Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 296 (2013) 61–77

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Exploiting product forms solution techniques in multiformalism modeling

Enrico Barbierato[1]

DI, Università degli Studi di Torino
corso Svizzera, 185, 10129 Torino (Italy)

Gian-Luca Dei Rossi[2]

DAIS, Università Ca' Foscari Venezia
via Torino, 155, 30172 Mestre Venezia (Italy)

Marco Gribaudo[3]

DEI, Politecnico di Milano,
via Ponzio 34/5, 20133 Milano (Italy)

Mauro Iacono[4]

DSP, Seconda Università degli Studi di Napoli
viale Ellittico, 81100 Caserta (Italy)

Andrea Marin[5]

DAIS, Università Ca' Foscari Venezia
via Torino, 155, 30172 Mestre Venezia (Italy)

Abstract

Multiformalism modeling has shown to be a valuable technique to cope with the complexity of the constraints that apply to specifications of computer-based systems state of the art. Multiformalism techniques help modelers and designers by providing a more (natural and) convenient approach in the specification process and in analysis of performance. Although their application does not necessarily provide an advantage in the solutions of the models, this paper shows how a compositional multiformalism modeling approach can leverage the power of product-form solutions to offer both efficient solution and specification of models for complex systems.

Keywords: Multiformalism modeling, product-form solution, compositionality, performance evaluation

1571-0661 © 2013 Elsevier B.V. Open access under CC BY-NC-ND license.
http://dx.doi.org/10.1016/j.entcs.2013.07.005

1 Introduction

Computer based systems that daily serve most human activities are characterized by an increasing level of complexity, a term that can be interpreted in different ways. Complexity can be found in the concurrent requirements these systems have to satisfy while functioning (e.g. for critical systems, that have temporal, safety, dependability and performance constraints), from the number of their different articulations, or from their extension. Furthermore, complexity can be the result of a substanding logic of design by composition of existing subsystems. In all these cases, designers face the challenge by exploiting models to define or understand the system's characteristics.

Multiformalism modeling techniques allow modelers to choose the most suitable modeling formalism to dominate different aspects, or parts, of the overall system model, thus providing more manageable and understandable support. This natural modularity supports a component-based divide-et-impera approach, but does not imply a more efficient model solution.

State space based model analysis techniques suffer from the so-called state space explosion problem. Modularity has been exploited differently in literature to confront this problem, suggesting that a proper management of submodels composition can generate solutions efficiently. Simulation can play the role of the silver bullet at the cost of possibly longer computing time. In a number of cases it is possible to generate efficient analytical solutions either by relying on specific characteristics of the composition structure, or by introducing proper artifices. This is the case for product-form solution techniques, that offer a number of known cases in which the analysis of a complex model can be obtained by analyzing its components.

In this paper we show how product-form solution theory easily couples with multiformalism compositional modeling techniques, to obtain a modeling and analysis framework that offers modeling flexibility and efficient solutions. The contribution of this paper is founded on the design and implementation of an extensible modeling and solution framework, supported by a tool solving multiformalism Markovian models with a threefold solution mechanism. The tool automatically verifies and performs a product-form solution. If this is not available it provides a state space based analytical solution or a simulation as final backup tool. The research extends the SIMTHESys framework and

[1] enrico.barbierato@mfn.unipmn.it
[2] deirossi@dais.unive.it
[3] gribaudo@elet.polimi.it
[4] mauro.iacono@unina2.it
[5] marin@dais.unive.it

a tool for product-form solutions, presented in [1], in order to encompass ERCAT/MARCAT product form models. To date, it appears there is no other similar, tool-supported approach in the current literature.

In the rest of the paper, Section 2 presents the approach introduced in this work, Section 3 shows a case study, and Section 4 presents some related works about multiformalism modeling and product-form solution. Conclusions are presented in Section 5.

2 From multiformalism models to product-form solutions

The approach proposed in this paper leverages the existing modularity in multiformalism models and the possibilities of the SIMTHESys framework to exploit Markov chains product-form solutions detection in a multiformalism solution process. The SIMTHESys framework supports the design and development of user-defined formalisms, for which a proper (multi)formalism solver is automatically generated. The framework defines a formalisms design technique, based on metamodeling and on a mechanism to specify the structural and dynamic characteristics of every element of a formalism [4,6,5]. Solver generation is based on the analysis of the formalism description and on a set of basic *solving engines*, where basic should be intended as able to implement fundamental solving techniques. In this paper, the Markov chain solving engines (analytical and simulative) are used, together with a new solving engine that benefits from the INAP algorithm and the MARCAT Theorem.

The set of specifications that a (multi)formalism should satisfy to apply MARCAT to the model will be presented by developing a proper example high level formalisms. We introduce a new *formalism family*, that is a new set of features that a formalism should have to be able to be solved using both standard Markov chain solving engines, and with the MARCAT product-form computation. The set of features that characterize this family of formalism is inspired by the features available in Continuous Time Stochastic Automata Networks with Master/Slave synchronization [10]. This new formalism family will be called *Labelled Exponential Events Formalisms*. Formalisms belonging to this family can be conveniently thought as Labelled Exponential Automata.

