Towards an Integrated Specification and Analysis of Functional and Temporal Properties:
Part I: Functional Aspect Verification

Mokdad Arous and Djamel-Eddine Saïdouni

MISC Laboratory, Mentouri University, Constantine, 25000, Algeria

Abstract. *Maximality-based Labeled Stochastic Transition Systems* (MLSTS) was presented [6, 11] as a new semantic model for characterizing the functional and performance properties of concurrent systems, under the assumption of arbitrarily distributed (i.e. non-Markovian) durations of actions. The MLSTS models can be automatically generated from S-LOTOS specifications according to the (true concurrency) maximality semantics [6]. The main advantage is to pruning the state graph without loss of information w.r.t. ST-semantic models [11]. As a first work on MLSTS, we focus in this paper on in the verification of functional properties of systems, using a variant of model-checking technique.

Keywords: CTL, Formal Verification, Maximality Semantics, Model-Checking, Semantic Models, Labeled Transition Systems.

1 Introduction

Since the use of concurrent and distributed systems has become more and more important in industry, the analysis, prediction and evaluation (of both qualitative and quantitative aspects) of their behavior have become mandatory. However, in concurrent systems multiple active agents interact together and with the environment, and this leads to phenomena like uncertainty, non-determinism and randomness in the global behavior of the system, and exhibit complex functional and temporal behavior that are often complex to predict. For instance, in transmission systems, concurrency and transmission errors in the traffic flow produced randomly, lead to various communication delays. Therefore, there is a need for adequate stochastic timing models to well specify and verify such (stochastic) behaviors.

Actually, two main approaches have been adopted for expressing random time properties of stochastic systems. In the first one, e.g. [1, 9, 13], the specification of the durations of actions is limited to exponential distributions, hence, it takes advantage from the memoryless property of exponential distributions, which yields analytically tractable models in the form of Continuous Time Markov Chains [2, 7]. Such models accord with the interleaving semantics [1, 9], therefore actions are considered as atomic (see Fig. 1-a), and the parallel execution of two actions is assumed to be equivalent to their interleaving execution.

A. Amine et al. (Eds.): *Modeling Approaches and Algorithms*, SCI 488, pp. 367–377.
DOI: 10.1007/978-3-319-00560-7_40

For responding to the limitation in expressiveness of the exponential distribution laws and the interleaving semantics imposed by the first approach, the second approach adopts general probability distributions to specify action durations, and it refers to the true concurrency semantics [10]. This last one allows to escape the action atomicity hypothesis imposed by the interleaving semantics. Thus, the system behaviors are not anymore represented like totally ordered sequences, but adequately like partial order ones. In fact, existing non-Markovian approaches, e.g. [10, 14, 15, 16, 17], introduce an explicit representation of the start and end events for every running actions (for example, see Fig. 1-b, where a^+ and a^- represent respectively the events of start and termination of execution of the action a). This allows considering a specific true concurrency semantics called ST-Semantics. In the ST-semantic models, the progression of a delay is represented as a combination of two events: the delay starting and termination. However, the price to pay is the generation of models which suffer from the state space explosion problem, due to the splitting of running actions into start and end events.

For our approach, we refer to an appropriate true concurrency semantics, namely Maximality semantics [3, 4]. Its principle consists in using the dependence relations between actions occurrences and by associating to every state of the system the set of actions which are potentially in execution (see Fig. 1-c).

Our *integrated* approach *based on MLSTS* aims at handling true concurrency notions and arbitrarily distributed durations for specifying functional and performance properties, without suffering from the state space explosion problem inherent to the splitting of actions. We interest in this paper in the concurrency logic verification, based on our MLSTS model, of functional aspects of concurrent systems. Along this paper, we assume that the reader is familiar with Labeled Transition Systems, Model-Checking, formal description technique LOTOS [8] and the temporal logic CTL.

The paper is scheduled as follows: In section 2, the main principles and formal definition of the MLSTS models are presented. In section 3, we present our approach for functional aspect verification based on MLSTS models by adopting a CTL model checking variant. Finally, section 4 concludes the paper and opens some perspectives.

