A Methodology based on EMPA for Modeling and Simulating Concurrent Systems

Marco Bernardo

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
Phone: +39-51-354516 Fax: +39-51-354510 E-mail: bernardo@cs.unibo.it

Abstract

An integrated approach for modeling and analyzing functional and performance properties of concurrent systems has been proposed in [3]. The approach is based on the stochastic process algebra EMPA and the class of generalized stochastic Petri nets. In this paper we assess the suitability of the approach, and in particular of EMPA, for simulation purposes. The result is that EMPA turns out to be adequate both from the modeling point of view due to its compositionality and its expressive power, and from the simulative analysis point of view since its semantics has been defined in the operational style.

1 Introduction

In the theory of concurrency several techniques have been developed in order to describe and analyze concurrent systems in a formal way. Initially, these techniques were concerned only with the functional aspects of concurrent systems: performance aspects were completely neglected. Because of the importance of integrating the performance modeling and evaluation of a concurrent system into the design process of the system itself, in the last two decades many formal description techniques have been enriched in order to take into account performance aspects.

One of the most successful attempts to integrate functionality and performance has been the development of stochastic Petri nets. Given a stochastic Petri net model of a concurrent system, well known techniques can be applied for the functional analysis of the underlying classical Petri net and the performance analysis of the underlying stochastic process. Furthermore, the stochastic Petri net model is amenable to simulative analysis, and this turns out to be quite helpful whenever the underlying stochastic process cannot be solved by means of analytical or numerical techniques.

Since the various analysis techniques mentioned above are usually supported by already existing tools, the main problem a system designer has to do with is the construction of a stochastic Petri net model of the system at hand. This stems from the lack of compositionality typical of stochastic Petri nets.

To overcome the problem above we have proposed in [3] the adoption of a stochastic process algebra as a formalism for modeling systems. This is motivated by the fact that stochastic process algebras naturally supply compositionality: the system designer is provided with a small set of powerful operators (e.g. parallel composition) whereby it is possible to construct process terms from simpler ones and hide irrelevant details. In order to exploit their complementary advantages, in [3] we have also proposed an integrated methodology for modeling and analyzing concurrent systems which is based on both stochastic process algebras and stochastic Petri nets. The methodology (see Fig. 1) consists of two phases:

1. The first phase requires the system designer to specify the concurrent system as a stochastic process algebraic term. Because of compositionality, the system designer is allowed to develop the algebraic representation of the system in a modular way. From the algebraic representation, an integrated interleaving semantic model is automatically derived in the form of a state-transition graph, where each

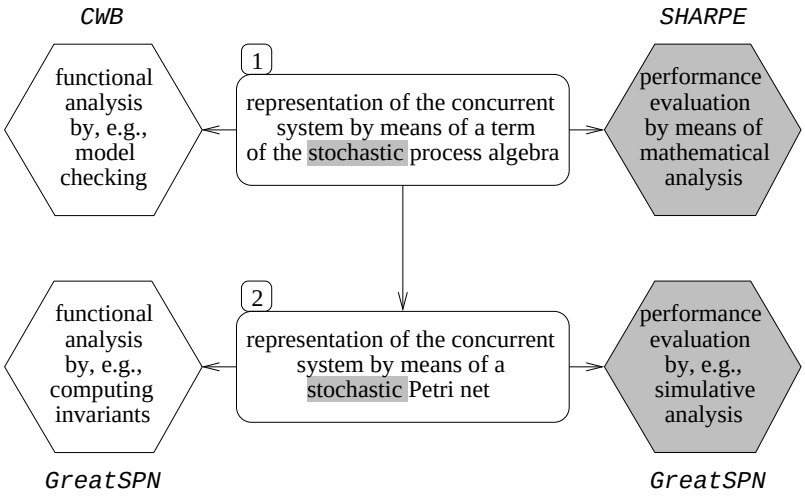

Figure 1: Integrated methodology

transition is labeled with both the type and the duration of the corresponding action. The integrated interleaving semantic model can be analyzed as a whole by a notion of integrated equivalence, otherwise it is projected on a functional semantic model and a performance semantic model that can be separately analyzed by means of existing tools such as e.g. CWB [7] and SHARPE [12].