The approach is implemented by using SIMTHESys*ER*, the SIMTHESys framework solver generation tool, and INAP, a tool for automatic detection of many product-form solutions based on the MARCAT algorithm. Within this approach, the proposed formalism family allows i) the SIMTHESys framework to generate product-form solution based optimized solvers and ii) INAP to support high level formalisms, extending its application field to more complex models. Moreover, a further result of the integration of the two tools is the enrichment of the benefits of MARCAT with the automatic verification of its hypotheses by using SIMTHESysER state space generation logic. This provides a tool that allows the identification of a greater number of known product-forms without the need of checking them one by one.

The main limit of the approach is that it is only applicable to models that cooperate pairwise. As a consequence more complex product-forms such as those based on instantaneous signal propagation [16] are not considered. When product-forms are not applicable, models are solved by the generated solver using the general analytic approach, or by simulation otherwise.

2.1 *Deciding and computing the product-form solution*

In this section we outline the main steps that are performed to decide if a model admits a MARCAT based product-form solution and, in case of positive answer, to compute it.

A model belonging to the *Labelled Exponential Events Formalism* considered in this work can be considered as a tuple (C, L), where $C = \{c_1 \ldots c_N\}$ is the set of sub-model components, and $L = \{l_1 \ldots l_K\}$ is a set of labels. Sub-models can be defined in any formalisms that belongs to the considered family. In particular, each sub-model component c_i is characterized by a set of variables $V_i = \{v_{i,1} \ldots v_{i,n_i}\}$ that define its state, and a set of events $E_i = \{e_{i,1} \ldots e_{i,m_i}\}$ that governs the transition from one state to another. Each state $\mathcal{S}_i$ is uniquely identified by the value of its variables: if two states have the same values for all the variables of the model, they are the same state. The information contained in a state $\mathcal{S}_i$ is capable of completely defining the possible events $e_i \in E_i$ that can cause a state change, and their temporal behavior. Events that triggers a change of state can occur either locally after an exponentially distributed time, or globally due to a synchronization. Exponential events e_i are characterized by a rate $\lambda(e_i, \mathcal{S}_i) \to \mathcal{R}^+$: as soon as the system enters state $\mathcal{S}_i$, event e_i will occur after an exponentially distributed random amount of time, with rate $\lambda(e_i, \mathcal{S}_i)$. If more than one event can occur in the same state $\mathcal{S}_i$, then *race policy* is used to choose between the two. Synchronization is performed following a *Master/Slave* paradigm over a label. *Active events* have an exponential time associated and a label: $\mu(e_i, \mathcal{S}_i) \to \mathcal{R}^+ \times L$. *Passive events* have only a label associated with them: $\gamma(e_i, \mathcal{S}_i) \to L$. When an *Active event* is available in a state, it is triggered after the corresponding exponentially distributed random amount of time. During the execution of an Active event, the associated label l_j is generated. Passive events are instead immediately executed as soon as the corresponding label l_j is generated by an active event in another sub-model: this allows synchronization among the sub-models. All the events moves the system into another state, by appropriately changing the values of the variables that defines the states of the various sub-models involved by the events (that is, of the considered sub-model for local events, or of the sub-models where the Active and Passive events belong), that is $\mathcal{S}'_i = f(e_i, \mathcal{S}_i) \quad \forall s_i \in S$.

In order to keep the paper self-contained we briefly present MARCAT in case of a pair of cooperating models $P \equiv c_1$ and $Q \equiv c_2$ (in this case the theorem is usually known as ERCAT). Henceforth we denote by $P \otimes Q$ as the joint model.

Assume that for each label a we know a positive real value x_a, and let P' (Q') be the process P (Q) in which all the passive transitions labelled by $a \in \mathcal{P}_P$ ($a \in \mathcal{P}_Q$) take x_a as a rate. Clearly, P' and Q' are now independent and, if the underlying CTMCs are ergodic (or have an ergodic subset of states) we can compute their steady-state distributions $\pi_{P'}$ and $\pi_{Q'}$. According to MARCAT, a product-form solution exists if for each ergodic state of (p, q) of $P \otimes Q$ we have that its steady-state probability $\pi(p, q)$ is: $\pi(p, q) \propto \pi_{P'}(p)\pi_{Q'}(q)$, where the direct proportionality symbol is an equality if the ergodic states of $P \otimes Q$ are the Cartesian product of the ergodic states of P' and Q'. The following sets play a pivotal role in the application of MARCAT, we recall that (p, q) is an ergodic state of $P \otimes Q$:

(i) $\mathcal{P}^{(p,q)\rightarrow}$ is the set of passive labels outgoing from (p, q)
(ii) $\mathcal{A}^{(p,q)\rightarrow}$ is the set of active labels outgoing from (p, q)
(iii) $\mathcal{P}^{(p,q)\leftarrow}$ is the set of passive labels entering into (p, q)
(iv) $\mathcal{A}^{(p,q)\leftarrow}$ is the set of active labels entering into (p, q)

MARCAT gives a purely algorithmic way to decide the existence of a product-form solution in the cooperation of two processes, and in case of existence it states its expression.

