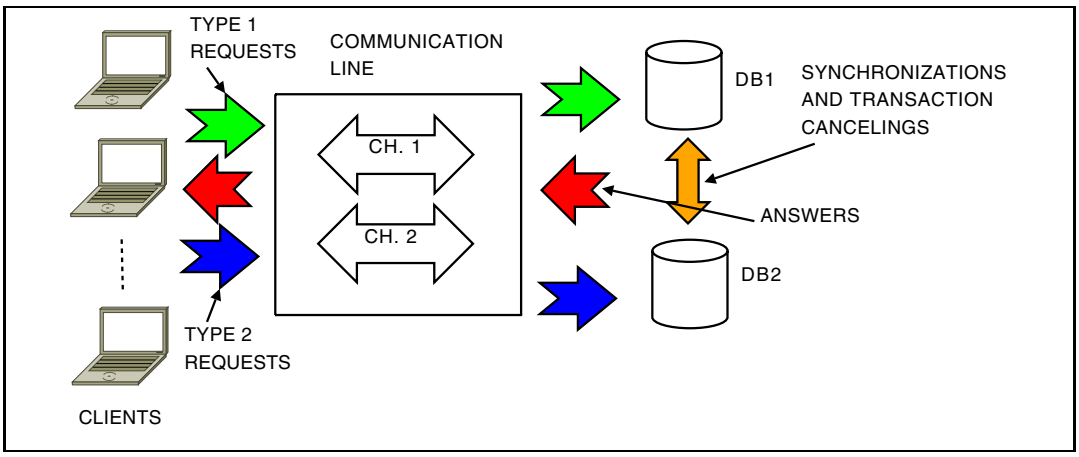

Figure 6: Sketch of the system analyzed in Section 4

- The requests of *TYPEA* and *TYPEB* arrive to the system according to independent Poisson processes with rates λ_1 and λ_2, respectively.
- The transmission type in the communication line is exponentially distributed, and the two lines work independently. Moreover, the transmission time is independent of the type of the message being sent (*TYPEA* or *TYPEB* request, or an answer). Let μ_{TR} be the transmission rate of one message.
- The databases process the requests in an exponentially distributed random time. *TYPEA* requests are served by *DB1* with rate μ_1 and *TYPEB* are served by *DB2* with rate μ_2.
- A *TYPEA* (*TYPEB*) request, after being served by *DB1*(*DB2*) generates a request for *DB2* (*DB1*) with probability p_1 (p_2) and the latter one will send the answer back. Finally, a *TYPEA* (*TYPEB*) request causes a transaction cancel in *DB2* (*DB1*) with probability q_1 (q_2). Note that $1 - p_i - q_i > 0$ is the probability that a database directly sends the answer back to the client without synchronizing with the other one, for $i = 1, 2$.

Figure 7 shows a model of the system using the queueing network usual notation.

Description of the model using LMAs

Although the queueing model represented in Figure 7 is useful for understanding the flows of the messages (requests or answers) among the network components, it still is incomplete. Indeed, none of the queues is a standard exponential queue of the type studied in [21] or in [7]. In other words, the modeler still needs to formally specify the behavior of those components, and how they interact. We think that using LMAs is a valid answer to these two problems.

In practice, we can identify two types of components. The communication line with two channels can be seen as a Multiserver Station with Concurrent Classes of Customers (MSCCC) as described in [25] while the databases can be seen as G-queues. In a G-queue there are two arrival streams: one of positive customers that behave exactly as normal customers of standard queues, and one of negative customers.

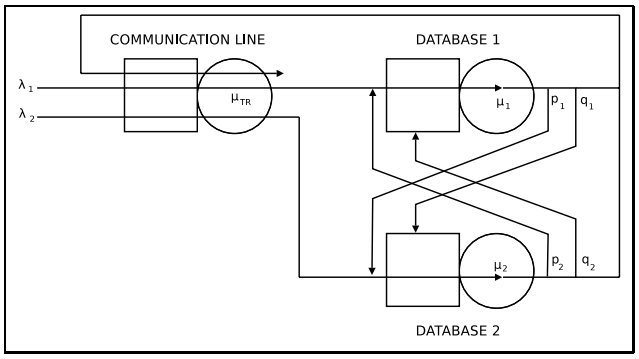

Figure 7: Queueing model of the system depicted by Figure 6

As soon as a negative customer arrives to a G-queue it deletes a queued positive customer if any exists, otherwise it simply vanishes. More details on applications and analysis of G-queues and G-networks can be found in [12]. Note that, as far as we now, such a composition of different models has never been studied before.

