HET-NETs 2010
ISBN 978-83-926054-4-7
pp. 311–324

A tool for the numerical solution of cooperating Markov chains in product-form

SIMONETTA BALSAMO GIAN-LUCA DEI ROSSI
ANDREA MARIN [a]

[a]Università Ca' Foscari di Venezia
Dipartimento di Informatica
via Torino, 155 Mestre
Italy
{balsamo,deirossi,marin}@dsi.unive.it

Abstract:
Performance modelling of complex and heterogeneous systems based on analytical models are often solved by the analysis of underlying Markovian models. We consider performance models based on Continuous Time Markov Chains (CTMCs) and their solution, that is the analysis of the steady-state distribution, to efficiently derive a set of performance indices. This paper presents a tool that is able to decide whether a set of cooperating CTMCs yields a product-form stationary distribution. In this case, the tool computes the unnormalised steady-state distribution. The algorithm underlying the tool has been presented in [10] by exploiting the recent advances in the theory of product-form models such as the Reversed Compound Agent Theorem (RCAT) [5]. In this paper, we focus on the peculiarities of the formalism adopted to describe the interacting CTMCs and on the software design that may have interesting consequences for the performance community.
Keywords: : Product-form, queueing networks, numerical solution of Markov chains

1. Introduction

Markovian models have proved to be a robust and versatile support for the performance analysis community. Performance modeling of complex heterogeneous systems and networks based on analytical model usually describes a system using a high-level formalism, such as Stochastic Petri Nets (SPNs), Performance Evaluation Process Algebra (PEPA), queueing systems or networks, from which its underlying Continuous Time Markov Chain (CTMC) is derived. The desired performance indices, at steady-state, are computed by the analysis of the model

CTMC. This computation is usually a hard task, when not unfeasible, because the solution of the CTMC usually requires to solve the system of Global Balance Equations (GBEs) (with a computational complexity of $O(Z^3)$, where Z is the number of states of the model) to derive the stationary probability of each state. Some algorithms that numerically solve the GBEs more efficiently for special cases or using approximations have been defined.

Product-form models take a different approach. They apply the *divide et impera* paradigm to efficiently solve complex models. Informally, a model S is seen as consisting of several interacting sub-models $S_1, \ldots, S_N$ so that $\mathbf{m} = (m_1, \ldots, m_N)$ is a state of S and m_i is a state of S_i. S is in product-form with respect to S_1, ..., S_N if its stationary distribution $\pi(\mathbf{m})$ satisfies the following property:

$$\pi(\mathbf{m}) \propto \prod_{i=1}^{N} g_i(m_i),$$

where g_i is the stationary distribution of sub-model i considered in isolation and opportunely parametrised. Roughly speaking, from the point of view of a single sub-model S_i, the parametrisation abstracts out the interactions with all the other sub-models S_j, $j \neq i$. It should be clear that, since the state space of a sub-model S_i is much smaller than that of S the solution of its GBEs may be computed efficiently. Note that modularity becomes a key-point both for the analysis and the description of the model, since it is a good engineering practice to provide modular models of systems.

Exploiting the product-form solutions requires to address two problems: 1) Deciding if model S is in product-form with respect to the given sub-models S_1, ..., S_N; 2) Computing the parametrisation of the sub-models S_1, ..., S_N in order to study them in isolation. Note that we have not listed the solution of the sub-model CMTCs as a problem because we suppose that the cardinalities of their state spaces are small enough to directly solve the GBEs. If this is not the case, a product-form analysis of the sub-models may be hierarchically applied. In literature, the first problem has been addressed in two ways. The first consists in proving that a composition of models that yield some high-level characteristics is in product-form. For instance the BCMP theorem [2] is based on this idea because the authors specify four type of queueing disciplines with some service properties and prove that a network of such models has a product-form solution. The second way is more general, i.e., the properties for the product-form are defined at the CTMC level. Although this can lead to product-form conditions that are difficult to interpret, this approach really enhances the compositionality of the models. In this paper, we often refer to a recent result about product-forms: the Reversed Compound Agent Theorem (RCAT) [5]. This theorem has been extensively used to prove a large set

of product-form results previously known in literature (BCMP product-form [4], G-networks with various types of triggers [6], just to mention a few). Problem 2 is usually strictly related to Problem 1. In general, the parametrisation of the sub-models requires the solution of a system of equations that is called system of traffic equations. For several years the fact that product-form solutions must be derived from linear systems of traffic equations has been considered true, but the introduction of G-networks has shown that this is not necessary.