2. The second phase starts by automatically obtaining from the algebraic representation of the system an equivalent representation in the form of a stochastic Petri net. The net representation is usually more compact than the state-transition graph, since concurrency is kept explicit instead of being simulated by alternative computations obtained by interleaving actions of concurrent components, and this is quite helpful for analysis purposes because one does not incur in the state space explosion problem typical of interleaving models. The functional and performance analysis of the stochastic Petri net, as well as its simulation, can be assisted by existing tools such as e.g. GreatSPN [6].

Since the two phases above are complementary, the choice between them is made depending on the adequacy of the related representation with respect to the analysis of the concurrent system under consideration, and the availability of the corresponding tools. In any case, the system designer is forced to start with an algebraic representation of the system in order to exploit the compositionality provided by stochastic process algebras and avoid possible graphical complexity of stochastic Petri nets.

In [3] we have implemented the methodology above in the case when action durations are exponentially distributed or zero: the stochastic process algebra is called EMPA (Extended Markovian Process Algebra) [4], the underlying stochastic processes are homogeneous continuous-time Markov chains, and the class of stochastic Petri nets we have considered is composed of generalized stochastic Petri nets [1].

The purpose of this paper is to investigate the possible use of the integrated methodology, and of EMPA in particular, from the standpoint of simulation. Concerning the integrated methodology in general, it seems to be very fruitful since it combines compositional modeling provided by stochastic process algebras with already existing techniques and tools for the simulation of stochastic Petri nets. However, it would be interesting to be able to simulate directly the stochastic process algebraic descriptions thereby avoiding the need to build the underlying stochastic Petri nets. As we shall see in the next section by means of an example, the simulative analysis of EMPA terms describing concurrent systems can be carried out because the underlying integrated interleaving semantic models are defined in the operational style.

2 Simulating a fork-join queueing system with EMPA

A fork-join queueing system [8, 2] is a queueing system composed of k independent servers with synchronized arrival streams. Customers arrive at the system in bulk of size k. Each bulk is split into the k customers it is composed of, and these customers are assigned to different servers (fork primitive). After that, each customer is served separately. Finally, when all the k customers composing the same bulk have been served, they are merged in order to obtain again the original bulk and then the bulk leaves the system (join primitive).

Fork-join queueing systems arise in many applications such as parallel processing (e.g. cobegin and coend structures in concurrent languages) and flexible manufacturing. In such cases, one of the most important performance measures of the system at hand is the mean bulk response time, i.e. the average time lapse between the fork and the join of a bulk. The purpose of this section is to show how a fork-join queueing system can be modeled with EMPA in a compositional way, and how the mean bulk response time can be assessed by means of the simulative analysis of the formal description. Concerning EMPA, we shall not give the definition of its syntax and its semantics due to lack of space: the interested reader is referred to [3, 4].

2.1 Compositional description of the system

In order to write down the EMPA specification of the fork-join queueing system ($k = 3$), we consider the entities involved in the system itself. These are the bulk arrival process, the fork primitive, the service centers, the join primitive, and the bulk departure process. Since we want to take advantage of the compositionality provided by EMPA, we model each of the five entities above by means of a different term (*Arrival*, *Fork*, *Centers*, *Join*, *Departure*) that we shall specify later, and we describe the whole system (*FJSystem*) as the parallel composition of such entities: [1]

$$FJSystem \triangleq Arrival \parallel_{\{bulk\text{-}arr\}} Fork \parallel_{\{cust\text{-}arr_i | 1 \leq i \leq 3\}} Centers \parallel_{\{cust\text{-}dep_i | 1 \leq i \leq 3\}} Join \parallel_{\{bulk\text{-}dep\}} Departure$$

where *bulk-arr* is the action type describing the arrival of a bulk, $cust\text{-}arr_i$ is the action type describing the arrival of a customer at center i, $cust\text{-}dep_i$ is the action type describing the departure of a customer served at center i, and *bulk-dep* is the action type describing the departure of a bulk.