The results of this paper are applied to this example in two ways: first we use LMAs to describe the interactions among models specified in different contexts and second we use Theorem 1 to derive the product-form solution, showing that previous results would not be straightforwardly applicable.

LMA of MSCCC station. A MSCCC station with three classes of customers (*TYPEA* and *TYPEB* requests and the answers) and two servers, i.e., the two channels, can be modeled as follows. Let $\mathcal{A}(\text{MSCC}) = \{d_1, d_2, d_3\}$, where d_1, d_2 and d_3 are the labels associated with the transitions that model the departures of *TYPEA* or *TYPEB* requests and the answers. $\mathcal{P}(\text{MSCCC}) = \{a_1, a_2, a_{13}, a_{23}\}$ where a_i are the labels associated with customer arrivals: a_1 and a_2 are the external arrivals, a_{j3} are the arrivals from DBj. Let $\alpha = (\vec{s}, \mathcal{L})$ be a state of the automaton, where: $\vec{s} = (c_1, c_2, c_3)$, $c_i \in \{0, 1\}$, denotes the classes of the customers being transmitted. Obviously, $\sum_{j=1}^{3} c_j \leq 2$.

$\mathcal{L} = \ell_1, \ell_2, \ldots \ell_i, \ldots, \ell_N$ is an ordered list representing the waiting room, where $\ell_i \in \{1,2,3\}$ is the class of the $i-th$ oldest customer in the queue. Note that if $\sum_{j=1}^{3} c_j < 2$ then $\ell_i = c_k$ for all $\ell_i \in \mathcal{L}$ and $c_k = 1$. In what follows $\vec{1}_i$ denotes a three-dimensional vector with 2 null components and a 1 in position i, $\mathcal{L} \cdot i$ denotes the append of a customer in queue according to a FCFS policy, $f_{\mathcal{W}}(\alpha)$ denotes the class of the first customer that does not belong to set $\mathcal{W} \subseteq \{1,2,3\}$ ($f_{\mathcal{W}} = 0$ if it does not exist) and $\mathcal{L} \setminus i$ removes the first class i customer from the list. Then, the possible transitions are:

- Class i customer arrival with busy channels: $(\vec{s}, \mathcal{L}) \overset{a_i, x_{a_i}}{\longrightarrow} (\vec{s}, \mathcal{L} \cdot i)$ if $\sum_{j=1}^{3} c_j = 2$ or $c_i = 1$.
- Class i customer arrival with an available channel: $(\vec{s}, \mathcal{L}) \overset{a_i, x_{a_i}}{\longrightarrow} (\vec{s} + \vec{1}_i)$ if $\sum_{j=1}^{3} c_j < 2$ and $c_i = 0$.
- Class i job completion, i.e., $c_i^s > 0$. In this case we have two possibilities, i.e., let $\mathcal{W} = \{j | c_j > 0\} \setminus \{i\}$:

$$(\vec{s}, \mathcal{L}) \overset{d_i, \mu_{\text{TR}}}{\longrightarrow} \begin{cases} (\vec{s} - \vec{1}_i, \mathcal{L}) & \text{if } f_{\mathcal{W}}(\vec{s}, \mathcal{L}) = 0 \\ (\vec{s} - \vec{1}_i + \vec{1}_k, \mathcal{L} \setminus k) & \text{if } f_{\mathcal{W}}(\vec{s}, \mathcal{L}) = k \end{cases}$$

Note that list $\mathcal{L}$, under stability condition, is always finite although it is unlimited if the population is unlimited. Nevertheless, the state space of the LMA is countable.