Theorem 2.1 (MARCAT [18]) *Let P and Q be two cooperating processes on set of labels $\mathcal{L}$ and assume that the following conditions are satisfied:*

- *For each label $a \in \mathcal{A}_P$ ($a \in \mathcal{A}_Q$), x_a is the reversed rate of all the transitions labelled by a in P' (Q');*
- *For each joint state (p, q) the following rate equation holds:*

$$\sum_{a \in \mathcal{P}^{(p,q)\rightarrow}} x_a - \sum_{a \in \mathcal{A}^{(p,q)\leftarrow}} x_a = \sum_{a \in (\mathcal{P}^{(p,q)\leftarrow} \setminus \mathcal{A}^{(p,q)\leftarrow})} \overline{\beta}_a(p, q) - \sum_{a \in (\mathcal{A}^{(p,q)\rightarrow} \setminus \mathcal{P}^{(p,q)\rightarrow})} \alpha_a(p, q) \quad (1)$$

where $\alpha_a(p, q)$ is the rate of the transition outgoing from state (p, q) labelled by a while $\overline{\beta}_a^{(p,q)}$ is the reversed rate of the transition entering into (p, q) labelled by a;

then for each ergodic state (p, q) of the joint process the following relation *holds:* $\pi(p,q) \propto \pi_{P'}(p)\pi_{Q'}(q)$

In practice, the application of MARCAT requires to address the following problems:

- Determining the values x_a for each synchronising label;
- Checking Equation (1) for each ergodic state of the joint model. In particular the computation of the values $\overline{\beta}_a$ requires the knowledge of the steady-sate distribution of the component in which a is passive;

Here, the first problem is solved by applying INAP [21] assuming that a product-form exists and we check Equation (1) *a posteriori*. INAP output contains both the value of the reversed rates and the steady-state distribution of each component in isolation. This information are therefore used to verify MARCAT rate condition (1) within a given numerical precision.

2.2 The formalisms

SIMTHESys is a flexible tool that allows for the definition of various modelling formalisms according to the users' needs and expertise. Here, we present only the two formalisms that are used in the case-study of Section 3. The first is a variant of open, finite capacity, blocking, repetitive services queuing networks, as already seen; the second is a variant of stochastic Petri nets: both the formalisms are enriched by including elements that implement the given specification. The elements of both formalisms are in Fig. 1.

For what concerns queueing networks, besides the *queue* element, characterized by its capacity, its length and its service rate, the queuing network formalism has six other node elements: the *source* element generates requests and is characterized by its rate; the *active source* element generates requests at its rate and exports a label; the *passive source* element generates requests at a rate depending on the bound label; the *sink* element consumes requests; the *active sink* element consumes requests and exports a label; the *passive sink* element consumes requests depending on the bound label. The formalism also has the *arc* element (that routes requests between sources, queues and sinks), the *test arc* element (that checks a condition over a node element to enable another node element) and the *inhibitor arc* element (that inhibits a node element according to the state of another node element).

The state of the sub-models is simply defined by the occupancy of the queues. The local events are used to account for ends of service in a queue, and for an arrival to the system. Active events are used to consider departures from the system from sinks that have an associated label (active sinks), and arrivals to the system coming from an active source. Passive events are used to model both passive arrivals and passive departures: the former corresponds to arrivals in the system triggered by the firing of an active transition belonging to a different sub-model. The latter to customer immediately leaving a system due to an interrupt coming from a different source. Moreover, queues, sources and sinks can be blocked when the total queue length of neighbour stations (but always in the same sub-mode) become larger or smaller than a given trheshold thanks to *Test* and *Inhibitor Arcs*.

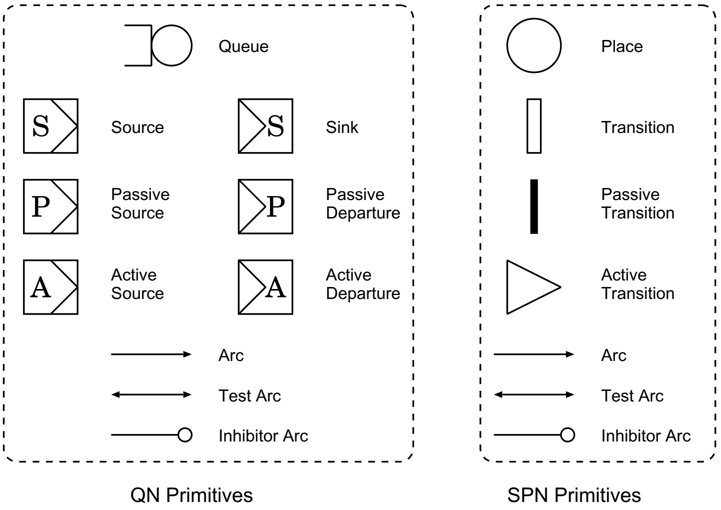


Fig. 1. Formalisms elements

The queueing network semantic is implemented in this way. First enabled events are determened by looking at the size of the queue of the various stations. All *Source* and *Active Source* primitives always generate an event (a *local* event for the former, and an *active* event for the latter). *Passive Source*s generate *passive* events. *Queue* primitives generate events if the corresponding queue has at least one customer. The event is *local* if the destination of the queue is another queue or a *Sink*, it is *active* if the destination is an *Active Departure*, and *passive* for *Passive Departure*s.

