Model transformations in structured stochastic Markovian formalisms

Preprint · May 2019

DOI: 10.13140/RG.2.2.16865.56162

2 authors:

Ricardo M. Czekster
Newcastle University
63 PUBLICATIONS **148** CITATIONS
SEE PROFILE

Thais Webber
University of St Andrews
73 PUBLICATIONS **281** CITATIONS
SEE PROFILE

Model transformations in structured stochastic Markovian formalisms

Ricardo M. Czekster[1], Thais Webber

Performanceware Technologies, Rua Dr Flores 262/55, Porto Alegre Rio Grande do Sul (RS), Brazil

{rczekster,webber.thais}@gmail.com

Abstract

The present paper addresses the transformation of modeling primitives in structured Markovian based formalisms such as Queueing Networks, Stochastic Petri Nets, Performance Evaluation Process Algebra, and Stochastic Automata Networks. Since all of those formalisms share the same underneath Markov Chain, there is a formal correspondence that yields the same results for a class of models described here using examples from the Queueing Networks formalism. Our idea is to provide insight for future research on this subject as we discuss how to shift the modeling considerations toward the model itself, instead of worrying about specific modeling primitives or other formalism intricacies when solving complex problems. We also discuss other approaches and translations available in the literature and discuss some key considerations when addressing model transformations among stochastic structured formalisms with Markovian assumptions.

Keywords: *stochastic modeling; structured models; Markov Chains; model translations.*

1. Introduction

Since the inception of *Markov Chains* (MC) as a formal approach suitable for system evaluation there is a growing interest by many practitioners and researchers to apply its simple to use primitives when decomposing realities into models. Pure MCs, however, were extremely costly in computational aspects, limiting a broader adoption. The main reason was related to the *state explosion problem*, where huge storage capacities were required for even small problems. That is the main reason why structured approaches have emerged as a valid research topic in recent years, which aims to represent systems more abstractly (on top of MCs) and allows more powerful analysis mechanisms.

All over the years many works have discussed how to represent system behaviors using many modeling primitives such as states, transitions, tokens, functions, guards, blocks, hierarchies, and many others. We observe, however, that little attention has been devoted towards models equivalence, i.e., models that yield the same MC underneath, causing solution to be numerically equivalent. The aim of the present paper is two-fold: firstly, it discusses the evolution of formalisms throughout the years; secondly, it analyzes a set of formalisms, particularly *Queueing Networks* (QN), *Performance Evaluation Process Algebra* (PEPA), *Stochastic Petri Nets* (SPN), and *Stochastic Automata Networks* (SAN), demonstrating model and solution equivalence, since the structured formalisms generate the same underlying MC. As a byproduct of our approach, we also address model decomposition, showing how to break down a system into smaller and thus more manageable parts that represents dynamic systems.

The work is presented as follows: Section 2 discusses modeling processes and lists several approaches to handle different realities. In Section 3 we show some formalism transformations with discussions about strengths and weaknesses, ending our work with Section 4 where we point out future works and our final considerations.

[1] Corresponding author.

2. Modeling systems using formal approaches

The decision when modeling a reality must consider the use of a *formalism* that maps states and connects entities altogether. A *formalism* defines a set of formal rules that maps properties and characteristics of systems into models. It should be able to describe behaviors unambiguously, as well as entities relations and interactions. When modeling, it is important to observe how entities behave and changes state, and the conditions that must be inspected when traversing states.

These observations are crucial to map entities to states and assign rates to transitions (connectors among states). The objective is to compute performance indices that serve as indicators of resource usage, throughput, utilization, and other metrics. Those values, when interpreted, are used to make decisions as to capacity and resource management (e.g. reallocations, or resource additions), underperformances, and unusual behaviors as new clients/transactions arrive to the system. It is possible, for instance, to study bottlenecks and general contention, i.e., discovering the limiting station or server where queueing is preventing the system to behave more efficiently.

Formal methods for computational performance assessment provide reasoning about operations that highlights bottlenecks, errors, and problems prior to physical implementations. These practices are considered cheap, consuming small amounts of resources. The idea is to apply formal concepts to represent realities with powerful modeling primitives. The models are subjected to solution allowing the investigation of stationarity properties, failure, equilibrium conditions, and state probabilities. It is important to perform those tasks in a timely fashion and waste the least amount of computational resources [42],[7][55].

The model could, for instance, inform whether the reality would start to degrade or even how it would begin to operate unsatisfactorily. The results computed by the analytical model, when interpreted, could indicate the need to upgrade the system (e.g. buying more computers) or hire more personnel *before* the expenditure of monetary sums. Because of this, analytical modeling is *de facto* a crucial technique to inspect systems.

The steps to inspect systems involve their fundamental operations, a mapping that consists in discovering the tasks to be represented as well as system evolution, i.e., how the system changes from state to state in its operation. When properly enumerated, it increases the chances to validate models and produce actionable efforts. Before embarking on a modeling effort, however, it is important to capture the systems' main operation into a simple model. The next step is to revisit this basic model and remove unnecessary states or transitions, invalid states, and impossible connections. It is worth to mention that this process is dependent on the analysts' efforts when modeling. Also, they may suggest possible optimizations and new configurations that may enhance system operation, or reduce overall resource consumption.