LMA of the G-queue.

For those who are familiar with G-queues and G-networks modeling a G-queue by an LMA is trivial. Let us consider the G-queue corresponding to *DB1*(the other LMA can be obtained by symmetry). The LMA passive actions are associated with the positive customer arrivals, and the negative customer arrivals. In the former case, we can have a synchronization with the LMA MSCCC (i.e., label d_1) or with the other G-queue (i.e., label D_{21+}). In the latter case, we have a synchronization with the other G-queue, i.e., label D_{21-}. Hence, we have $\mathcal{P}(DB_1) = \{d_1, D_{21+}, D_{21-}\}$. The active transitions are those corresponding to a departure for the communication channel (a_1), a departure for the other G-queue either as positive (D_{12+}) or negative (D_{12-}) customer. Hence, $\mathcal{A}(DB_1) = \{a_1, D_{12+}, D_{12-}\}$. The state space is $\mathbb{N}$ in case of unlimited population, and let s be a state. Then, the transitions from s are:

- Customer arrival from MSCCC: $s \overset{d_1, x_{d_1}}{\longrightarrow} s+1$
- Positive customer arrival from DB2: $s \overset{D_{21+}, x_{D_{21+}}}{\longrightarrow} s+1$
- Negative customer arrival from DB2:

$$s \overset{D_{21-}, x_{D_{21-}}}{\longrightarrow} \begin{cases} s-1 & \text{if } s > 0 \\ s & \text{if } s = 0 \end{cases}$$

- Job completion ($s > 0$): $s \overset{\delta, \Delta}{\longrightarrow} s-1$, where (δ, Δ) is either $(D_{12+}, \mu_1 p_1)$, $(D_{12-}, \mu_1 q_1)$ or $(a_1, \mu_1(1-p_1-q_1))$ according to the destination of the customer, i.e., DB_2 as positive customer, DB_2 as negative customer or MSCCC.

LMA of the external arrival.

The external arrivals are simple LMAs with one state 0 and one active transition from 0 to itself labeled by a_i, with $i = 1, 2$ and rate λ_i. Let us call these automata A_1 and A_2 for *TYPEA* and *TYPEB* arrivals, respectively.

Combining the models.

The automaton that we are going to study is:

$$M = (((\text{MSCCC} \oplus_{\{a_1\}} A_1) \oplus_{\{a_2\}} A_2) \oplus_{\{d_1, a_{13}\}} DB_1) \oplus_{\{d_2, a_{23}, D_{12+}, D_{12-}, D_{21+}, D_{21-}\}} DB_2$$

Model analysis and product-form solution

Before solving the model, we recall some theoretical results about MSCCC stations and G-queue. In [25] the author derives the stationary distribution of MSCCC stations and proves that it satisfies the $M \Rightarrow M$ property. In [15] the author proves that the product-form of the G-queues (with several extensions) can be seen as an application of RCAT. However, it is worthwhile pointing out that MSCCC does not satisfy RCAT. Indeed, if we consider the state $(1,1,0,\{\})$, then we observe that it can be reached from $(0,1,1,\{1\})$ or $(1,0,1,\{2\})$ through a transition with the same label d_3, i.e., a class 3 job completion. On the other hand, it is well-know the G-queues do not satisfy the $M \Rightarrow M$ property.

In this case, Theorem 1 can be applied in order to obtain the stationary distribution of the model, but it would not be possible to apply only RCAT or only the $M \Rightarrow M$ theorem.

The traffic equations.