Contribution. This paper illustrates a tool that given the description of a set of cooperating CTMCs (i.e., when some transitions in one chain force transitions in other chains) it decides whether the model is in product-form and, in this case, computes its stationary distribution. The tool is based on the algorithm presented in [10] which is briefly resumed in Section 2.. Since the analysis of the product-form is performed at the CTMC level, it is able to study product-form models that are originated form different formalisms, such as exponential queueing networks, G-networks or queueing networks with blocking. To this aim, we observe that it is important to decouple the analyser and the model specification interface (MSI). We propose both a Java implementation of the analyser and of a general MSI (note that multiple specification interfaces may be implemented according to the modeller needs). With this tool, a modeller has a library of product-form models that, even if they were created using some (possibly high-level) formalism, are stored as stochastic automata (basically a CTMC with labelled transitions allowing self-loops or multiple arcs between states). Using the MSI (which acts as a client with respect to the analyser), the various sub-models can be instantiated and their interactions be specified. The operations that the modeller performs in the MSI are translated into commands for the server side, i.e., the analyser. The analysis is requested from the MSI, computed by the analyser and displayed by the MSI. We have also developed a textual interface that will not be presented in this paper to allow the usage of the analyser from non-graphical clients.

Paper structure. The paper is structured as follows. Section 2. briefly illustrates the formalism used to describe the interactive CTMCs, the idea underlying RCAT and the algorithm presented in [10]. Section 3. gives the details of the software implementation. In particular, Section 3.1. presents the naming conventions which are an important matter to enhance the modularity of the tool, Section 3.2. the client server architecture, and finally Section 3.3. gives a brief idea of some use-cases. Section 4. shows an instance of implemented MSI. Some final remarks conclude the paper in Section 5..

2. Theoretical background

In this section we briefly recall some basic definitions and models to keep the paper self-contained. We present the topics in a way that allows us to simplify the description of the tool algorithm, features and architecture in what follows.

Let us suppose to have N model $S_1, \ldots, S_N$ that cooperate, i.e., some transitions in a model S_i force other transitions in a model S_j, $i \neq j$. At a low-level we can describe each model by a set of labelled matrices: $\mathbf{M}_i^a$ is the matrix with label a associated with model S_i. Labels may be chosen arbitrarily when a model is defined. However, we always assume that every model has at least one label called ϵ. We consider, at first, models with a finite number of states, Z_i. $\mathbf{M}_i^a$ is a $Z_i \times Z_i$ matrix with non-negative elements that represent the transition rates between two states of the model. Note that self-loops, i.e., transitions from a state to itself, are allowed. The infinitesimal generator $\mathbf{Q}_i$ can be easily computed as the sum of all the matrices associated with a model, where the diagonal elements are replaced with the opposite of the sum of the extra-diagonal row elements. If the stationary distribution $\boldsymbol{\pi}$ exists (and hereafter we will work under this hypothesis) then it can be computed as the unique solution of $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ subject to $\boldsymbol{\pi}\mathbf{1} = 1$. From $\boldsymbol{\pi}$ we can compute the rates in the reversed process associated with each label [9, 5] in a straightforward way. Suppose that $\mathbf{M}_i^a[\alpha, \beta] > 0$, with $1 \leq \alpha, \beta \leq Z_i$ and $1 \leq i \leq N$, then the reversed rate of this transition, denoted by $\overline{\mathbf{M}_i^a[\alpha, \beta]}$ is defined as follows:

$$\overline{M_i^a[\alpha, \beta]} = \frac{\pi(\alpha)}{\pi(\beta)} M_i^a[\alpha, \beta]. \tag{1}$$