After the objective is set, one must select the formalism that offers the best primitives when describing the reality under study, capturing its behavior and how the entities or components interact among each other. Is it also important to consider the best tradeoffs in terms of offered tools for solution and verification. A formalism may possess several strengths when defining a system, and very poor toolset and software suites for seamless solution and analysis. The next procedure aims to investigate equilibrium, trying to verify whether the system converges to valid results when confronted to the physical operation.

When solving Markovian models, a probability array is usually mapped to the model states. The idea is to perform a numerical solution mechanism that exhaustively simulates the model to a point where the initial conditions do not influence the final results. If one achieves this objective, it means that the model has reached equilibrium, and the state probabilities can be analyzed. Figure 1 shows the steps when conducting a modeling study.

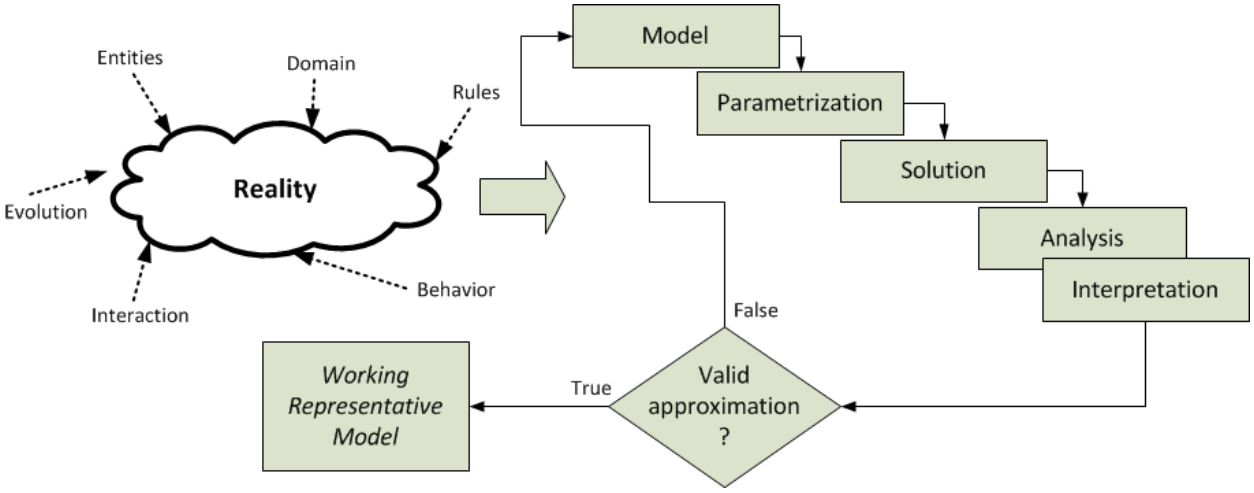


Figure 1: *Modeling process from a reality to a working representative model.*

The main system abstraction is important to derive the indices in the solution phase, because they will be inspected and interpreted in the analysis phase, followed by model modifications, change of parameters that reflect better systems operations, and feedback, in an iterative process that complements each phase. Analysts are able to validate the model using simulation [58], or monitoring [44].

The mapping of distinct realities to formalisms has been an active area of research as observed by several formal definitions created throughout the years. Table 1 shows a listing of several formalisms, and the year of its first mentions. The list is not comprehensive, i.e., it is showing a subset of available formalisms since it would be very hard to describe all specifications in their totality. The list only mentions important formalisms that are the subject of the current translation mechanism described in this paper.

Table 1: *Formalism definition, first mentions, and related research papers.*

Formalism Description	***Year***	***References***
Markov Chains (MC)	1906	[7],[48],[13],[41][56]
Queueing Networks (QN)	1909	[23][43][29][7][56]
Petri Nets (PN)	1962	[49][9]
→ *Colored Petri Nets* (CPN)	1981	[34][35][36]
→ → *Hierarchical Colored Petri Nets* (HCPN)		
→ *Superposed Stochastic Automata (SSA)*	1991	[20][21]
→ → *Superposed Generalized Stochastic Petri Nets (SGSPN)*	1994	[22]
Stochastic Automata Networks (SAN)	1985	[50][51]
Stochastic Activity Networks (SAN)	1985	[53]
Well Formed Networks (WFN)	1990	[10][11][12]
Process Algebras (PA)	1970	[30][5][2]
→ *Performance Evaluation Process Algebra* (PEPA)	1994	[37][38]
→ → *PEPA nets*	2001	[26][27][28]

Several formalisms discussed above share the same origin, which is *Markov Chains* (MC). It is based solely on *states*, *rates,* and *transitions* [41]. The strength when considering Markovian approaches rely on the fact that many systems behave and operate having the Markov property (memorylessness). There are several tools to solve MCs, we highlight MARCA for historical purposes [54], *The Octave Queueing Package* [45] for GNU/Octave has also some MC based functions, and even spreadsheets like MS-Excel package could be used to compute probability vectors, since they operate with matrix products functions.