The stochastic Petri nets formalism has four node elements: the *Place* element, characterized by its marking; the *Transition* element, characterized by its rate; the *Active Transition* element, characterized by its rate and the label it exports; the *Passive Transition* element, the rate of which depends on the bound label. The formalism has arc elements analogously to the queuing network formalism. *Active Transitions* behave exactly has standard SPNs timed transition, but they also expose a label when they fire. *Passive Transitions* instead, are completely governed by the *active* events happening in other sub-models. Petri Nets are implemented by generating an event for each enabled transition. In particular, the type of the event generated corresponds to the type of transition: *local* events for standard *Transition*s, *active* events for *Active Transition*s and *passive* events for *Passive Transition*s.

Note that in our approach labels, active and passive events, are used to provide a formalism independent cooperation scheme that allows the synchronization among sub-models, specified using different modeling languages. The sub-models are connected using cooperation arcs in the enclosing main model. Each cooperation arc has associated a set of labels and it is directed from a source sub-model to a destination sub-model: whenever an active events, with the corresponding label, happens in the source sub-model, it triggers enabled passive events, associated with the same label, in the destination sub-model.

3 Case study

In this section we introduce an example that is analysed with the proposed tool. Notice that the system has been chosen to spot the original features of the technique described in this paper.

A data stream processing system for the detection and monitoring of seismic phenomena is structured in two main subsystem: a pre-processing subsystem and a critical detection subsystem. The first is composed of two stages, each of which processes batches with a temporally variable computation, whose duration is exponentially distributed. Both the stages receive different jobs to be processed, with an exponentially distributed rate. In addition, the second receives part of the output batches of the first and some of the jobs processed by the second are sent back to be processed by the first, while the others constitute the output of the subsystem. Each of the stages can buffer a number of timestamped jobs. A stage that is not allowed to dispatch a job to the next one has to reprocess the data, to account for the time elapsed.

The second subsystem processes the output jobs of the first one, together with additional jobs that are sent to it. To protect this critical subsystem, a protection mechanism can shut it temporarily off when the number of jobs is greater than a fixed threshold and an overload condition is detected. In such cases, the arrival of new jobs is blocked and losses occur. The protection mechanism is supplied by a proper subsystem.

3.1 Overall model description

A high-level, conceptual model of the system is given in Fig. 2. The first subsystem can be described by a two nodes queueing network with finite capacity and repetitive service blocking of the type studied in [2]. The second subsystem can be described by a finite capacity queue controlled by an ON/OFF switching element when the number of jobs in the queue is not less than the threshold value m. In the figure, λ_1, λ_2 and λ_3 denote the job arrival rates; B_1, B_2 and B_3 are the capacity of the queues; μ_1, μ_2 and μ_3 are the job service rates; p is the probability of forwarding a processed job from the first to the second queue; q is the probability of forwarding a processed batch from the second to the first queue; n is the number of jobs in the third queue; γ represents the trigger by which ON/OFF blocks forwarding from the first to the second subsystem, given that $n \geq m$.

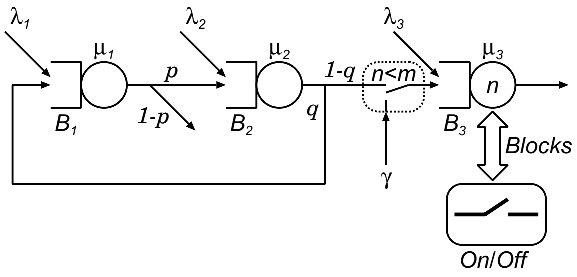


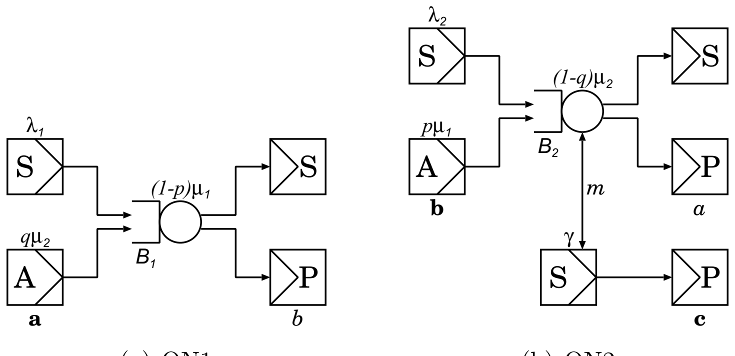
Fig. 2. Overall model description

3.2 Model specification

The first subsystem is modeled by a queueing network with finite capacity and repetitive service blocking. With reference to Fig. 2, the first stage is described by submodel QN1 in Fig. 3. The source with rate λ_1 represents the external arrivals, the active source with rate $q\mu_2$ and exporting label a represents the jobs arriving from the second stage (μ_2 is the second stage's service rate and q the probability that a job enters the first stage after being served at the second). The sink represents the jobs that leave the system, the passive departure importing label b represents the jobs available for entering the second stage, and the queue with capacity B_1 and rate $(1-p)\mu_1$ represents the processing unit. The queue rate expression results from the fact that the only rate that can be defined here is the rate of departures by the sink, because the rate of departures for the second stage depends on the behavior of the second stage (see [2]).