Let us show how we specify the interaction of two models. According to RCAT restrictions, we just deal with pairwise interactions, i.e., a transition in a model may cause a transition just for another model. The cooperation semantics used in this paper (but also in [5]) is very similar to that specified by PEPA, i.e., a Markovian stochastic process algebra introduced by Hillston in [8]. Consider sub-models S_i and S_j and suppose that we desire to express the fact that a transition labelled with a in S_i can occur only if S_j performs a transition labelled with b, and vice-versa. Specifically, if S_i and S_j are in states s_i, s_j such that they are able to perform a transition labelled with a and b, respectively, that take the sub-models to state s_i' and s_j', then they can move simultaneously to state s_i' and s_j'. The rate at which this joint transition occurs is decided by the active sub-model that can be S_i or S_j. We express such a cooperation between S_i and S_j, with S_i active, as follows:

$$S_i \underset{(a^+, b^-)}{\overset{y}{\times}} S_j,$$

which means that transitions labelled by a is S_i are active with respect to the cooperation with transitions labelled by b of S_j and originate a models where the joint transitions are labelled with y. The fact that the resulting model is still Markovian should be obvious because the synchronisation inherits the properties derived for that of PEPA. Note that the major difference is that we can synchronise different labels and assign a different name to the resulting transitions. This happens because we would like a modeller to be able to use a library of models whose labels have a local scope. In this way the library items can be created independently and instantiated several times in the same model.

Example 1 (Example of cooperation) *Suppose we would like to model within the presented framework the trivial queueing network depicted in Figure 1 where two identical exponential queues with finite capacities B are composed in tandem. When the first queue if saturated, arrivals are lost. When the second queue is saturated at a job completion of the first queue, the customer is served again (repetitive service blocking). Customers arrive to the first queue according to a Poisson process with rate λ. A queue can be described by three matrices with dimension*

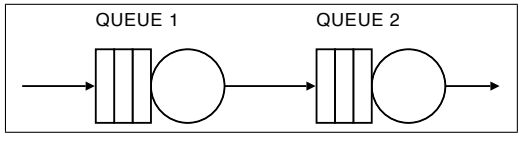

Fig. 1. Tandem of two exponential finite capacity queues.

$B \times B$:

- $\mathbf{M}^{\epsilon} = \mathbf{0}$ *that describes the transitions that cannot synchronise (something like the private part of the model).*
- $\mathbf{M}^{a}$ *where* $\mathbf{M}^{a}[\alpha, \beta] = \lambda$ *if* $\beta = \alpha + 1$ *or* $\mathbf{M}^{a}[\alpha, \beta] = 0$*, otherwise. This matrix describes the transitions corresponding to arrival events.*
- $\mathbf{M}^{d}$*, where* $\mathbf{M}^{d}[\alpha, \beta] = \mu$ *if* $\beta = \alpha - 1$ *or* $\mathbf{M}^{d}[\alpha, \beta] = 0$*, otherwise. This matrix describes the transitions corresponding to job completion events.*

Consider two instances of this model, S_1 and S_2. The tandem network of Figure 1 can be described by the model $S_1 \times^{y}_{(d^+, a^-)} S_2$.

A pairwise cooperation may involve more than one label. In this case we may write:

$$S_1 \underset{(a_1^+, b_1^-)}{\overset{y_1}{\times}} \underset{(a_2^-, b_2^+)}{\overset{y_2}{\times}} S_2$$

to specify that S_1 (S_2) is active on y_1 (y_2) and passive on y_2 (y_1) with transitions labelled a_1 (b_1) and a_2 (b_2), respectively.

The following operator allows us to change all the rates of a matrix labelled by a: $S_1\{a \leftarrow \lambda\}$ is the sub-model S_1 with only matrix $\mathbf{M}^a$ modified so that all its non-zero elements are set to λ.