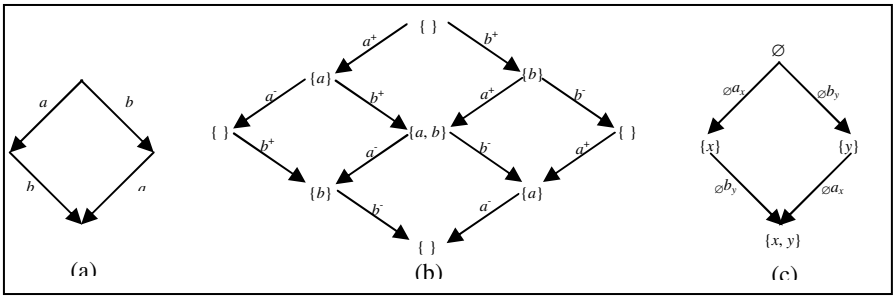

Fig. 1. Behavior of two parallel actions according to the interleaving semantics (a), the ST-semantics (b) and the Maximality semantics (c)

2 Maximality-Based Labeled Stochastic Transition Systems

The aim of our approach is to combine functional and performance modeling and analysis, and allow any kind of probability distribution function for specifying action durations. From a modeling views point, such an approach showed interest in easing the modeling stages. The main interest of our maximality based approach is to make possible to deal with true concurrency with less prone to state space explosion problems [4, 6].

2.1 Informal Presentation of MLSTS

Within the semantic model MLSTS, each transition only represents the start of an action execution. Since actions are not considered as atomic, the concurrent execution of multiple actions can be represented, and distinguishing between sequential and parallel executions is possible.

The running actions are represented at the states level, and each instance of running actions is called a maximal event and is identified by a distinct name. In fact, each state of the system is featured by a unique configuration. The configuration of a state s is denoted ${}_{M}[E]$, such that M is the set of maximal events and E is the behavior expression in the state s. Every transition defined from s is labeled by ${}_{C}(a, f)_{x}$ whenever a is an action that can be activated from E iff. the maximal events of the subset $C \subseteq M$ are terminated. Further C is called the causality set of the transition, and x is the name identifying the start event of the new execution of a. The event identification is required to avoid confusion since several instances of running actions can have the same action name.

Fig. 2 illustrate by a simple example that MLSTS models allow distinguishing between concurrent and sequential executions of actions. Consider tow actions a and b (whose durations follow probability distribution functions f and g respectively), and the following two behavior expressions:

E= (a,f); *stop* |[]| (b,g); *stop*
F= (a,f); (b,g); *stop* [] (b,g); (a,f); *stop*.

Such that E executes a in parallel with b, and F executes either a followed by b or b followed by a.

Initially, no action has yet been executed, then the set of maximal events is empty, and the initial configurations associated with E and F are, respectively, ${}_{\varnothing}[E]$ and ${}_{\varnothing}[F]$. From configuration ${}_{\varnothing}[E]$ the actions a and b can be executed independently (according to the semantic of the parallel composition operator |[]|), and the both execution paths lead to the configuration ${}_{\{x\}}[stop]$ |[]| ${}_{\{y\}}[stop]$, where x and y are the event names identifying the starting of a and b respectively. However, from configuration ${}_{\varnothing}[F]$, depending on the action which happens first (for example the action a), and according to the semantic of the prefix operator (;) expressing the sequentiality in execution, the start of the other action (i.e. the action b) is constrained by the causality

dependence against the event identifying the potential execution of the first one (i.e. the action *a*). This results in the following execution path:

$$\varnothing[F] \xrightarrow{\varnothing(a,f)_x} {}_{\{x\}}[(b, g); stop] \xrightarrow{{}_{\{x\}}(b,g)_y} {}_{\{y\}}[stop]$$

In a symmetric scenario, if the action *b* happens first, the behavior can be represented by the following execution path:

$$\varnothing[F] \xrightarrow{\varnothing(b,g)_y} {}_{\{y\}}[(a, f); stop] \xrightarrow{{}_{\{y\}}(a,f)_x} {}_{\{x\}}[stop]$$

The MLSTSs representing the behaviors of *E* and *F*, obtained by applying the maximality semantics, are represented in **Fig. 2**. It is clear that from **Fig. 2**-a and **Fig. 2**-b the two behaviors of *E* and *F* are not equivalent.