2.1 Related works

Several structured formalisms aim to reduce the storage requirements of the full Markov Chains by adopting Kronecker (e.g. tensor) operations, the approach of the SAN formalism. There are many works inspecting conversions from PEPA to a Kronecker format [39][40]. Several researches are directed towards multiformalism research [31][32], i.e., a modeling technique that aims to direct efforts to modeling instead of choosing a formalism. That is the approach of the *Möbius Modeling Framework* [19], accepting the description of systems written in several formalisms such as PN or PEPA, to name a few. Other frameworks are available for multiformalism modeling such as *SIMTHESys* [33] or *OsMoSys* [57]. Several authors direct attention to the conversion of *Unified Modeling Language* (UML) and *Business Process Modeling and Notation* (BPMN) to a stochastic counterpart. It is out of the scope of this paper to discuss these researches, however, we highlight a translation from UML to PN available in [52], and a translation from BPMN to SAN [8].

3. Transformations of structured formalisms

Despite its broad use, MC possesses some problems and disadvantages, where the foremost issue is the *state-space explosion*. The problem is well known and affects even very simple problems, as the state space enlarges to a point where it is impossible to solve the underlying algebraic system in a timely fashion. Other solution methods such as iterative procedures such as the *Power Method* are also affected. Since the inception of Markovian modeling this problem is present, e.g., a system with three queues and twenty available positions for clients to wait processing would have 3^{20} states, nearly 3.4 billion states, rendering it impractical for numerical solution, since it would be necessary to store the transition matrix as well as the probability array (usually they are of *double* size data type where each structured have 8 bytes).

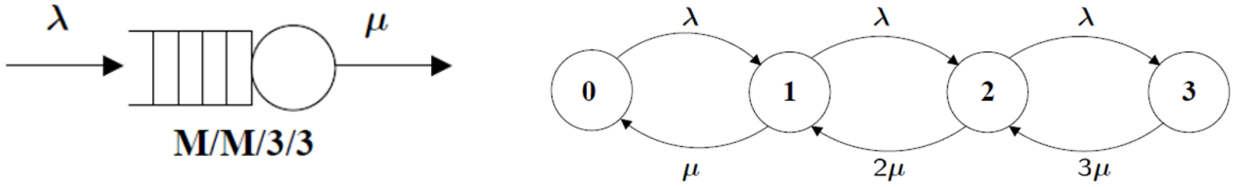
Several researchers tackled the reduction of the problematic effects of state-space explosion where they tried to devise alternatives to represent stochastic systems on top of a MC, enlarging model representativeness and at the same time, still working with state-transition systems. These alternatives are called *structured formalisms* and they use high level descriptions to abstract the implicit MC. The problem of working with structured formalisms shifted the research as to how to create interesting modeling primitives and access the underlying MC. Those structured alternatives presented with different abstractions to modelers, such as queues to *Queueing Networks* (QN), automata to *Stochastic Automata Networks* (SAN), processes in *Performance Evaluation Process Algebras* (PEPA), or tokens, in *Stochastic Petri Nets* (SPN).

It is worth mentioning that the structured formalisms significantly reduces the state-space problems, however, it does not eliminate it completely. It is still possible to device models where the state space is so large that it makes impossible for solution methods to find stationary measures efficiently. Another problem is directed towards unattainable states, i.e., from a starting point, following the transition rules for state changes, there is a chance that a set of states in the underneath MC will never be reached. This problem only exists in structured representations, since they are simply removed in MC modeling. The *Product State Space* (PSS) comprehends all states that were formed from the abstractions in structured formalisms (e.g. the *Cartesian product* of all entities' local state spaces) whereas the *Reachable State Space* (RSS) designates only the states that are possible to be reached according to the model rules. These concerns are discussed thoroughly in many researches [13][4].

3.1 Queues and Queueing Networks

The simplest case of MC is called *Birth-Death Process* (BDP), because clients arrive with rate λ and leave the system with rate μ. According to the *Kendall Notation* [43] that follows the

nomenclature A/B/C/D/E/F, where [A,B] designates the [*arrival*,*departure*] rate distribution, C marks the number of servers, D represents the queue capacity, E (not used here) maps the queue policy (e.g. FIFO, LIFO, priority, *etc*.), and F (not used) indicates the system population. For Figure 2, the BDP has arrival and departure rates drawn from the Exponential (memoryless) distribution (i.e., A,B='M'), with three *servers* (C=3), and *capacity* (D=3) equal to three (assigning the number of available positions allocated for clients to wait).