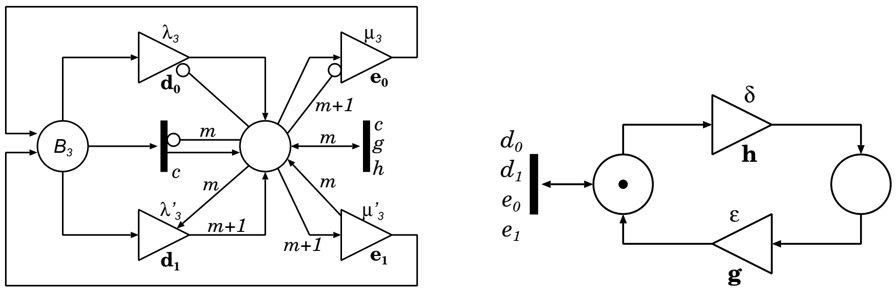
The second stage is described by submodel QN2 in Fig. 3. Similarly as seen for the first stage, the source with rate λ_2 represents the external arrivals, the active source with rate $p\mu_1$ and exporting label b represents the jobs arriving from the first stage, the sink represents the jobs that leave the system, the passive departure importing label a represents the jobs available for the first stage, and the queue with capacity B_2 and rate $(1-q)\mu_2$ represents the processing unit. Moreover, the second stage also has a source with rate γ, enabled by the queue by the test arc that checks if it contains less than m pending requests, and a passive departure, that imports the label c.

The second subsystem is modeled by two stochastic Petri nets submodels. The third queue in Fig. 2 is described by submodel SP1 in Fig. 4. The active transition with rate λ_3 and exporting label d_0 represents the external arrivals, and is enabled if the left place (initially marked with B_3 tokens to represent



(a) QN1 (b) QN2

Fig. 3. QN submodels



(a) SPN1 (b) SPN2

Fig. 4. SPN submodels

the finite capacity) is marked and if the right place (marked by the same active transition and representing a request under processing) is not marked. The active transition with rate λ_3' and exporting label d_1 represents the external arrivals when the length of the queue is at least m, marks the right place, and is enabled if the left place is marked and if the right place has at least m tokens (thus the queue length is at least m); the passive transition importing label c represents arrivals from the first subsystem, and is enabled when the left place is enabled and the right place is marked with less than m tokens (thus the queue length is less than m); the active transition with rate μ_3 and exporting label e_0 represents the processing of a request when the length of the queue is less than $m+1$. The active transition with rate μ_3' and exporting label e_1 represents the processing of a request when the length of the queue is at least $m+1$. The passive transition importing labels c, g and h authorizes incoming requests from the first subsystem and the ON/OFF mechanism when the length of the queue is at least m.

The ON/OFF submodel is described by two places, representing the ON (left) and the OFF (right) conditions. The active transitions with rate δ and exporting label h and with rate ϵ and exporting label g respectively represent the switch off and the switch on; the passive transition importing labels d_0, d_1, e_0 and e_1 enables the interactions with the previous submodel when the left place is marked.

Fig. 5 shows how submodels are connected by the bridge model. In particular, each submodel is drawn as a rectangle, and cooperation between models is denoted by arcs. Arrows are directed from submodels performing active transitions, to submodels subject to passive events. Active event labels are written in boldface near an `OUT` keyword, while passive event labels in italic near an `IN` keyword.

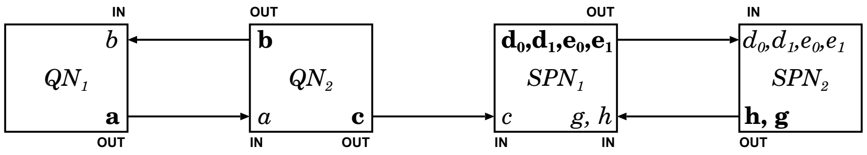


Fig. 5. Submodels composition

3.3 Model analysis and results

The output produced by SIMTHESysER for the model considered in this example is shown in Fig. 6 and is expressed in terms of Labelled Exponential Automata where symbol $\top$ denotes passive transitions. Notice that a brute-force analysis relying on the solution of the system of global balance equation could be unfeasible. Indeed, although finite, the cardinality of the state space grows as $\mathcal{O}(B_1B_2B_3)$, where B_i is the capacity of the i-th queue. Assuming for simplicity that $B_i = B$, the standard solution of the global balance equation system with a Gauss elimination equivalent method has a time complexity of $\mathcal{O}(B^9)$ which could quickly become punitive also for the numerical stability of the involved algorithms. Simulation could be another approach for estimating the model's performance measures. However, we should note that the precision of the simulation estimates strongly depends on the model's parameters. In fact, some events are likely to be rare such as the saturation of the queues or the blocking mechanism implemented for the system protection. As a consequence, time expensive simulations could be required and the validation of the estimates play an important role.