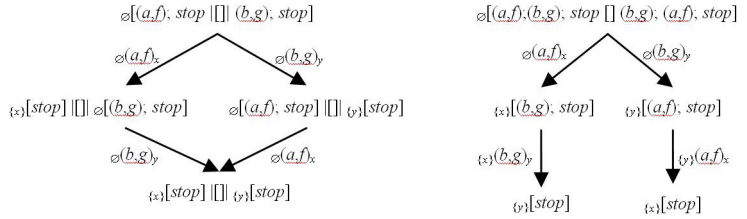

Fig. 2. MLSTSs representing behaviors of concurrent and sequential executions

2.2 Formal Definition of MLSTS

An MLSTS is defined as follows:

Definition 1. Maximality-based Labeled Stochastic Transition System (MLSTS).
Let $\mathcal{M}$ be a countable set of event names.
An MLSTS is a structure (Ω, *A*, *DF*, *L*, μ, ξ, ψ) with:

- $\Omega = (S, s_0, T, \alpha, \beta)$: is a Transition System s.t. *S* is the countable set of states for the system, at least including the initial state s_0; *T* is the countable set of transitions specifying the states changes; α and β are two functions: $T \rightarrow S$ mapping every transition with its source $\alpha(t)$ and its target $\beta(t)$.
- *A* : is a (finite) set of actions.
- *DF* : is a finite set of probability distribution functions ($\Re \rightarrow [0, 1]$).
- L: $T \rightarrow (A \times DF)$: is a function which associates to each transition a pair composed of an action and a probability distribution function specifying the action duration.
- $\psi : S \rightarrow 2^{\mathcal{M}}_{fn}$: is a function which associates to each state the finite set of maximal event names in this state.
- $\mu : T \rightarrow 2^{\mathcal{M}}_{fn}$: is a function which associates to each transition the finite set of maximal event names of actions that have started their execution so that their terminations allow the start of this transition (i.e. the direct causes of the transition).

- $\xi : T \rightarrow \mathcal{M}$: is a function which associates to each transition an event name identifying its occurrence, such that for any transition $t \in T$:

$\mu(t) \subseteq \psi(\alpha(t))$
$\xi(t) \notin \psi(\alpha(t)) - \mu(t)$
$\psi(\beta(t)) = (\psi(\alpha(t)) - \mu(t)) \cup \{\xi(t)\}$ ♦

MLSTS is intended for modeling both qualitative (functional) and quantitative (stochastic temporal) behavior. Moreover, it is able to deal, in a true concurrency semantics, with any kind of probability distribution instead of restricting only to the exponential distributions. We also defined a Stochastic Process Algebras (SPA), called S-LOTOS [12], as a language to describe MLSTS. From S-LOTOS specifications, the underlying MLSTS models can be generated automatically, according to the maximality semantics [6].

3 Functional Aspect Verification Based on MLSTS

From an MLSTS of a given system, one can derive two semantic models [12]: a functional one enhancing true concurrency behaviors, and a performance one allowing quantitative evaluation. The functional model is obtained by abstracting the quantitative information related to the various durations of actions, whereas the performance model is obtained by abstracting the functional information. We showed in [12] that the performance model of an MLSTS is a Generalized Semi-Markov Process (GSMP). An interest for S-LOTOS is that it can be considered as a high level formalism for GSMP, and further analyses over the GSMP structures can yield performances evaluations. In this section, we focus in the functional aspect verification based on MLSTS models, using a variant of model checking approach resumed in Fig. 3.

In this approach (based on models), the system to be verified is firstly specified by means of the S-LOTOS language [12]. Next, the specification will be translated in an operational way towards the underlying MLSTS model [6, 11]. The expected properties of good behaviors of the system are written in CTL (Computation Tree Logic) [18], and they are finally verified by means of model-checker.