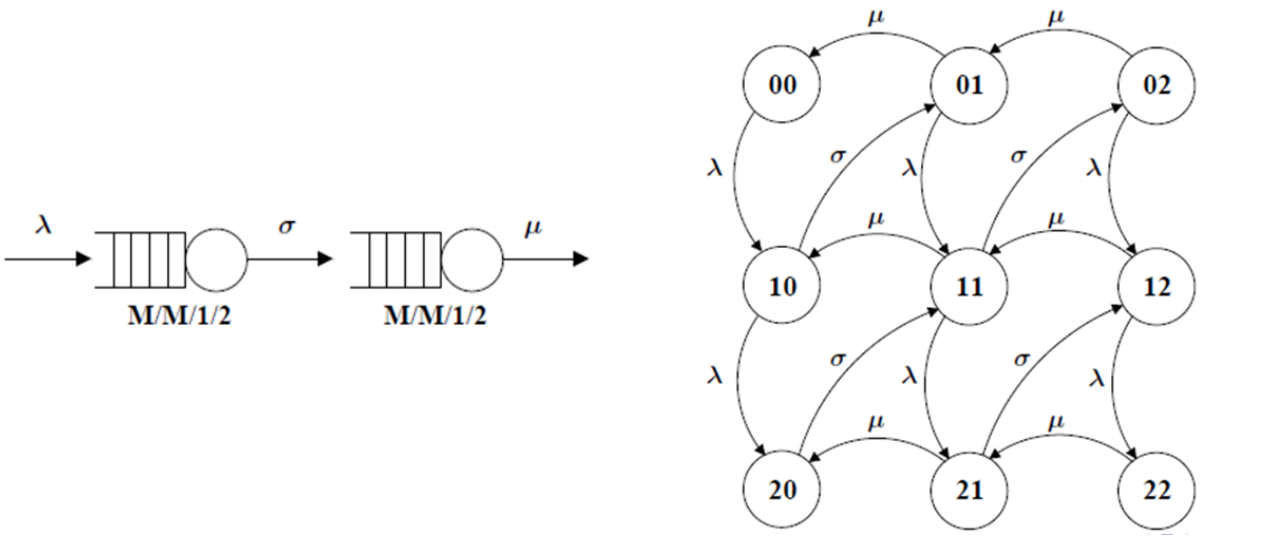


(a) (b)

Figure 2: *A queue with three servers and capacity equal to three. In (a), the QN notation for representing queues is used whereas in (b) it shows the MC representation for the same model.*

There are closed equations (i.e. *Product Form*) to numerically solve this model based on the rates, however, we will convert the model to a matrix and then apply a solution mechanism to discover the permanence probabilities for each state (for the case of the example, 0, 1, 2, and 3). With these results, analysts could reason about system operation and vary parameters to discover better configurations, e.g., those that would yield the minimum associated costs.

BDP are special cases of *Queueing Networks* (QN), because they work with several queues connected according to the problem. One must connect each queue to a set of other queues, and generate the PSS from the combinations of local states that the system assume, in this case, we are modeling the capacity, so the *Cartesian* product of $\{0,1,2\}$ *x* $\{0,1,2\}$, resulting in nine states. This conveys that the system could have zero, one, or at most two clients without reaching its full capacity. The equivalent MC is presented in Figure 3:



(a) (b)

Figure 3: *A Queueing Network consisting of two queues, with one server each, and two waiting places (capacity) in (a) and its MC representation in (b).*

QNs are structured representations of MCs since they have a visual way to communicate behavior and have an underlying MC. It is worth mentioning that QNs, similarly to BDPs, have *Product Form* equations, sparing algebraic solutions. There are several software tools to solve QNs, we cite *The Octave Queueing Package* [45], *Java Modeling Tools* – JMT [6], and common spreadsheets (e.g. MS-Excel).

3.2 Petri Nets

Petri Nets – PN [49][3] are graphical model representations of systems using basic primitives such as Places, Nodes, Transitions, and arcs connecting the places. Inside each Place, the formalism defines the instantiation of Tokens, which crudely could be associated to resources. In PN, the modeler creates the representation and assign a set of token inside each place, where the transitions allows the token movements according to a set of firing rules. The number of tokens inside each place is known as a *marking*. The objective is to determine the full set of markings (among other studies) for any PN model that indicates the reachable states that the system assumes. It is not without cause that these procedures are known as the "*Token Game*". With PN, modelers easily understand cause-effect relations, causality and conflict, as well as parallel behavior. The formalism has presented a set of extensions since its inception, for instance, [24] has discussed transitions governed by distributions, naming the technique as *Stochastic Petri Nets* (SPN), allowing non-determinism while firing transitions. Figure 4 shows an example of a SPN that maps the QN of Figure 3(a).

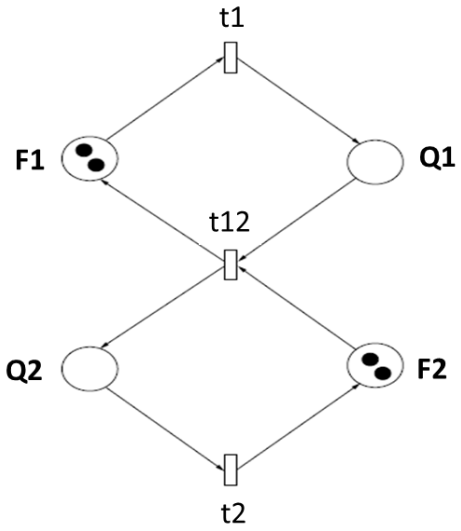


Figure 4: *A SPN of a Queueing Network, with feeding places (F1,F2), queueing places (Q1,Q2), and transitions, t1 (arrive), t12 (change from queue 1 to queue 2), and t2 (departure).*

The figure shows the necessity to define "*Feeding Places*" (in Places F1 and F2), having a number of tokens equal to each queue capacity. It is important to notice that the SPN notation is very different from the model using MCs or even QNs, requiring tokens in specific places. One could generate the same MC of Figure 3(b), counting the number of tokens in places Q1 and Q2 and assigning the transitions accordingly.

One important advantage of a SPN is the visual depiction of information flow through tokens in the network, how they interact and depend upon other tokens (e.g. resources), as well as movements restrictions along the places. SPN are solved with specialized software tools such as SMART [14]. For further clarifications and researches on the topic, refer to [46][1][47][3].