RCAT. Theorem RCAT [5] gives sufficient conditions to decide and derive the product-form solution of pairwise interactions (possibly involving more than one label) between two sub-models. Le us consider the following synchronisation:

$$S = S_1 \underset{(a_1^*, b_1^*)}{\overset{y_1}{\times}} \dots \underset{(a_T^*, b_T^*)}{\overset{y_T}{\times}} S_2,$$

where symbol $*$ stands either for a $+$ or a $-$. The conditions to apply RCAT are:

1. If a_t is active (passive) in S_1 and b_t is passive (active) in S_2 then $M_1^{a_t}[\cdot, z]$ ($M_1^{a_t}[z, \cdot]$) has exactly one non-zero element for every $1 \leq z \leq Z_1$, and $M_2^{b_t}[z, \cdot]$ ($M_2^{b_t}[\cdot, z]$) has exactly one non-zero element for every $1 \leq z \leq Z_2$[1].
2. Suppose that for every pair (a_t^+, b_t^-) ((a_t^-, b_t^+)) we know a value β_t (α_t) such that:

$$\begin{aligned} S_1' &= S_1\{a_t \leftarrow \alpha_t\} && \text{for all } a_t \text{ passive in the cooperation} \\ S_2' &= S_2\{b_t \leftarrow \beta_t\} && \text{for all } b_t \text{ passive in the cooperation} \end{aligned}$$

   and given an active label a_t (b_t) in S_1 (S_2) all the transitions with that label have the same reversed rate β_t (α_t).

If these conditions are satisfied, then the stationary distribution $\boldsymbol{\pi}$ of S is $\boldsymbol{\pi} \propto \boldsymbol{\pi}_1 \boldsymbol{\pi}_2$ (for each positive recurrent state).

Basically, the first condition says that every state of a model which is passive (active) with respect to a label must have one outgoing (incoming) transition with that label. To understand the second condition, suppose that (a_t^+, b_t^-) is a pair in the synchronisation between S_1 and S_2. Then, we must determine a rate β_t to assign to all the transitions labelled by b_t that is also the constant reversed rate of the active transitions a_t in S_1. Note that, in general, this task is not easy, and is shown to be equivalent to the solution of the traffic equations in Jackson networks and G-networks. The algorithm proposed in [10] aims to give an iterative, numerical and efficient way to perform this computation.

[1] $M_1^{a_t}[z, \cdot]$ represents the z-th row vector of the matrix, and analogously $M_2^{a_t}[\cdot, z]$ represents the z-th column vector.

Although it is out of the scope of this paper discussing the modelling implications of RCAT conditions, it is worth pointing out that several works in literature have proved that this result has not only a theoretical relevance but can be actually used to characterise the product-form solutions for models that may be used for practical case-studies.

The underlying algorithm. The algorithm that underlies our tool has been presented in [10]. It takes the matrices that describe models $S_1, \ldots, S_N$ and the synchronisations as input, and computes as output a boolean value which is true if a product-form has been identified, false otherwise. In case of product-form, the unnormalised stationary distribution is output. In its simplest formulation (two submodels and without optimisations) it can be summarised in the following steps:

1. Generate randomly $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$
2. Compute the reversed rates of the active transitions using Equation (1)
3. Use the mean of the reversed rates for each label to set the rates of the corresponding passive transitions. For instance let a be active for S_1 and b passive for S_2. Then let x be the mean of the reversed rates of the non-zero elements in $\mathbf{M}_1^a$. $\mathbf{M}_2^b$ is updated by setting all the non-zero elements to x
4. Compute $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ as solution of the GBEs of the underlying CTMCs of S_1 and S_2
5. Are the reversed rates of the transitions constant for each active label?
   - **true** $\Rightarrow$ product-form found and the stationary distribution is $\boldsymbol{\pi} \propto \boldsymbol{\pi}_1\boldsymbol{\pi}_2$ and terminate.
   - **false** and the maximum number of iterations has been reached $\Rightarrow$ product-form not found and terminate.
   - **false** and the maximum number of iterations has not been reached $\Rightarrow$ go to step 3

The algorithm is extended in order to include the possibility of multiple pairwise synchronisations (as proposed in [7]) and several optimisations: an efficient way to define an order in the solution of the sub-models (based on Tarjan's algorithm [12]), a parallel implementation, and a special technique to deal with self-loops.