In spite of temporal logics facilitation of specification of system properties to be verified [5], model checking approach is limited by the state space explosion problem, particularly when the specification model underlying semantics is the ST-semantic one. A priori, the maximality transition relation appears to be more complicated than that of interleaving semantics or ST-semantic, because supplementary information is associated to states and transitions. However, this information can allow more reductions without loss of (qualitative and quantitative) information w.r.t. the ST-semantic models [11]. To benefit from the expression power of the MLSTS, in this paper, we present, as a first step in the analysis of this model, the application of the model checking technique to verify behavioral properties.

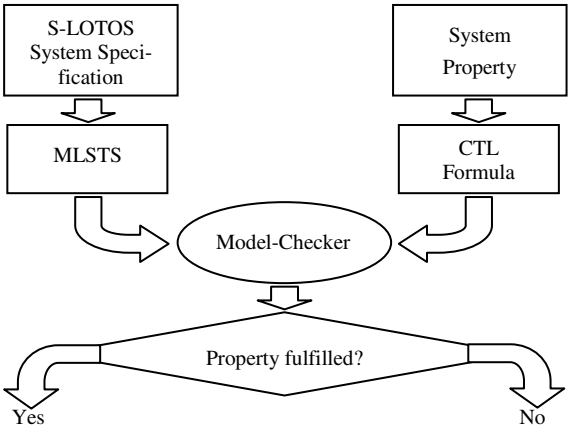

Fig. 3. Model checking based verification approach

3.1 CTL Model-Checking

Qualitative properties of concurrent systems refer to their behaviors, that is to say the sequences of states or actions generated during their executions. Temporal logics are well suited to specify these properties; because they allow to obtain abstract and modular specifications of systems (i.e. specifications are independent from any implementation and easily changeable).

Many temporal logics were defined and studied in the literature. In our approach, we adopt the Computation Tree Logic (CTL) as language to define the properties to be verified, mainly because the evaluation of CTL formulae is more efficient. In fact, classical algorithms are polynomial in the size of the model and also in the size of the formula [18, 19].

CTL is a branching time temporal logic widely used in the model-checking verification techniques. CTL contains the usual logical operators ($\neg$, $\vee$, $\wedge$, $\Rightarrow$ and $\Leftrightarrow$), and the usual temporal operators: X (Next: The X operator in "X ϕ" means that ϕ has to hold at the next state), F (Finally: The F operator in "F ϕ" means that ϕ eventually has to hold somewhere on the subsequent path), G (Globally: The G operator in "G ϕ" means that ϕ has to hold on the entire subsequent path), and U (Until: The U operator in "ϕ U φ" means that ϕ has to hold at least until at some position φ holds), whom have to be at once preceded by one of the path quantifiers: A (along All paths : The A operator in "A ω" means that ω has to hold on all paths starting from the current state.), and E (there Exists one path: The E operator in "E ω" means that there exists at least one path starting from the current state where ω holds).

In CTL, the operators must always be grouped in two: one path quantifier followed by a temporal operator. For example, "AG *p*" is satisfied in a state if for all paths from this state, *p* is always true. Thus, we can distinguish eight basic operators: AX, EX, AG, EG, AF, EF, AU and EU. Using these operators, CTL can express formally behavioral properties of concurrent systems, including principally safety and liveness properties.

Definition 2. The syntax of CTL formulae is as follows:

$\varphi ::= true \mid false \mid p \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \Rightarrow \varphi) \mid (\varphi \Leftrightarrow \varphi) \mid$
$AX\ \varphi \mid EX\ \varphi \mid AF\ \varphi \mid EF\ \varphi \mid AG\ \varphi \mid EG\ \varphi \mid A\ [\varphi\ U\ \varphi] \mid E\ [\varphi\ U\ \varphi]$

Where $p \in AP$ is an atomic proposition. ♦

Obviously, several model-checkers were developed in the literature. The global state graph of the system can be generally viewed as a finite Kripke structure, wherein each state is labeled with a set of atomic propositions true in that state, and an efficient algorithm is given to determine whether a structure is a model of a particular formula.