Applying product-form analysis to this model is not trivial. Indeed, none of submodels involved in the cooperation is quasi-reversible, and therefore the triviality of the derivation of the steady-state distribution is avoided. The queueing network with feedback consisting of models $QN1$ and $QN2$ does not admit a Jackson's product-form since the stations have finite capacity. Nevertheless, their rate-dependent product-form relies on the analysis carried out in [2]. The output process of $QN2$ feeds $SPN1$; this process combined with the external arrivals at $SPN1$ satisfies RCAT condition. Finally, the pro-

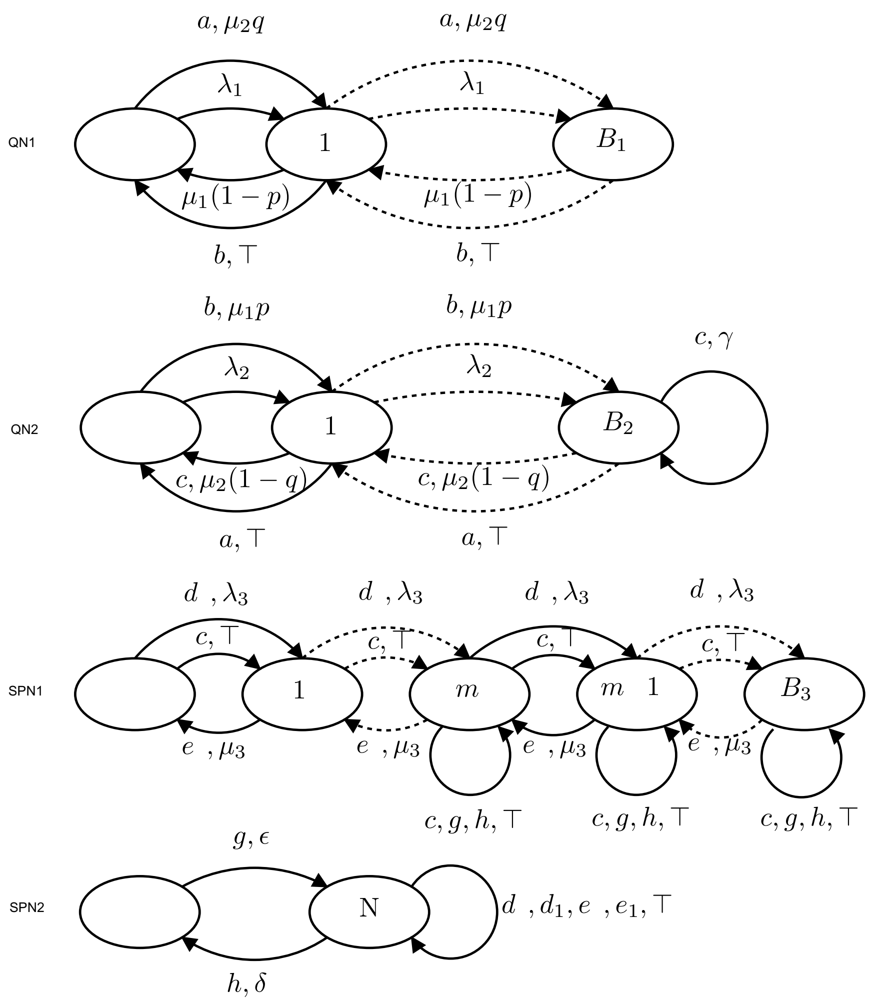


Fig. 6. Labelled Exponential Automata produced by SIMTHESys*ER*

cess that regulates the interaction between the blocking protection mechanism ($SPN2$) and $SPN1$ belongs to the Boucherie's product-form model class [8]. The combination of these types of product-forms have a non-trivial solution given by the following proposition.

Proposition 3.1 *The case-study model has a product-form solution if the following rate-conditions are satisfied:*

$$(1-p)q\lambda_2 = (1-q)p\lambda_1 \tag{2}$$

$$\gamma = \frac{\lambda_1(1-q)(\lambda_2 + p\mu_1)}{\lambda_1(1-q) + (1-p)q\mu_1} \tag{3}$$

Proof. It can be algebraically proved that the set of equations (1) are satisfied if and only if Condition (2) and (3) are satisfied. □

The product-form analytical solution of the case-study model depends on the choice of some rates as illustrated by Proposition 3.1. Although this is somehow disappointing from a modelling point of view, the importance is not only theoretical. For instance, a product-form parameterisation of the model could be applied to validate the simulation results, or in case the rate conditions are not satisfied, an approximated analysis could be carried out.

INAP numerically solve the model in isolation and then checks Condition 1) for each ergodic joint state computed by SIMTHESys. The time complexity for the model solution in isolation is proportional to the cube of each queue capacity, therefore if $B_1 = B_2 = B_3 = B$ we have $\mathcal{O}(kB^3)$, where k is the number of iterations required to converge withing a certain precision. In our experiments, we observed $k \leq 9$ for all the parameterisation we tried. Due to the sparsity of the transitions in the joint state, Condition 1 can be checked in constant time for each joint-state, therefore leading to a total time complexity of $\mathcal{O}(B^3)$. We can conclude that in this case the product-form analysis reduces the time complexity of the solution from $\mathcal{O}(B^9)$ to $\mathcal{O}(B^3)$.