3.3 Performance Evaluation Process Algebra

A growing interest was verified throughout the years towards *Markovian Process Algebras* (MPA), where complex systems are mapped to equations that represent their behavior and operation. They are structured alternatives that work with an underlying MC, and examples are *Performance Evaluation Process Algebra* – PEPA [37], *Extended Markovian Process Algebra* – EMPA [5], among others. One special interest when defining systems with process algebras is the ability to build a compositional model using a high level abstraction. The formalism presents very simple primitives that represent system evolution and interaction with other modules (in this case, other processes).

We will direct our attention to PEPA and its internals. In this formalism, the abstraction is to allow the modeler the creation *processes* with *actions* that take a certain *duration*. So, the expression $(\alpha, r).P$ denotes a component that has performed an action α, and has evolved to a component P. Here, $\alpha \subset A$, where A is the set of actions and $P \subset C$, where C is the set of available components. PEPA is equipped with a restrict set of *combinators*, and the system descriptions are built from concurrent execution and simple interaction of sequential components. The combinators syntax is informally described as follows, however, more details can be found at [37][38].

- ***Prefix***: represents a full stop, assigning the designated action to the component. As mentioned earlier, $(\alpha, r).P$ represents that an action α with rate r is performed, and then the component starts behaving as P;
- ***Choice:*** the component $P + Q$ represents a system that is allowed to behave as P or Q, as their activities are *enabled*. The first activity to end is *distinguished* from the other, which is discarded;
- ***Constant:*** it is often convenient to associate names to behavioral component parts, and constants could be used to this end, i.e., $X \overset{\text{def}}{=} E$. The name X is in right side of the expression, signaling that, for example, $X \overset{\text{def}}{=} (\alpha, r).X$ performs α with rate r for the rest of the time;
- ***Hiding:*** this combinatory allows the abstraction of certain component behaviors, denoted by P/L. The set L identifies internal or private activities;
- ***Cooperation:*** the notation $P \underset{L}{\bowtie} Q$ is used to describe the cooperation between P and Q over L. The set L is the cooperation set and determines the activities that could be used by components to synchronize activities. The alternative $P \bowtie Q$ is used when L is empty;

The activities of the cooperation set are only performed if both activities are authorized for execution. Then, both components proceed together to complete the shared activity (a concept very similar to tokens in *Petri Nets* when firing transitions). The syntax of PEPA is formally defined by:

$$\begin{cases} S ::= (\alpha, r).S \mid S + S \mid C_s \\ P ::= P \underset{L}{\bowtie} P \mid P/L \mid C \end{cases}$$

Where S is a sequential component and P is a parallel component. C is a constant and C_S are constants for sequential components. This is a necessary condition for guaranteeing that the underlying MC will be ergodic [55]. The solution of PEPA models involves the creation of a Derivation Graph that represents the underlying MC system in a Transition Matrix, according to the rates and conditions set by the model. Figure 5 shows a QN consisting of three queues with different capacities (K) with single servers. The arrival of clients is denoted by the rate λ_1 in Q_1, where clients could be routed to Q_2 or Q_3. Clients are only routed to Q_2 if there is space ($K_2 < 3$), whereas all clients could visit Q_3. If there is no space in the queue ($K_3 \geq 2$), then this client is lost. This behavior is known as a *loss condition*.

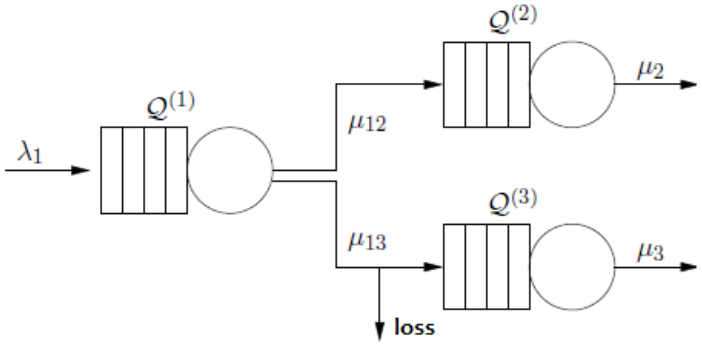


Figure 5: *A QN of three queues and different capacities.*

We have translated this QN to a PEPA model that encompasses all needed behaviors (loss, capacities, queue interactions, and clients exiting the system) in Figure 6. We chose to maintain the rates names for each case. The figure shows *three* processes ($Q_{1_n}, Q_{2_n}, Q_{3_n}$), where n represents the queue *capacity* (K_n, according to the queues definitions in Figure 5).