3.2 MLSTS Based Qualitative Verification

In MLSTS models, the information included in the states represents the actions that are potentially in execution. For this fact, one can express belonging properties such as mutual exclusion in a more natural way, as well as from new properties that concern actions and their concurrent execution. The expression of these properties does not require the use of a new logic or the introduction of a new operators since one can use CTL temporal logic and consider actions in states as being atomic propositions. However, what changes is the intuition behind formulae. For example, the formula "EF ($a \wedge b$)", where a and b are names of actions, means that there is at least a path which leads to a state where parallel execution of a and b can take place. In a similar way, one can explain intuitively all the formulae of the CTL logic that may be checked using MLSTS model as follows:

- "$a \wedge b$" in a state s means that a and b can be executed in parallel in the state s.
- "$\neg a$" in a state s means that the execution of a cannot take place in the state s.
- "EX a" in a state s_0 means that there is at least a path (s_0, s_1, …) where a will be able to comply in the state s_1.
- "AX a" in a state s_0 means that for any path (s_0, s_1, …), a may be executed at state s_1.
- "E [a U b]" in a state s_0 means that there is a path (s_0, …, s_k, …), where b will be able to comply in the state s_k and a will be able to comply in every state of this path that precedes the state s_k.
- "A [a U b]" in a state s_0 means that for any path (s_0, …, s_k, …), there is a state s_k in this path where b may be comply and a may be comply in every preceding state.

Therefore, to express behavioral properties, it is not necessary to use logical formulae indicating the state of evolution of a process, we only would reason directly about actions. By proceeding so, properties will be easier to express and their meaning seems more natural. One can express new properties such as specifying actions incompatibility, we may express that a and b are incompatible by "AG$\neg$ ($a \wedge b$)" which means that the actions a and b will never be able to be executed concurrently. If one takes the example of a simple system of concurrent readers and writers of a shared variable, the mutual exclusion between readers and writers can be simply expressed

by "AG ¬(*write* ∧ *read*)", where *write* is the action of writing on the variable by a writer process, and *read* is the action of reading the variable by a reader process.

Moreover, in the MLSTS structure, every action is associated with an event name that allows distinguishing between multiple executions of the same action at any state (auto-concurrency). Considering this point will allow us to reason about the number of parallel execution of an action at any state, in other words, we may verify the degree of the auto-concurrency in a system. For this aim, one will have the form of proposition "*a*:*n*" (where *n* is a positive natural number) to express the fact that there is *n* parallel execution of the action *a*. Hence, one can express new properties, for instance, "AG ¬(*write*:2)" expresses that along all possible executions, two writing actions may not be in execution simultaneously.

3.3 Model Checking Algorithm

In this section, we adapt the standard CTL model-checking algorithm, e.g. [18], to our study for properties verification on MLSTS models by considering actions at MLSTS states as atomic propositions.

Let us suppose that one has an MLSTS model $M = (\Omega, A, DF, L, \mu, \xi, \psi)$ and a CTL formula φ. The purpose is to compute (recursively) states *s* of *M* satisfying φ, we call the set of these states *Sat*(φ). Finally, the (model of the) system satisfies the desired property, denoted by $M \models \varphi$, if the initial state $s_0 \in Sat(\varphi)$. The algorithm presented in **Fig. 4** treat only the formulae of the form (φ' ∧ φ"), (¬φ), EX φ, E[φ' U φ"], A[φ' U φ"] which are sufficient. It may be noted that the other logical and temporal operators are implicit and defined as much as the following abbreviations:

$\varphi' \vee \varphi'' \equiv \neg (\neg\varphi' \wedge \neg\varphi'')$
$\varphi' \Rightarrow \varphi'' \equiv \neg \varphi' \vee \varphi''$
$\varphi' \Leftrightarrow \varphi'' \equiv (\varphi' \Rightarrow \varphi'') \wedge (\varphi'' \Rightarrow \varphi')$
$AX\ \varphi \equiv \neg EX \neg\varphi$
$EF\ \varphi \equiv E\ [true\ U\ \varphi]$
$AF\ \varphi \equiv A\ [true\ U\ \varphi]$
$EG\ \varphi \equiv \neg AF \neg\varphi$
$AG\ \varphi \equiv \neg EF \neg\varphi$