4 Related works

4.1 Multiformalism modeling

Different approaches to multiformalism modeling can be found in literature: a first, loose classification suggests either extensible or non extensible frameworks. In the first category, SHARPE [24], SMART [9] and DEDS [7] are the key references, while AToM^3 [11], Möbius [23], OsMoSys [25,13] and SIMTHESys [20,19] to different extents fall in the second category.

SHARPE is a modelling framework capable of studying Markov models, queueing networks expressed in product form and Generalized Stochastic Petri Nets. SMART is a software package for designing complex discrete-state systems; it provides both numerical solution algorithms and discrete-event simulation techniques. DEDS is able to integrate models defined according to different formalism by creating a translation to a common abstract notation while AToM^3 exploits metamodeling to implement model transformations, used to solve models by its solver. Möbius supports Stochastic Activity Networks (SANs), Petri nets, Markov chains and Performance Evaluation Process Algebra (PEPA), and offers a very articulated complex model composition technique, that allows the generation of optimized solutions. OsMoSys can create multi-formalism models and uses workflow management to achieve multi-solutions, relying on meta-modeling and object-orientation in models and formalisms. SIMTHESys is a multiformalism framework for the definition of new formalisms and the generation of related solvers, based on the description of elements behavior and behavioral interfaces to integrate elementary solvers.

In the majority of these approaches, modularity or compositionality are supported, and in some cases exploited to enhance the solution process. In SHARPE modularity is managed at model level by its source code; in Möbius complex model composition policies allow optimized solution; in OsMoSys composed models are solved by the orchestration of different solvers for different submodels in a workflow [14]. In other cases, multiformalism and/or modularity are not explicitly used by the modeler but used for optimized analysis (e.g. [22]): in general, modularity suggests the possibility of a solution in parts, that is possible if certain hypotheses are verified.

The solution process can be based on the translation to a single solution formalism, as in AToM^3 and, to some extent, Möbius, or by using native solvers and a composition formalism, as for OsMoSys, or by a combination of the two techniques, as in SIMTHESys. Besides these approaches, the case is also noted where the same model can be solved with different solvers in the same tool, known as multisolution paradigm. Note that solution by reduction to another solvable formalism is widely used also in single formalism modeling techniques. This is a widespread approach providing modelers with very abstract specification mechanisms without the need for manually generate unmanageable chains. This logic is very relevant for the scope of this paper, as product-form solution applicability will be enabled by implementing in the SIMTHESys framework a true high level formalisms interaction mechanism, that can be mapped onto (product-form) Markov chains.

4.2 Product-forms

The definition of stochastic models by means of compositions of interacting sub-models plays an important role in performance engineering field. Indeed, this allows one to tackle the complexity of modern systems' hardware and/or software architectures. Nevertheless, in general, the steady-state analysis of such models do not exploit their modular definition and relies on the exact or approximate solution of the system of global balance equation of the underlying Markov chain. Product-form theory has been introduced in the field of queueing networks and, under a set of conditions, allows to perform the stationary analysis of complex models by isolating its components. Once each component is parameterized taking into account the interactions with the remaining parts of the model and is solved, the steady-state distribution of the joint model is derived as normalized product of the distributions of the isolated components. Although introduced for queueing networks, product-forms have been identified for various formalisms, such as stochastic Petri nets, stochastic automata networks and Markovian process algebra.

In this paper we have used the Multi-agent Reversed Compound Agent

Theorem (MARCAT) [18] to check the conditions and to derive the product-form steady-state distribution of multi-formalism stochastic models. MARCAT is a very general result capable of studying uniformly various classes of product-form models including Boucherie's product-forms and the product-form results derived for stochastic Petri nets [3] and queueing networks with finite capacity and Repetitive service blocking. With respect to other results such as the quasi-reversibility and the Reversed Compound Agent Theorem (RCAT) [17], MARCAT is more general but it requires an analysis of the ergodic part of the model's joint state space.

The generality of the theorems for the product-form arises problems that had not been addressed in the seminal works on product-form queueing networks. In particular, the computation of the component's parameterisation may be time expensive, requiring the solution of system of non-linear traffic equations (see e.g. [15]). This problem, associated with the generality of RCAT, has been addressed in [21] where an iterative algorithm (INAP) is given to efficiently compute the correct components' parameterisation starting from their definition in terms of stochastic automata.

In this paper we have extended INAP implementation in order to encompass MARCAT product-forms, hence checking the equations on the joint state space and develop an interface to allow SIMTHESys multiformalism tool to apply INAP for efficiently deciding is the model admits an MARCAT product-form solution and in case of positing answer computing it.

For what concerns Stochastic Automata Networks and product forms, this topic has been addressed in several works, such as for example [10,12]. The topic of our work is however quite different: instead of exploiting relations among tensor algebra and product forms, using SANs as an tool to express the dynamic of the cooperating models, we are interested in exploring the relations among product forms and the evolution of a submodel as determined by an high level tool such as SIMTHESys. In our approach, the SAN-like structure has been mainly used as an interchange format to interface the model analysis tool with the underlaying solving engine.