Now we can focus our attention on the individual entities separately. Concerning the arrival of bulks, we assume it is a Poisson process with rate λ. As a consequence, the arrival of bulks can be modeled through the following term which is able to execute infinitely many actions whose type is *bulk-arr* and whose duration is exponentially distributed with rate λ: [2]

$$Arrival \triangleq <bulk\text{-}arr, \lambda>.Arrival$$

The fork primitive is modeled by means of the parallel composition of as many terms as there are centers in the system, where each term is responsible for the delivery of one of the customers composing the arrived bulk to the corresponding center. The fork primitive can be modeled as follows:

$$Fork \triangleq F_1 \parallel_{\{bulk\text{-}arr\}} F_2 \parallel_{\{bulk\text{-}arr\}} F_3$$

where

$$F_i \triangleq <bulk\text{-}arr, *>.<cust\text{-}arr_i, \infty_{4,1}>.F_i$$

Note that action *bulk-arr* is passive (rate "$*$") from the point of view of *Fork*, i.e. its duration is undefined: the duration will get a value upon synchronization with *Arrival*. Furthermore, note that action $cust\text{-}arr_i$ is immediate (rate $\infty_{4,1}$ where 4 is the priority level and 1 is the weight), i.e. its duration is zero. The capability of expressing immediate actions in EMPA provides a flexible way to describe logical events (such as the splitting of a bulk of customers) which are irrelevant from the performance evaluation point of view.

In the system there are three independent service centers:

$$Centers \triangleq C_1 \parallel_\emptyset C_2 \parallel_\emptyset C_3$$

Service center i can be modeled as follows:

[1] The parallel composition operator of EMPA has the form "$\_ \parallel_S \_$" where S is the set of action types on which the synchronization between the two involved terms is forced.

[2] The prefix operator of EMPA "$<a, \tilde{\lambda}>.\_$" expresses the sequential composition of an action and a term.

$$C_i \triangleq Queue_{i,0} \parallel_{\{cust\text{-}del_i\}} Server_i$$

where $cust\text{-}del_i$ is the action type describing the delivery of a customer from the queue to the server. Since we assume that each queue is an infinite FIFO queue, it can be modeled as follows: [3]

$$\begin{aligned} Queue_{i,0} &\triangleq <cust\text{-}arr_i, *>.Queue_{i,1}, \\ Queue_{i,h} &\triangleq <cust\text{-}arr_i, *>.Queue_{i,h+1} + <cust\text{-}del_i, *>.Queue_{i,h-1}, \quad h > 0 \end{aligned}$$

Concerning the servers, we assume that: the service time of the first server is exponentially distributed with rate μ, the service time of the second server has a two stage Erlang distribution with rate γ, and the service time of the third server has a two stage hyperexponential distribution with branching probabilities p_1 and p_2 and rates δ_1 and δ_2 respectively. The first server can be modeled as follows:

$$Server_1 \triangleq <cust\text{-}del_1, \infty_{1,1}>.<cust\text{-}serv_1, \mu>.<cust\text{-}dep_1, \infty_{1,1}>.Server_1$$

where $cust\text{-}serv_i$ is the action type describing the fact that a customer is being served. Note that actions $cust\text{-}del_i$ and $cust\text{-}dep_i$ have been modeled as immediate because they are irrelevant from the performance evaluation point of view. The second server can be modeled as follows:

$$Server_2 \triangleq <cust\text{-}del_2, \infty_{2,1}>.<cust\text{-}serv_{2,1}, \gamma>.<cust\text{-}serv_{2,2}, \gamma>.<cust\text{-}dep_2, \infty_{2,1}>.Server_2$$

where the Erlang distributed service time has been simply modeled by means of a sequence of two exponentially timed actions. Finally, the third server can be modeled as follows:

$$Server_3 \triangleq <cust\text{-}del_3, \infty_{3,p_1}>.Server_{3,1} + <cust\text{-}del_3, \infty_{3,p_2}>.Server_{3,2}$$

where

$$Server_{3,j} \triangleq <cust\text{-}serv_3, \delta_j>.<cust\text{-}dep_3, \infty_{3,1}>.Server_3$$

and the hyperexponentially distributed service time has been modeled by means of the interplay of exponentially timed actions and the weights of immediate actions. As a matter of fact, the capability of associating weights with immediate actions enables the designer to express probabilistic choices.