$$Q10 \stackrel{def}{=} (e_1, \lambda_1).Q11$$
$$Q11 \stackrel{def}{=} (e_1, \lambda_1).Q12 + (e_{13}, \mu_{13}).Q10 + (e_{12}, \mu_{12}).Q10$$
$$Q12 \stackrel{def}{=} (e_1, \lambda_1).Q13 + (e_{13}, \mu_{13}).Q11 + (e_{12}, \mu_{12}).Q11$$
$$Q13 \stackrel{def}{=} (e_{13}, \mu_{13}).Q12 + (e_{12}, \mu_{12}).Q12$$

$$Q20 \stackrel{def}{=} (e_{12}, \mu_{12}).Q21$$
$$Q21 \stackrel{def}{=} (e_{12}, \mu_{12}).Q22 + (e_2, \mu_2).Q20$$
$$Q22 \stackrel{def}{=} (e_{12}, \mu_{12}).Q23 + (e_2, \mu_2).Q21$$
$$Q23 \stackrel{def}{=} (e_2, \mu_2).Q22$$

$$Q30 \stackrel{def}{=} (e_{13}, \mu_{13}).Q31$$
$$Q31 \stackrel{def}{=} (e_{13}, \mu_{13}).Q32 + (e_3, \mu_3).Q30$$
$$Q32 \stackrel{def}{=} (e_3, \mu_3).Q31 + (e_{13}, \mu_{13}).Q32$$

$$\mathbf{S} \stackrel{def}{=} Q20 \underset{e_{12}}{\bowtie} Q10 \underset{e_{13}}{\bowtie} Q30$$

Figure 6: *Queueing Networks model of Figure 5 using PEPA's notation.*

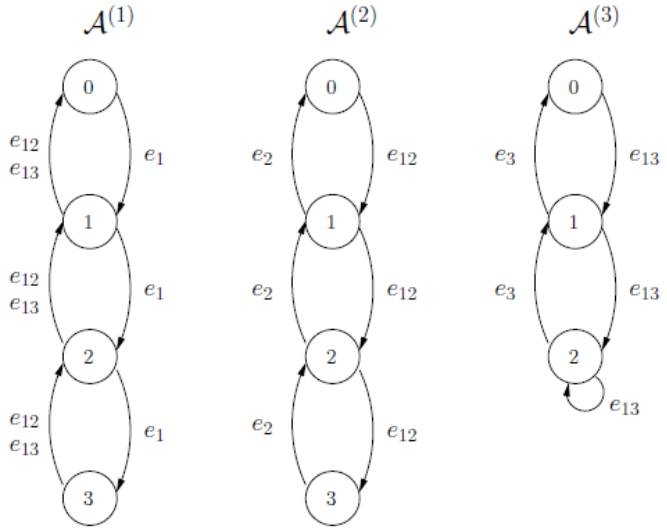
S represents the *System Equation*, i.e., the set of rules that enables the construction of the underlying MC. It associates how the clients of the first queue are routed to the other queues. Evidently, the same MC is generated for both models, yielding the same results. Process algebras are interesting system abstractions, considering processes and behaviors, however, they are purely textual (when compared to PNs, for example), a major disadvantage to users.

3.4 SAN – Stochastic Automata Networks

SAN is a structured formalism used to compute performance indices quantitatively. The formalism is based on semi-autonomous entities called automata that may interact with other automata to perform tasks, synchronize activities, or change states. The set of automata defines a network that could be used to map complex system behaviors. The formalism was originally developed to analyze parallelism [51] and synchronicity of independent entities that eventually cooperate to perform tasks. In SAN, the modeler creates the network where each automata has a set of local states and a set of transitions among states, where events are assigned. Each event has a rate of occurrence and can be one of two types: *local* or *synchronizing* events. Local events are independent from other entities, whereas synchronizing depends on the local state of other automaton to be triggered. These two types of events captures the operational semantics of several complex systems, i.e., systems that operates independently from other systems, however, sometimes, they must synchronize activities with other entities.

In SAN, the solution mechanism is based on *Tensor Algebra*, so the underlying MC is never fully generated, only accessed through tensor properties well defined by the formalism. The automata network creates a *Kronecker Descriptor*, i.e., a notation that indicates the set of local matrices that will be used to address the underlying MC in a memory efficient manner [50]. Special algorithms are devised to work with those huge sized descriptors [18].

Figure 7 shows an example of a SAN for the QN described in Figure 4. The model shows three automaton ($A^{(1,2,3)}$, one for each queue), where their local state spaces are based to their capacity (K_i). The arrival of clients are a *local* event named e_1 with rate λ_1, being independent from other automaton. For example, event e_2 is *synchronizing*, i.e., the departure of automaton $A^{(1)}$ is related to the arrival at automaton $A^{(2)}$, and so forth.



Type	Event	Rate
loc	e_1	λ_1
loc	e_2	μ_2
loc	e_3	μ_3
syn	e_{12}	μ_{12}
syn	e_{13}	μ_{13}

Figure 7: *A SAN representation of the Queueing Network described in Figure 5.*

A SAN model is easily solved by *GTA Express*, a software suite that efficiently handles *Kronecker Descriptors* and its algorithms [17]. Several techniques are used to extend SAN capabilities, directed towards *Perfect Simulation* [25]. We stress that the same results are attainable for all models since they share the same underlying MC.

3.5 Discussion

All formalisms presented here are in direct relation with Markov Chains, which has the simplest set of primitives to model complex systems, even having all state-space explosion problems associated. The key when studying and analyzing systems is the ability to create abstract versions that closely reflects the behavior into an entity called a model. These approximations are crucial to allow modelers to consider other approaches prior to physical implementation, reducing costs and other issues (e.g., personnel adjustments, expensive machinery acquisition, among others).

It is worth mentioning that all structured formalisms allows formal semantics to describe systems, some are extremely visual, whereas others are built with simplicity considerations. It is the job of the modeler to choose the best formalism to model the system under study. All structured formalisms decompose systems into entities, e.g., processes, queues, automaton, tokens, and so on, and this is crucial when addressing complex systems. The modeler should be able to break down the problem into smaller, manageable parts and then extract reasoning and consider other scenarios.