3. Tool

In this section we describe some salient characteristics of the proposed tool. First, we explain our approach in the specification of the interactions between

the sub-models. Then, we describe the client-server architecture and illustrate its strengths.

3.1. Specifying the interactions

In order to better understand the motivations of this section, let us consider again the model depicted by Figure 1 with a variation, i.e., after a job completion at the first station the customer may exit the system with probability p or go to the second station with probability $1 - p$, as depicted by Figure 2. We note that the

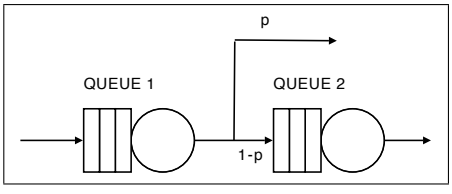

Fig. 2: Probabilistic routing in the model of Figure 1.

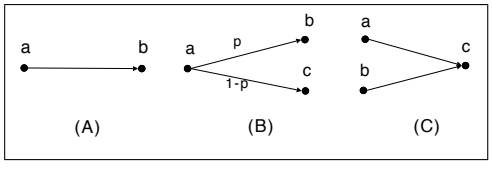

Fig. 3. Types of connections between labels.

processes underlying the first and second queue are different, and we could not use two instances of the same model anymore. Indeed, in the first queue the transition corresponding to a job completion from state j to state $j - 1$ must be split in two: one synchronising with the arrivals in the second queue with rate $(1-p)\mu_1$ and one without synchronisation with rate $p\mu_1$. We decided that this splitting of transitions should be done automatically by our tool, so that the library of sub-models can be defined without any knowledge about the future usage and connections.

From the modeller point of view, a sub-model is seen just as a black box where the labels are exported, i.e., a model specification consists of a set of connections about instances of modules. The simplest possible connection between two labels is that depicted by Figure 3-(A). Note that in this Figure we use a graphical representation of the connections which is coherent with the MSI that we developed, however different approaches are possible (such as a PEPA-like syntax). Figure 3-(A) illustrates a label a of a sub-model that interacts with a label b of another sub-model. The arrow is oriented, meaning that a is active and b is passive. This specification of synchronisation does not require any modification to the structure of the active or passive sub-models. Let us now consider Figure 3-(B). In this case the active action a of one sub-model synchronises with passive actions b (with probability p) or c (with probability $1 - p$) of other sub-models. In this case, we need to alter the structure of the active model. Recall that matrix $\mathbf{M}^a$ represents the transitions labelled by a. Then we define $\mathbf{M}^{a'} = p\mathbf{M}^a$ and $\mathbf{M}^{a''} = (1 - p)\mathbf{M}^a$. Hence, in the active sub-model, matrices $\mathbf{M}^{a'}$ and $\mathbf{M}^{a''}$ replace matrix $\mathbf{M}^a$. Note that this tech-

nique can be applied to an arbitrary number of probabilistic synchronisations under the obvious constraint that the synchronisation probabilities must sum to a value which is less or equal to 1. Suppose that the sum of the probabilities $p_1, \ldots, p_K$ is $p_t < 1$ (see Figure 2 for an example). In this case we have $\mathbf{M}^{a^k} = p_k \mathbf{M}^a$ for $k = 1, \ldots K$, and $\mathbf{M}^\epsilon$ (which is always present in a model description an represents the transition that cannot synchronise) is replaced by $\mathbf{M}^\epsilon + \mathbf{M}^a(1 - p_t)$. We use the notation $S_1 \times^{y,p}_{(a^+,b^-)} S_2$ to denote that a in S_1 is active in the synchronisation with b in S_2, and the synchronisation is called y and occurs with probability p. The latter case is depicted by Figure 3-(C) where two active labels a and b (that can belong to the same or different sub-models) synchronise with the same passive label c. In this case we simply replace matrix $\mathbf{M}^c$ of the passive model with two matrices $\mathbf{M}^{c'}$ and $\mathbf{M}^{c''}$ identical to the former (we do not need to modify the rates since they are replaced with the rate of the corresponding active transitions).