The join primitive is modeled by means of the parallel composition of as many terms as there are centers in the system, where each term is responsible for the acceptance of one of the customers composing the bulk. The join primitive can be modeled as follows:

$$Join \triangleq J_{1,0} \parallel_{\{bulk\text{-}dep\}} J_{2,0} \parallel_{\{bulk\text{-}dep\}} J_{3,0}$$

where:

$$\begin{aligned} J_{i,0} &\triangleq <cust\text{-}dep_i, *>.J_{i,1}, \\ J_{i,h} &\triangleq <cust\text{-}dep_i, *>.J_{i,h+1} + <bulk\text{-}dep, *>.J_{i,h-1}, \quad h > 0 \end{aligned}$$

Finally, the departure of bulks can be modeled as follows:

$$Departure \triangleq <bulk\text{-}dep, \infty_{5,1}>.Departure$$

where action $bulk\text{-}dep$ has been modeled as immediate in that irrelevant from the standpoint of performance evaluation.

In conclusion, we have obtained a formal description of the fork-join queueing system. Such a formal description has been developed in a modular way thanks to the compositionality of EMPA. Moreover, this description is much more concise than descriptions given by means of standard programming languages, as well as more readable than descriptions given by means of graphical formalism such as Petri nets.

[3] The alternative composition operator of EMPA "$\_ + \_$" expresses a choice between two terms: the term executing the action having the least duration is selected, while the other is discarded (race policy).

2.2 Discrete-event simulation of the formal description

Suppose now we are interested in computing the mean bulk response time of the fork-join queueing system described in the previous section. As we can see from the EMPA specification, the underlying integrated interleaving semantic model and Markov chain have infinitely many states because we have considered infinite FIFO queues. Even if we consider queues whose capacity is small, we would obtain an integrated interleaving semantic model and a Markov chain with hundreds of states, despite of the careful choice of priority levels of immediate actions which changes unnecessary interleavings of immediate actions into a single sequence.

As a consequence, in order to determine the mean bulk response time we must resort to a simulative analysis. So now the question arises as to whether it is possible to simulate the fork-join queueing system specification given by means of EMPA. Fortunately, this is possible because the integrated interleaving semantics for EMPA has been defined in the operational style [11] instead of the denotational style [13]: given an EMPA term, the actions it can execute are inductively computed starting from the actions executable by its subterms, while in the denotational style the whole semantic model (which may be huge especially in the interleaving setting, or even infinite) of the term should be inductively computed starting from the whole semantic models of its subterms. This means that the state-transition graph underlying an EMPA term must not be completely built since it can be computed on a "by need" basis, thus only the visited states and their outgoing transitions are actually generated as the simulation run proceeds.

In order to carry out the discrete-event simulation of *FJSystem*, we have to introduce a clock variable which accounts for the time elapsed since the beginning of the run. Furthermore, we need a variable where the number of arrived bulks is stored, and a variable where the sum of the response times of the arrived bulks is stored. The former variable must be updated whenever an action having type *bulk-dep* is executed, while the latter variable must be updated whenever an action having type *bulk-arr* or *bulk-dep* is executed. Finally, we also need pseudo-random number generators for exponentially distributed durations and for probabilistic choices. After having all these items available, the discrete-event simulation of *FJSystem* can take place by means of standard techniques such as the method of independent replications [14], and an estimate of the mean bulk response time can be derived together with the corresponding confidence interval.

3 Conclusion

In this paper we have assessed the adequacy of EMPA for simulation purposes. The result is that EMPA turns out to be helpful for modeling concurrent systems thanks to its compositionality, which should have been emphasized by the example considered in the previous section, and its expressive power, which is the sum of the expressive power of a classical, a prioritized, a probabilistic and an exponentially timed process algebra as recognized in [5]. Furthermore, the discrete-event simulation of EMPA terms is made possible by the fact that their underlying integrated interleaving semantic models are defined in the operational style, hence they can be built on a by need basis.

To the best of our knowledge, other two proposals have appeared in the literature which are related to the use of stochastic process algebras for discrete-event simulation.