We only consider the case in which φ = A[φ' U φ"] here, since all the other cases are either straightforward or similar. For this case, one will need information about successor states of *s* as well as on the state *s* itself, because A[φ' U φ"]= φ" ∨ (φ' ∧ AX A[φ' U φ"]). Initially, A[φ' U φ"] is added to all the states already labeled by φ". Then, A[φ' U φ"] will be propagated and added to any state labeled by φ' having all successor labeled by A[φ' U φ"]. In the same way one may argues for E[φ' U φ"]. We also take into account the case where φ has the form (*a*:*n*), in this case, a state *s* satisfies φ if it is labeled by *n* event names representing the execution of the action *a*. we use the function *act* which has in input an event name *x* and a state *s*, and returns the action to which is associated the event name *x* in this state.

In the worst case, this algorithm version require $O(length(\varphi) * card(S)^2 * card(T))$ running time. Therefore, this CTL model-checker has a linear temporal complexity according to the length of formula to be verified and quadratic complexity according to the size of the structure *M*.

Function
Input :
A CTL formula φ,
An MLSTS structure $M=(\Omega, A, DF, L, \mu, \xi, \psi)$, with S is the set of states for the system, T is the set of transitions, and ψ is the function that associates every state with a finite set of maximal event names in the state.
Output : The set $Sat(\varphi)$ of all states satisfying formula φ.

```
Begin
  Initialize Sat(φ) by ∅
  Case φ of :
    φ = a (atomic proposition representing that
          the action a is in execution) :
      for all s ∈ S do
        if ∃ x ∈ ψ(s) / act(x, s) = a
          add s to Sat(φ)
        endif
      endfor
    φ = φ' ∧ φ'' :
      Compute Sat(φ')
      Compute Sat(φ'')
      for all s ∈ S do
        if s ∈ Sat(φ') and s ∈ Sat(φ'') then
          add s to Sat(φ)
        endif
      endfor
    φ = ¬φ':
      Compute Sat(φ')
      for all s ∈S do
        if s ∉ Sat(φ') then
          add s to Sat(φ)
        endif
      endfor
    φ = EX φ':
      Compute Sat(φ')
      for all s ∈S do
        if ∃ successor s' of s / s' ∈ Sat(φ') then
          add s to Sat(φ)
        endif
      endfor
    φ = A [φ' U φ'']:
      Compute Sat(φ')
      Compute Sat(φ'')
      for all s ∈S do
        if s ∈ Sat(φ'') then
          add s to Sat(φ)
        endif
      endfor
      for j = 1 to Card(S) do
        for all s ∈S do
          if s ∈ Sat(φ') and if ∀ successor s'
of                  s / s' ∈ Sat(φ) then
            add s to Sat(φ)
          endif
        endfor
      endfor
    φ = E [φ' U φ'']:
      Compute Sat(φ')
      Compute Sat(φ'')
      for all s ∈S do
        if s ∈ Sat(φ'') then
          add s to Sat(φ)
        endif
      endfor
      for j = 1 to Card(S) do
        for all s ∈S do
          if s ∈ Sat(φ') and if ∃ successor s'
of                  s / s'∈ Sat(φ) then
            add s to Sat(φ)
          endif
        endfor
      endfor
    φ = (a:n),
      for all s ∈ S do
        if ∃ x1, x2,… xn, / x1, x2,… xn ∈ ψ(s) and
          act(xi, s)=a, 1≤i≤n, then
          add s to Sat(φ);
        endif
      endfor
  endcase
  return (Sat(φ))
end
```

Fig. 4. MLSTS based Model-checking algorithm

4 Conclusion

Considering functional correctness and performance evaluation in a common framework is desirable. Our integrated approach is based on an algebraic formalism for specifying functional and performance properties of concurrent systems, which lies in the class of Stochastic Process Algebras. The main advantage is the capability of specifying and modeling, in the same approach, stochastic concurrent systems under the assumption of generally distributed durations of actions, and reducing, relatively, the state space models, w.r.t. standard true concurrency ST-semantic models.