Example 2 (Application to the model of Figure 2) *Let us show how we model the tandem of exponential queues with finite capacities B depicted by Figure 2. We still consider two identical instances of the same sub-model which is described in Example 1. The user specifies in some way the interactions. The model corresponding to the second queue does not change, while that corresponding to the first queue becomes the following:*

- $\mathbf{M}^\epsilon = p\mathbf{M}^d$ *that describes the transitions that cannot synchronise*
- $\mathbf{M}^a$,
- $\mathbf{M}^{d'} = (1 - p)\mathbf{M}^d$,

where $\mathbf{M}^a$ and $\mathbf{M}^d$ are the matrices defined in Example 1.

It may be worth pointing out some notes about this approach to the specification of the sub-model interactions: 1) Its scope is to allow the specification of a model despite to the synchronisations it will be involved in. For instance, if we have a model of a simple exponential queue, we can straightforwardly define a Jackson queueing network with probabilistic routing by simply instantiating several copies of the same model. Moreover, connections have a simple and intuitive meaning. 2) When an active label is split the infinitesimal generator of the sub-model does not change, i.e., its stationary distribution does not change. Moreover, if the revered rates of the transitions corresponding to active label a are constant in the original model, then also the transitions corresponding to a split label associated with a have constant reversed rates. 3) The effects of the replication of passive label matrices on the algorithmic analysis of the product-form is that the rate associated with the passive transition is the sum of the (constant) reversed rates of every associated

active transition. 4) Specifying pairwise interactions where the same label is simultaneously active and passive with respect to two or more synchronisations is not allowed. This characteristic is inherited from the semantics of the cooperation given in the theoretical paper which this tool is based on.

3.2. Client-server architecture

The tool consists of two parts: the analyser (the server) and the MSI (the client). The idea is that although we propose a graphical client side that exploits the strengths of our modular approach and the specification of the module synchronisation, one could write his own MSI in order to make it compatible with the favourite formalism.

The server opens an independent session for each MSI connected. It provides a character interface which is used by the MSI to: 1) Create/Import a sub-model, 2) Specify a synchronisation between two labels of two sub-models, 3) Require the solution of a model given a precision and a maximum number of iterations. In the first and second case the server just answers the client if the required operation has been executed correctly, while the latter one returns the following data: 1) A flag that specifies if the product-form has been identified, 2) The steady-state probabilities of each sub-model, 3) The reversed rates of all the active transitions. Note that knowing the reversed rates of the active transitions means knowing the solution of the system of traffic equations. In [10] it is proved that when the algorithm analyses a Jackson queueing network, the iterations are equivalent to the Jacobi scheme for the solution of the model linear system of traffic equations. Similarly, when it is applied to a G-network it is equivalent to iterative scheme proposed by Gelenbe et al. for the solution of the non-linear system of traffic equations [3].

3.3. Use cases

In this section we illustrate some examples of case studies. We give a description of the model which is independent of the MSI that will be adopted. We just focus our attention on three well-known results about product-form that have been widely used in the communication networks performance evaluation analysis, although several other instances may be easily produced.

Jackson networks. Jackson networks are easy to study because they are characterised by a linear system of traffic equations. However, in our framework, they require some attention since each sub-model (i.e., each exponential queue) has an infinite state space. In many cases in which the sub-model is known to have a geometric steady-state distribution and the transitions between states n and $n+1$ are the same for all $n \geq 0$, we can simply represent the sub-model using just a pair of

adjacent states [10]. We apply this technique to reduce the infinite state space of a sub-model we must disable the RCAT structural check (Condition 1) because some transitions that are present in the real model, are omitted in the finite one. Figure 4 shows the truncation of an exponential queue. If the synchronisations it will be involved in impose a to be passive and d to be active, we note that Condition 1 of RCAT is satisfied for the infinite model but is not satisfied for the reduced one (e.g., state $n+1$ does not have any incoming active transition or outgoing passive transition). Nevertheless, the algorithm may still be applied, so the structural check for this model must be disabled.