The traffic equations are derived by the application of Theorem 1. The unknowns are the rates of the xs:

$$\begin{cases} x_{a_1} = \lambda_1 \\ x_{a_2} = \lambda_2 \\ x_{a_{13}} = \frac{x_{d_1}}{x_{D21-} + \mu_1} \mu_1 (1 - p_1 - q_1) \\ x_{a_{23}} = \frac{x_{d_2}}{x_{D12-} + \mu_2} \mu_2 (1 - p_2 - q_2) \\ x_{d_1} = x_{a_1} \\ x_{d_2} = x_{a_2} \\ x_{D_{12+}} = \frac{x_{d_1}}{x_{D21-} + \mu_1} \mu_1 p_1 \\ x_{D_{12-}} = \frac{x_{d_1}}{x_{D21-} + \mu_1} \mu_1 q_1 \\ x_{D_{21+}} = \frac{x_{d_2}}{x_{D12-} + \mu_2} \mu_2 p_2 \\ x_{D_{21-}} = \frac{x_{d_2}}{x_{D12-} + \mu_2} \mu_2 q_2 \end{cases} \tag{9}$$

The solution of the system can be obtained with standard numerical algorithms for non-linear equations. However, in most of the cases, we may not know that a model satisfies the $M \Rightarrow M$ property or RCAT conditions, and we may not know the expression of the reversed rates of the active transitions. Indeed, System (9) has been derived thanks to the theoretical knowledge of the LMAs that the model consists of. In general, we would like to specify the LMAs, their interactions, and apply an algorithm that tells us whether there is product-form or not, and the solution for the unknowns. Although an exact solution of this problem is not yet available, in [26] the authors define an interactive algorithm that has shown to perform well for a large class of models.

Once the unknowns are found, each LMA of the model can be closed and solved, since they have an underlying CTMC. Then, Theorem 1 states that the stationary distribution of the model is proportional to the product of the stationary distributions of the interacting LMAs closed as specified.

5. CONCLUSION

In this paper we have proposed an extension of the Reversed Compound Agent Theorem (RCAT) for the analysis

of product-form models. In particular, we have shown that it possible to relax its structural conditions in order to deal with more general synchronization types. We think that one of the strengths of Theorem 1 is that it unifies two of the most important results (at in least in our opinion) about product-form models: the original RCAT and the Markov implies Markov property ($M \Rightarrow M$). This property has been widely used to explain the product-form solutions of the queueing networks and to define service disciplines that extend the BCMP theorem [7].

Another peculiar aspect of this work concerns the formalism that we decided to adopt, i.e., the LMAs. Although this is not essential for the validity of the theoretical results, we think that it has some strengths with respect to other formalism. In particular, the idea of closure of an automaton, and the structural conditions of Theorem 1 should result easy to understand. Moreover, using LMAs allows us to work in a very general framework, because we can define a LMA equivalent to any stochastic model with an underlying CTMC. For instance, this can be very helpful in a tool that allows for the definition of models whose components are specified using different formalisms.

We think that future works should deal with the possibility of defining automata with multiple passive transitions (with the same label) outgoing from the states, and with synchronizations that involve more than two automata. Moreover, the definition of an efficient technique to compute the normalizing constant is still an open problem (shared with many other product-form model classes) that should be addressed.

6. ACKNOWLEDGMENTS

We gratefully acknowledge Simonetta Balsamo, Peter Harrison and Jane Hillston for the useful and insightful discussions on various parts of this paper.

7. REFERENCES

[1] P. V. Afshari, S. C. Bruell, and R. Y. Kain. Modeling a new technique for accessing shared buses. In *Proc. of the Computer Network Performance Symp.*, pages 4–13, New York, NY, USA, 1982. ACM Press.

[2] C. Baier and H. Hermanns. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Trans. Softw. Eng.*, 29(6):524–541, 2003.

[3] G. Balbo, S. C. Bruell, and M. Sereno. Product form solution for Generalized Stochastic Petri Nets. *IEEE Trans. on Software Eng.*, 28:915–932, 2002.

[4] G. Balbo, S. C. Bruell, and M. Sereno. On the relations between BCMP Queueing Networks and Product Form Solution Stochastic Petri Nets. *Proc. of 10th Int. Workshop on Petri Nets and Performance Models, 2003.*, pages 103–112, 2003.

[5] S. Balsamo, R. Mamprin, and M. Marzolla. Performance evaluation of software architectures with Queueing Network Models. In Carmen Bobenau, editor, *Proc. of the European Simulation and Modeling Conf. (ESMc'04)*, pages 206–213, UNESCO, Paris, France, 2004.