4. Final considerations and future works

The process of mapping complex realities in models depends on intuition and experience so useful models are created and studied. The present work showed valid system decompositions to structured stochastic as well as business process notations counterparts, establishing the need to divide-and-conquer problems to assess the performance of complex systems. The main idea of structured formalisms is to break down systems within many parts, having local behaviors and eventually synchronizing activities, a common behavior that encompasses many systems.

In this paper, we have addressed the fact that there are many ways to decompose systems within models, however, the objective is to identify the most important parts and understand its relations in a deeper level. Many future works are possible when addressing those concerns, for instance, there is a need to propose a valid top level conceptual framework and description language that enable modelers to consider only the problem at hand, hiding inner intricacies that sometimes slow down productivity.

We aim, in the future, to address multiformalism representations and solution mechanisms that potentially yields the same results, demonstrating the strengths for each formalism and the classes of problems that are best dealt for each.

References

[1] M. Ajmone-Marsan, G. Conte, G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems, ACM Transactions on Computer Systems, vol. 2-2 (1984) 93-122.

[2] J. C. M. Baeten. A brief history of process algebra, Theoretical Computer Science, vol. 335-(2-3), (2005) 131-146.

[3] G. Balbo. Introduction to Generalized Stochastic Petri Nets, Lecture Notes in Computer Science, vol. 4486 (2007) 83-131.

[4] A. Benoit, L. Brenner, P. Fernandes, B. Plateau. Journal of Linear Algebra and its Applications, vol. 386 (2004) 111-136.

[5] M. Bernardo, R. Gorrieri. A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time, Theoretical Computer Science, vol. 202-(1-2) (1998) 1-54.

[6] M. Bertoli, G. Casale, G. Serazzi. JMT: performance engineering tools for system modeling. ACM SIGMETRICS Performance Evaluation Review, ACM press, vol. 36(4) (2009) 10-15.

[7] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi. Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. John Wiley & Sons, 1998.

[8] K. R. Braghetto, J. E. Ferreira, J.-M. Vincent. Performance evaluation of business processes through a formal transformation to SAN. Computer Performance Engineering (2011) 42-56.

[9] G. Chiola, M. Ajmone-Marsan, G. Balbo, G. Conte. Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications, IEEE Transactions on Software Engineering, vol. 19-2 (1993) 89-107.

[10] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad. On well-formed coloured nets and their symbolic reachability graph. In: 11th International Conference on Application and Theory of Petri Nets (1990) 387-410.

[11] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications, IEEE Transaction on Computers, vol. 42-11 (1993) 1343-1360.

[12] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad. A symbolic reachability graph for coloured petri nets, Theoretical Computer Science, vol. 176-(1-2) (1997) 39-65.

[13] M.-Y. Chung, G. Ciardo, S. Donatelli, N. He, B. Plateau, W. J. Stewart, E. Sulaiman, J. Yu. A Comparison of Structural Formalisms for Modeling Large Markov Models. In: 18th International Parallel and Distributed Processing Symposium (2004) 8p.

[14] G. Ciardo, R. L. Jones, A. S. Miner, R. Siminiceanu. SMART: Stochastic Model Analyzer for Reliability and Timing. In: Aachen International Multiconference on Measurement, Modelling, and Evaluation of Computer-Communication Systems (2001) 29-34.

[15] G. Ciardo, A. S. Miner. Storage alternatives for large structured state spaces. In: Proceedings of the 9th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation, Springer Berlin Heidelberg, vol. LNCS 1245 (1997) 55-57.

[16] R. M. Czekster, P. Fernandes, J. M. Vincent, T. Webber. Split: a flexible and efficient algorithm to vector-descriptor product. In: 2nd International Conference on Performance Evaluation Methodologies and Tools – VALUETOOLS (2007) 8p.

[17] R. M. Czekster, P. Fernandes, T. Webber. GTAexpress: a Software Package to Handle Kronecker Descriptors. In: 6th International Conference on Quantitative Evaluation of Systems – QEST (2009) 281-282.

[18] R. M. Czekster, P. Fernandes, T. Webber. Efficient vector-descriptor product exploiting time-memory trade-offs. Performance Evaluation Review (PER), vol. 39 (2011) 2-9.

[19] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, P. G. Webster. The Möbius Framework and Its Implementation. IEEE Transactions on Software Engineering, vol. 28(10) (2002) 956-969.

[20] S. Donatelli. Superposed Stochastic Automata: a class of stochastic Petri nets amenable to parallel solution. In: Proceedings of the Fourth International Workshop on Petri Nets and Performance Models – PNPM91 (1991) 54-63.

[21] S. Donatelli. Superposed stochastic automata: a class of stochastic Petri nets with parallel solution and distributed state space. Performance Evaluation, vol. 18.1 (1993) 21-36.

[22] S. Donatelli. Superposed generalized stochastic Petri nets: definition and efficient solution. Application and Theory of Petri Nets. Springer Berlin Heidelberg (1994) 258-277.

[23] A. K. Erlang. The Theory of Probabilities and Telephone Conversations. Nyt Tidsskrift for Matematik, vol. 20-B (1909) 33-39.