In this paper, we are interested in the functional verification part, we present the possibility of applying the model-checking approach on our MLSTS model for verifying functional behavior and concurrency properties. This contribution was mainly led by the presence of information about actions potentially in execution in states.

As perspective of this work, we can intend, on one hand, to apply our approach for the study of concrete applications of, for instance, communication protocols domain. Moreover, the final goal of our work is to give a unifying framework for functional verification and performance evaluation. We plan to complete this work by improving (existing) tools to investigate the problems of performance verification.

References

[1] Clark, A., Gilmore, S., Hillston, J., Tribastone, M.: Stochastic Process Algebras. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 132–179. Springer, Heidelberg (2007)

[2] Haverkort, B.R.: Markovian models for performance and dependability evaluation. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) FMPA 2000. LNCS, vol. 2090, pp. 38–83. Springer, Heidelberg (2001)

[3] Saïdouni, D.E., Courtiat, J.P.: Prise en compte des durées d'action dans les algèbres de processus par l'utilisation de la sémantique de maximalité, In. In: Proceedings of CFIP 2003, Hermes, France (2003)

[4] Saidouni, D.E.: Sémantique de maximalité: Application au raffinement d'actions en LOTOS, PhD Thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077. Toulouse Cedex France (1996)

[5] Lamport, L.: What good is temporal logic? In: Manson, R.E.A. (ed.) Information Processing. IFIP, vol. 83, pp. 657–668. Elsevier Science Publishers B.V., North Holland (1983)

[6] Arous, M., Ilié, J.M., Saidouni, D.E.: A Compact Semantic Model for Characterization of Stochastic Temporal Properties of Concurrent Systems. IJCSI International Journal of Computer Science Issues 96(1) (November 2012)

[7] Stewart, W.J.: Performance modelling and markov chains. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 1–33. Springer, Heidelberg (2007)

[8] Bolognesi, T., Brinksma, E.: Introduction to the ISO specification language LOTOS. Computer Networks and ISDN Systems 14, 25–59 (1987)

[9] Hillston, J.: Process Algebras for Quantitative Analysis. In: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005), pp. 239–248 (2005)

[10] Katoen, J.-P., D'Argenio, P.R.: General distributions in process algebra. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) FMPA 2000. LNCS, vol. 2090, pp. 375–429. Springer, Heidelberg (2001)

[11] Arous, M., Saidouni, D.E., Ilié, J.M.: Addressing State Space Explosion Problem in Performance Evaluation Using Maximality-based Labeled Stochastic Transition Systems. In: IPCSIT, vol. 54, pp. 41–48. IACSIT Press, Singapore (2012); Proceedings of the 2nd International Conference on Computer and Software Modeling (ICCSM 2012), Cochin, India

[12] Arous, M., Saidouni, D.E., Ilié, J.M.: Maximality Semantics based Stochastic Process Algebra for Performance Evaluation. In: 1st IEEE International Conference on Communications, Computing and Con-trol Applications (CCCA 2011), Hammamet, Tunisia, March 3-5 (2011)

[13] Bernardo, M.: Theory and Application of Extended Markovian Process Algebra. Ph.D. Thesis, University of Bologna, Italy (1999)
[14] Bravetti, M., Gorrieri, R.: The theory of interactive generalized semi-Markov processes. Theoretical Computer Science 282, 5–32 (2002)
[15] Bravetti, M., Bernardo, M., Gorrieri, R.: Towards performance evaluation with general distributions in process algebras. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 405–422. Springer, Heidelberg (1998)
[16] Bravetti, M.: Specification and Analysis of Stochastic Real-time Systems. PhD thesis, Università di Bologna, Padova, Venezia (2002)
[17] D'Argenio, P.R., Katoen, J.-P.: A theory of stochastic systems. Part II: Process algebra. Information and Computation 203, 39–74 (2005)
[18] Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. ACM Transactions on Programming Languages and Systems 8(2), 244–263 (1986)
[19] Schnoebelen, P.: The Complexity of Temporal Logic Model Checking. In: Advances in Modal Logic, pp. 393–436 (2002)