[6] S. Balsamo and M. Marzolla. Performance evaluation of UML software architectures with multiclass Queueing Network models. In *WOSP '05: Proc. of the 5th int. workshop on Soft. and Perf.*, pages 37–42, Palma, Illes Balears, Spain, 2005. ACM.

[7] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.

[8] R. J. Boucherie. A characterisation of independence for competing Markov chains with applications to stochastic Petri nets. *IEEE Tran. on Software Eng.*, 20(7):536–544, 1994.

[9] G. Clark. *Techniques for the Construction and Analysis of Algebraic Performance Models*. PhD thesis, The University of Edinburgh, 2000.

[10] J. L. Coleman, W. Henderson, and P. G. Taylor. Product form equilibrium distributions and a convolution algorithm for Stochastic Petri nets. *Perform. Eval., Elsevier*, 26:159–180, 1996.

[11] N.J. Dingle, P.G. Harrison, and W.J. Knottenbelt. Uniformization and hypergraph partitioning for the distributed computation of response time densities in very large Markov models. *Journal of Parallel and Distributed Computing*, 64(8):908–920, August 2004.

[12] E. Gelenbe. Product form networks with negative and positive customers. *Journal of Applied Prob.*, 28(3):656–663, 1991.

[13] P. Harrison and J. Hillston. Exploiting quasi-reversible structures in Markovian process algebra models. *The Computer Journal*, 38(7):510–520, 1995.

[14] P. G. Harrison. Turning back time in Markovian process algebra. *Theoretical Computer Science*, 290(3):1947–1986, January 2003.

[15] P. G. Harrison. Compositional reversed Markov processes, with applications to G-networks. *Perform. Eval., Elsevier*, 57(3):379–408, 2004.

[16] P. G. Harrison. Reversed processes, product forms and a non-product form. *Linear Algebra and Its Applications*, 386:359–381, July 2004.

[17] Peter G. Harrison and Maria G. Vigliotti. Algebraic product-forms do not need balance equations. Submitted to Computer Journal, 2009.

[18] Peter G. Harrison and Maria G. Vigliotti. Perturbation of a non-product-form network into a new product-form: equilibrium state probabilities and response time density. In *Proc. of Fourth Int. Conf. on Perform. Eval. Methodologies and Tools (VALUETOOLS'09)*, October 2009.

[19] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, Department of Computer Science, University of Edinburgh, 1994.

[20] J. Hillston and N. Thomas. Product form solution for a class of PEPA models. *Perform. Eval., Elsevier*, 35(3–4):171–192, 1999.

[21] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10:131–142, 1963.

[22] K. Kant. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992.

[23] F. Kelly. *Reversibility and stochastic networks*. Wiley, New York, 1979.

[24] J. Kramer and J. Magee. *Concurrency: State Models Java Programs, 2nd Edition*. Worldwide Series in Computer Science. John Wiley Sons, April 2006.

[25] J. Y. Le Boudec. A BCMP extension to multiserver stations with concurrent classes of customers. In *SIGMETRICS '86/PERFORMANCE '86: Proc. of the 1986 ACM SIGMETRICS Int. Conf. on Computer perf. modelling, measurement and eval.*, pages 78–91, New York, NY, 1986. ACM Press.

[26] A. Marin and S. Rota Bulò. A general algorithm to compute the steady-state solution of cooperating markov chains. In *proc. of 17th Annual Meeting of the IEEE Int. Symp. on Modelling, Analysis and Simulation of Computer and Telecommunication System, MASCOTS 09*, pages 515–524, London, UK, 2009.

[27] R. R. Muntz. Poisson departure processes and queueing networks. Technical Report IBM Research Report RC4145, Yorktown Heights, New York, 1972.

[28] M. Sereno. Towards a product form solution for stochastic process algebras. *The Computer Journal*, 38(7):622–632, December 1995.

[29] Connie U. Smith and Lloyd G. Williams. Five steps to establish software performance engineering. In *Int. CMG Conference*, pages 507–516, Reno, Nevada, USA, 2006.