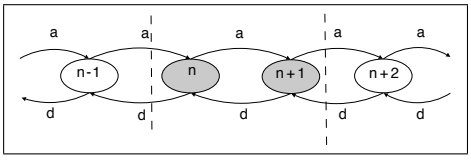

Fig. 4: Truncation of the birth and death process underlying an exponential queue.

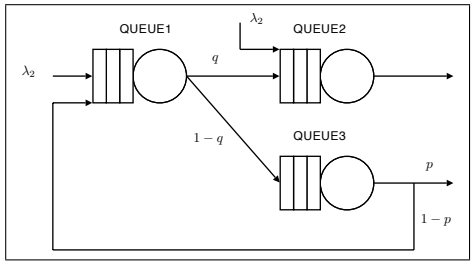

Fig. 5. Jackson network of Example 3.

Example 3 (Jackson network) *Consider the Jackson network depicted by Figure 5. A sub-model of an exponential queue consists of three matrices (states are in the order n and $n+1$):*

$$\mathbf{M}^{\epsilon} = \mathbf{0} \quad \mathbf{M}^{a} = \begin{bmatrix} 0 & \lambda \\ 0 & 0 \end{bmatrix} \mathbf{M}^{d} = \begin{bmatrix} 0 & 0 \\ \mu & 0 \end{bmatrix}$$

We also use a single-state sub-model to represent the external Poisson arrivals with $\mathbf{M}^{\epsilon} = \mathbf{0}$ and $\mathbf{M}^{a} = [\lambda]$. Supposing the service rates for Queue 1, 2 and 3 are μ_1, μ_2 and μ_3, let S be the library model for the queue and A that for the external arrivals, then we have:

$$S_i = S\{d \leftarrow \mu_i\}\ i = 1,2,3 \qquad A_t = A\{a \leftarrow \lambda_1 + \lambda_2\}$$

The synchronisations are specified with the following commands to the server:

$$A_t \underset{(a^+,a^-)}{\overset{y_1,\lambda_1/(\lambda_1+\lambda_2)}{\times}} S_1, \quad A_t \underset{(a^+,a^-)}{\overset{y_2,\lambda_2/(\lambda_1+\lambda_2)}{\times}} S_2, \quad S_1 \underset{(d^+,a^-)}{\overset{y_3,q}{\times}} S_2, \quad S_1 \underset{(d^+,a^-)}{\overset{y_4,1-q}{\times}} \underset{(a^-,d^+)}{\overset{y_5,1-p}{\times}} S_3.$$

G-networks. G-networks can be modelled in our frameworks in an analogous way of that presented for Jackson networks. Note that although the models are

different both in the specification and in the analysis, our tool treats them uniformly by exploiting the RCAT theoretical result. The truncation mechanism presented for Jackson queueing centers is applied also for G-queues which consist of three matrices: the epsilon, A representing the transitions for positive customer arrivals, d representing the transitions for the job completion and, finally, a representing the transitions for the negative customer arrivals:

$$\mathbf{M}^{\epsilon} = \mathbf{0}, \quad \mathbf{M}^{A} = \begin{bmatrix} 0 & \lambda_A \\ 0 & 0 \end{bmatrix}, \quad \mathbf{M}^{d} = \begin{bmatrix} 0 & 0 \\ \mu & 0 \end{bmatrix}, \quad \mathbf{M}^{a} = \begin{bmatrix} 0 & 0 \\ \lambda_a & 0 \end{bmatrix}.$$

Finite capacities queueing networks with blocking. Akyildiz's product-form queueing networks with blocking [1] can be analysed by this tool. Finite capacity queues have a finite state space so the truncation mechanism is not needed. In order to reproduce Akyldiz's results on the reversible routing it suffices to synchronise a departure label of a queue with an arrival label of another queue considering the former passive and the latter active.