[24] G. Florin, S. Natkin. In: Technique et Science Informatiques, Vol. 4, No. 1 (1985) 143-160.

[25] P. Fernandes, J.-M. Vincent, T. Webber. Perfect Simulation of Stochastic Automata Networks. In: 15th International Conference of Analytical and Stochastic Modeling Techniques and Applications – ASMTA (2008) 249-263.

[26] S. Gilmore, J. Hillston, M. Ribaudo. PEPA-coloured stochastic Petri nets. In: 17th UK Performance Engineering Workshop – UKPEW (2001) 155-166.

[27] S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo. PEPA nets: a structured performance modelling formalism, Performance Evaluation, vol. 54-2 (2003) 79-104.

[28] S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo. Software performance modelling using PEPA nets. In: 4th International Workshop on Software and Performance – WOSP (2004) 13-24.

[29] E. Gelenbe, G. Pujolle. Introduction to Queueing Networks. John Wiley & Sons, 192p, 1987.

[30] N. Götz, H. Hermanns, U. Herzog, V. Mertsiotakis, M. Rettelbach. Quantitative Methods in Parallel Systems: Stochastic process algebras. Springer, 1995.

[31] M. Gribaudo. Theory and Application of Multi-Formalism Modeling. IGI Global, 293p, 2013.

[32] M. Gribaudo, M. Iacono. An introduction to multiformalism modeling. Theory and Application of Multi-Formalism Modeling (2013) 314-329.

[33] M. Iacono, E. Barbierato, G. Marco. The SIMTHESys multiformalism modeling framework. Computers & Mathematics with Applications vol. 64(12) (2012) 3828-3839.

[34] K. Jensen. Coloured Petri Nets and the Invariant Method, Theoretical Computer Science, vol. 14-3 (1981) 317-336.

[35] K. Jensen. Coloured Petri nets: basic concepts, analysis methods and practical use. Springer, 1996.

[36] K. Jensen, L. M. Kristensen, L. Wells. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems, International Journal on Software Tools for Technology Transfer, vol. 9-(3-4) (2007) 213-254.

[37] J. Hillston. A compositional approach to performance modelling. Cambridge University Press, 1996.
[38] J. Hillston. Process algebras for quantitative analysis. In: 20th Annual IEEE Symposium on Logic in Computer Science – LICS (2005) 239-248.
[39] J. Hillston, L. Kloul. An efficient Kronecker representation for PEPA models. Process Algebra and Probabilistic Methods. Performance Modelling and Verification. Springer Berlin Heidelberg (2001) 120-135.
[40] J. Hillston, L. Kloul. Formal techniques for performance analysis: blending SAN and PEPA. Formal Aspects of Computing, vol. 19.1 (2007) 3-33.
[41] R. A. Howard. Dynamic Probabilistic Systems, Volume I: Markov Models. Dover Publ., 2007.
[42] R. Jain. The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation, and Modeling. Digital Equipment Corporation – Littleton, Massachusetts. John Wiley & Sons, Inc. 1991.
[43] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik. Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Prentice Hall, 1984.
[44] D. J. Lilja. Measuring computer performance: a practitioner's guide. Cambridge University Press, 2005.
[45] M. Marzolla. The Octave Queueing Package, In: Proceedings of the 11th International Conference on Quantitative Evaluation of SysTems (QEST 2014), September 8-10, Florence, Italy, vol. 8657 of LNCS (2014) 174-177.
[46] M. K. Molloy. Performance analysis using stochastic Petri nets, IEEE Transactions on Computers, vol. C-31-(9) (1982) 913-917.
[47] T. Murata. Petri nets: Properties, analysis and applications, Proceedings of the IEEE, vol. 77-4 (1989) 541-580.
[48] J. R. Norris. Markov Chains. Cambridge University Press, 1998.
[49] C. A. Petri. Communication with automata. DTIC Research Report AD0630125, vol. 1-1, 1966.
[50] B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. In: International Conference on Measurements and Modeling of Computer Systems (1985) 147-154.
[51] B. Plateau, K. Atif. Stochastic Automata Networks for modeling parallel systems, IEEE Transactions on Software Engineering, vol. 17-10 (1991) 1093-1108.
[52] J. Saldhana, S. M. Shatz. UML diagrams to object petri net models: An approach for modeling and analysis. International Conference on Software Engineering and Knowledge Engineering. (2000) 103-110.
[53] W. H. Sanders, J. F. Meyer. Stochastic Activity Networks: Formal Definitions and Concepts, Lecture Notes in Computer Science, vol. 2090 (2001) 315-343.
[54] W. J. Stewart. MARCA: Markov chain analyzer, a software package for Markov modeling. In: 1st International Workshop on the Numerical Solution of Markov Chains – NSMC (1991) 37-62.
[55] W. J. Stewart. Performance Modelling and Markov Chains, Lecture Notes in Computer Science, vol. 4486 (2007) 1-34.
[56] W. J. Stewart. Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University Press, 2009.
[57] V. Vittorini, M. Iacono, N. Mazzocca, G. Franceschinis. The OsMoSys approach to multi-formalism modeling of systems. Software and Systems Modeling, vol. 3(1) (2004) 68-81.
[58] G. Wainer. Discrete-event modeling and simulation: a practitioner’s approach. CRC Press, 2010.