5 Conclusions and future work

In this paper we have presented a technique to identify product form solutions in a multiformalism framework. The check for the existence and the computation of the solution is performed during the generation of the state space of the submodels that composes the main model under study. State space is generated using behaviors that are associated with formalism specification. This provides a greater flexibility and allows the application of product form solutions to new formalism without the development of new tools.

Future work will include the study of more complex synchronization mechanisms, and the use of the product forms solutions to approximate real solution when the necessary conditions are not respected.

References

[1] Balsamo, S., G. Dei Rossi and A. Marin, *A tool for the numerical solution of cooperating Markov chains in product-form*, in: *Proc. Of Int. Conf. HET-NETs*, Zakopane, PL, 2010, pp. 311–324.

[2] Balsamo, S., P. G. Harrison and A. Marin, *A unifying approach to product-forms in networks with finite capacity constraints*, in: *Proc. of the 2010 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, New York, NY, USA, 2010, pp. 25–36.

[3] Balsamo, S., P. G. Harrison and A. Marin, *Methodological Construction of Product-form Stochastic Petri-Nets for Performance Evaluation*, J. of System and Software **85** (2012), pp. 1520–1539.

[4] Barbierato, E., M. Gribaudo and M. Iacono, *Defining formalisms for performance evaluation with SIMTHESys*, Electron. Notes Theor. Comput. Sci. **275** (2011), pp. 37–51.

[5] Barbierato, E., M. Gribaudo and M. Iacono, *Exploiting multiformalism models for testing and performance evaluation in SIMTHESys*, in: *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '11 (2011), pp. 121–130.

[6] Barbierato, E., M. Gribaudo, M. Iacono and S. Marrone, *Performability modeling of exceptions-aware systems in multiformalism tools*, in: *Proceedings of the 18th international conference on Analytical and stochastic modeling techniques and applications*, ASMTA'11 (2011), pp. 257–272.

[7] Bause, F., P. Buchholz and P. Kemper, *A toolbox for functional and quantitative analysis of deds*, in: *Proceedings of the 10th International Conference on Computer Performance Evaluation: Modelling Techniques and Tools*, TOOLS '98 (1998), pp. 356–359.

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

[9] Ciardo, G. and A. S. Miner, *Smart: The stochastic model checking analyzer for reliability and timing*, Quantitative Evaluation of Systems, International Conference on **0** (2004), pp. 338–339.

[10] Dao Thi, T. H. and J. M. Fourneau, *Stochastic automata networks with master/slave synchronization: Product form and tensor*, in: *Proceedings of the 16th International Conference on Analytical and Stochastic Modeling Techniques and Applications*, ASMTA '09 (2009), pp. 279–293.

[11] de Lara, J. and H. Vangheluwe, *Atom3: A tool for multi-formalism and meta-modelling.*, in: R.-D. Kutsche and H. Weber, editors, *FASE*, Lecture Notes in Computer Science (2002), pp. 174–188.

[12] Fourneau, J. M., B. Plateau and W. J. Stewart, *An algebraic condition for product form in stochastic automata networks without synchronizations*, Perform. Eval., Elsevier **65** (2008), pp. 854–868.

[13] Franceschinis, G., M. Gribaudo, M. Iacono, S. Marrone, N. Mazzocca and V. Vittorini, *Compositional modeling of complex systems: Contact center scenarios in osmosys*, in: *ICATPN'04*, 2004, pp. 177–196.

[14] Franceschinis, G., M. Gribaudo, M. Iacono, S. Marrone, F. Moscato and V. Vittorini, *Interfaces and binding in component based development of formal models*, in: *Proc. of the Fourth Int. ICST Conf. on Performance Evaluation Methodologies and Tools*, VALUETOOLS '09 (2009), pp. 44:1–44:10.

[15] Gelenbe, E., *Product form networks with negative and positive customers*, J. of Appl. Prob. **28** (1991), pp. 656–663.

[16] Gelenbe, E., *G-networks with triggered customer movement*, J. of Appl. Prob. **30** (1993), pp. 742–748.

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

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

[19] Iacono, M., E. Barbierato and M. Gribaudo, *The SIMTHESys multiformalism modeling framework*, Computers and Mathematics with Applications (2012).

[20] Iacono, M. and M. Gribaudo, *Element based semantics in multi formalism performance models*, in: *MASCOTS* (2010), pp. 413–416.

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

[22] Raiteri, D. C., M. Iacono, G. Franceschinis and V. Vittorini, *Repairable fault tree for the automatic evaluation of repair policies*, in: *DSN*, 2004, pp. 659–668.

[23] Sanders, W., T. Courtney, D. Deavours, D. Daly, S. Derisavi and V. Lam, *Multi-formalism and multi-solution-method modeling frameworks: The Möbius approach*, in: *Proc. of Symp. on Performance Evaluation–Stories and Perspectives*, 2003, pp. 241–256.

[24] Trivedi, K. S., *Sharpe 2002: Symbolic hierarchical automated reliability and performance evaluator*, in: *DSN '02: Proceedings of the 2002 International Conference on Dependable Systems and Networks* (2002), p. 544.

[25] Vittorini, V., M. Iacono, N. Mazzocca and G. Franceschinis, *The osmosys approach to multi-formalism modeling of systems*, Software and System Modeling **3** (2004), pp. 68–81.