4. MSI implementation example

In this Section we illustrate a possible implementation of the MSI. Recall that the tool client-server architecture allows for different MSIs according to the modeller's needs. We show a general-purpose MSI that is independent of the formalism used by the modeller. As an example we model the Jackson network depicted by Figure 5. Each sub-model is represented by a coloured circle and arcs represent the synchronisations. Each object, circle or arc, has a name. In the former case it is the sub-model name, in the latter it has the form $y(a, b)$ that stands for $S_1 \times^{y}_{(a^+,b^-)} S_2$, where S_1 is the sub-model from which the arc outgoes from, and S_2 is the destination sub-model. A screen-shot is shown in Figure 6. By clicking on a sub-model circle a window appears with its description in matrix-form and the user is allowed to perform changes (add or remove transitions or change rates). When a arc is set between two sub-model the window shown in Figure 7 appears (the required parameters should be clear). Note that, although one could point out that a standard tool for the analysis of Jackson networks may present a more intuitive interface, we would like to remark that this is the same interface we would use for any stochastic model that can be solved by the algorithm presented in [10]. However, one could also decide to extend the MSI in order to be able to associate a specific symbol to some sub-models of the library, but this is out of the scope of this presentation.

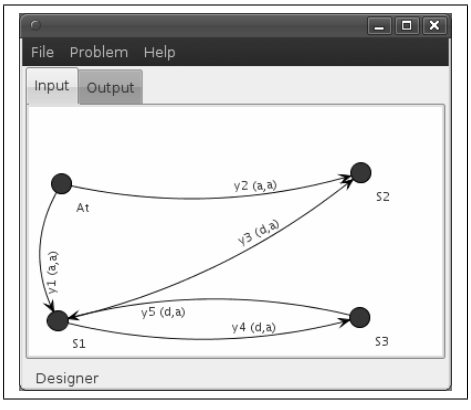

Fig. 6: Screen-shot of the model corresponding to the Jackson network of Figure 5.

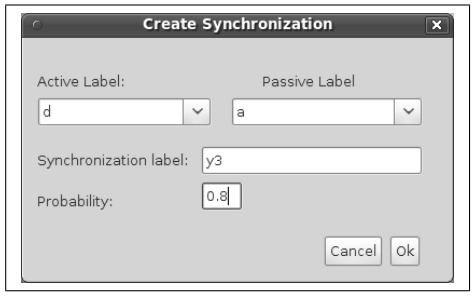

Fig. 7: Screen-shot of the window for the synchronisation details.

5. Final remarks

We have presented a tool that we are developing for the analysis of product-form models. It exploits some new results that appeared in product-form model theory and the algorithm presented in [10]. It has proved to be able to identify and compute several product-form results based on pairwise synchronisations, such as Jackson networks, G-networks, Akyildiz's results about product-form networks with blocking and other that have been described in [10]. Current research has three objectives: 1) allow for the specification of models with multiple incoming active transitions, exploiting the result presented in [11], 2) allow for the specification of models with multiple outgoing passive transitions, and 3) allow for the specification of models with regular but infinite structure. The last goal seems to be the hardest one. Indeed, an approximation is needed to truncate the model and we would like it to be decided dynamically in order to produce results which are correct within a specified bound.

References

[1] I. F. Akyildiz. Exact analysis of queueing networks with rejection blocking. In H. G. Perros and T. Atliok, editors, *Proc. of the 1st Internat. Workshop on Queueing Networks with Blocking*, pages 19–29, North-Holland, Amsterdam, 1989.

[2] 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.

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

[4] P. G. Harrison. Reversed processes, product forms, non-product forms and a new proof of the BCMP theorem. In *Int. Conf. on the Numerical Solution of Markov Chains (NSMC 2003), Urbana IL, USA, September 2-5 2003*, pages 289–304, September 2003.

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

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

[7] P. G. Harrison and T. T. Lee. Separable equilibrium state probabilities via time reversal in markovian process algebra. *Theoretical Computer Science*, 346(1):161–182, 2005.

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

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

[10] A. Marin and S. Rota Bulò. A general algorithm to compute the steady-state solution of product-form cooperating Markov chains. In *Proc. of MASCOTS 2009*, pages 515–524, London, UK, September 2009.

[11] A. Marin and M. G. Vigliotti. A general result for deriving product-form solutions of markovian models. In *Proc. of First Joint WOSP/SIPEW Int. Conf. on Perf. Eng.*, San Josè, CA, USA, To appear.

[12] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. on Computing*, 1(2):146–160, 1972.