In [9] a stochastic process algebra is defined where actions can be executed after a given delay whose duration is explicitly fixed or sampled from a general probability distribution. As a consequence, this stochastic process algebra provides the designer with the possibility of expressing directly any continuous probability distribution. However, unlike EMPA the possibility of describing priorities is missing. Also, it is worth pointing out the lack of an integrated notion of action, viewed as being made out of a type and a duration, which could be more appealing from the modeling point of view.

In [10] a stochastic process algebra is defined where each action is composed of a type and an arbitrarily distributed delay, and the semantics is defined in terms of stochastic bundle event structures instead of interleaving state-transition graphs, thus resulting in a more elegant treatment of general distributions with respect to [9] and more compact semantic models with respect to both EMPA and [9]. The main difference with respect to EMPA is again the fact that delays can follow arbitrary continuous distribution functions closed under product. However, unlike EMPA it is possible to express neither priorities nor probabilities.

Moreover, the event structure based semantics is defined in the denotational style instead of the operational style, so finite representation problems may arise when dealing with recursive terms.

Concerning future work, we plan to add discrete-event simulation utilities to the tool under construction which will implement the integrated methodology in Fig. 1. We are also currently investigating how to extend the expressiveness of EMPA in order to cope with general distributions.

Acknowledgements

We would like to thank Lorenzo Donatiello for the valuable discussions. This research has been partially supported by MURST and CNR.

References

[1] M. Ajmone Marsan, G. Balbo, G. Conte, *"A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems"*, in ACM Trans. on Computer Systems 2:143-172, 1984

[2] F. Baccelli, A.M.Makowski, A. Shwartz, *"Simple Computable Bounds and Approximations for the Fork-Join Queue"*, in Proc. of the *Int. Workshop on Computer Performance Evaluation*, pp. 437-450, 1985

[3] M. Bernardo, L. Donatiello, R. Gorrieri, *"Integrating Performance and Functional Analysis of Concurrent Systems with EMPA"*, Technical Report UBLCS-95-14, University of Bologna (Italy), 1995

[4] M. Bernardo, R. Gorrieri, *"Extended Markovian Process Algebra"*, in Proc. of the *7th Int. Conf. on Concurrency Theory (CONCUR '96)*, LNCS 1119:315-330, 1996

[5] M. Bernardo, *"On the Coexistence of Exponential, Immediate and Passive Actions in EMPA"*, to appear in Proc. of the *4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, 1996

[6] G. Chiola, *"GreatSPN 1.5 Software Architecture"*, in Proc. of the *5th Int. Conf. on Modeling Techniques and Tools for Computer Performance Evaluation*, Elsevier, pp. 121-136, 1991

[7] R. Cleaveland, J. Parrow, B. Steffen, *"The Concurrency Workbench: a Semantics-Based Tool for the Verification of Concurrent Systems"*, in ACM Trans. on Programming Languages and Systems 15(1):36-72, 1993

[8] L. Flatto, S. Hahn, *"Two Parallel Queues Created by Arrivals with Two Demands"*, in SIAM Journal on Applied Mathematics 44:1041-1053, 1984

[9] P.G. Harrison, B. Strulo, *"Stochastic Process Algebra for Discrete Event Simulation"*, in *"Quantitative Methods in Parallel Systems"*, F. Baccelli, A. Jean-Marie and I. Mitrani editors, Springer, pp.18-37, 1995

[10] J.-P. Katoen, E. Brinksma, D. Latella, R. Langerak, *"Stochastic Simulation of Event Structures"*, to appear in Proc. of the *4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, 1996

[11] G.D. Plotkin, *"A Structural Approach to Operational Semantics"*, Technical Report DAIMI-FN-19, University of Aarhus (Denmark), 1981

[12] R.A. Sahner, K.S. Trivedi, *"Reliability Modelling Using SHARPE"*, in IEEE Trans. on Reliability 13:186-193, 1987

[13] J.E. Stoy, *"Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory"*, MIT Press, 1977

[14] P.D. Welch, *"The Statistical Analysis of Simulation Results"*, in *"Computer Performance Modeling Handbook"*, S.S. Lavenberg editor, Academic Press, pp.267-329, 1983