Towards the Integration of Real-Time and Probabilistic-Time Process Algebras

Mario Bravetti

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: bravetti@cs.unibo.it

Abstract

A formal approach for the design and analysis of concurrent systems is proposed which integrates two different orthogonal aspects of time: the aspect of real-time, concerning the expression of time constraints and the verification of exact time properties, and the aspect of probabilistic-time, concerning the probabilistic quantification of non-deterministic durations and the evaluation of system performance. We show that these two aspects (that led to the introduction of different specification languages called real-time and stochastic process algebras) can be expressed by a single language, namely a stochastic process algebra capable of representing general distributions of durations. Introducing such an algebra gives the opportunity of testing the system for mixed probabilistic-time and real-time properties. Moreover a pure real-time model and a pure performance model which are guaranteed to be consistent can be formally derived from the initial model of the system.

1 Introduction

The importance of considering the behavior of concurrent systems with respect to time during their design process has been widely recognized. In particular two different approaches for expressing and analyzing time properties of systems have been developed which are based on formal description languages.

A first approach concentrates on the aspect of real-time, i.e. the expression of time constraints and the verification of exact time properties (see [1, 12] and the references therein). Via this approach the parts of the system that are critical from the viewpoint of time bounds can be validated during the design phase through techniques as e.g. model checking [1]. In this view the real-time process algebras have been developed by extending the standard process algebras with the representation of time and, typically, by introducing some time-out operator (see e.g. [12]).

A second approach is devoted to the evaluation of the performance of concurrent systems (see [5] and the references therein). According to this approach the time spent by a system in a certain activity is expressed probabilistically through a distribution of duration. Performance measures of systems can then be evaluated via mathematical or simulative techniques. This approach has led to the definition of stochastic process algebras, an extension of standard process algebras where a distribution of duration is associated with each action of a process. In most cases, as in [5], the expressiveness of such algebras is limited to exponential distributions of time, because this causes the passage of time to be "memoryless" and allows to avoid completely from representing durations in semantic models. Moreover the limitation to exponential distributions allows a straightforward transformation of the semantic model of a system into a Markov Chain, a stochastic process which is easily mathematically analyzable for deriving performance measures. It is worth noting that the limitation imposed over durations is very strong because not even deterministic (fixed) durations can be expressed.

The different aspects of time expressed by these two approaches can be seen as being orthogonal. According to the first approach some interval of time is definable for doing something, but the actual time the system spends in-between interval bounds is expressed non-deterministically. According to the second approach the possible values for the duration of an activity are quantified through probabilistic distributions, but no time constraint is expressible (all duration values are possible with probability greater than zero).

A specification language capable of expressing both aspects of time should be able of expressing both time constraints and a probabilistic quantification for the possible durations which satisfy such constraints. Such an expressive power is obtained by considering a stochastic process algebra capable of expressing general probability distributions for the duration of activities. In this way time constraints are expressible via probability distribution functions that associate probability greater than zero only to time values that are possible according to the constraints. In particular such a stochastic process algebra gives the possibility of expressing deterministic durations via trivial distribution functions that give all the probability to a single value of time.

We have developed an extension to the standard approach of stochastic process algebras which allows to express general distributions [3, 4] that has led to the definition of Generalized Semi-Markovian Process Algebra (GSMPA). The name of the algebra derives from the class of "Generalized Semi-Markov Processes" [11] that are stochastic processes (representing the performance of systems specified with GSMPA) capable of expressing general distributions.

In the light of the previous observations we can see GSMPA as a core language which integrates the expressiveness of real-time and probabilistic-time process algebras. Whilst in [3, 4] we have concentrated on performance evaluation only, in this paper we will develop the idea of integrating the real-time and probabilistic-time approaches for modeling and analyzing concurrent systems.

The semantics of GSMPA has been defined so as to obtain semantic models similar to GSMPs. In particular when general distributions are considered we can no more rely on a tricky "memoryless time". In order to describe correctly the parallel execution of actions at the semantic level, we have to represent actions with an actual duration, i.e. actions that start in a certain state, evolve through several states and terminate in another state. Therefore, an action can no longer be considered as being executed atomically in a single transition, but is characterized in the semantics by the two events of *action start* and *action termination* where the termination of an action is uniquely related to its start. A semantics which represent action execution in this way is the ST semantics (see e.g. [9, 2]). The ST idea of representing durational actions has turned out to be useful to express the semantics of languages with time also in other contexts, e.g. [14] argues that ST equivalence is the correct equivalence for comparing the time behavior of concurrent systems in an asynchronous setting. The semantics of GSMPA has been defined as a variant of the ST semantics especially made for expressing probabilities for choices and durations.

Representing the real-time and probabilistic-time in a single language allows to model a concurrent system more precisely by expressing and analyzing the relationships between the two aspects of time.

In [3] a notion of bisimulation based equivalence has been defined for GSMPA which is a congruence with respect to all the operators of the language. Thus we can verify if a system meets a specification through equivalence checking accounting for both aspects of time.

Moreover in this paper we show that we can develop a suitable testing scenario (see [8]) which reveals both the real-time and the probabilistic-time behavior of the system. This approach generalizes previous results in the context of (non-deterministic) timed models [14] and probabilistic models [7]. Through testing we can compare systems by considering at the same time both their behavior and their efficiency.

Besides we show that we can analyze a system for properties expressed via a real-time logic (that of [1]) extended by generalizing the requirement for the possibility of doing something with the requirement for a certain minimal probability. In this way we obtain a generalization of the standard approach of real-time for the validation of a system against a specification. Besides, since our models are fully probabilistic, the new possibility is introduced (see [10]) of verifying properties through testing, instead of, e.g., model checking.

Finally, with a single language, pure performance and real-time tests must no longer be performed on two different models of the same system that are not guaranteed to be consistent. With our approach a pure performance model and a pure real-time model can be formally derived from the initial model of the system. The derived models can, then, be analyzed by employing the techniques and tools already developed.

The paper is organized as follows. In Sect. 2 we briefly describe the stochastic process algebra GSMPA. In Sect. 3 we introduce some analyzing methodologies for GSMPA. Finally, in Sect. 4 we report some concluding remarks.

2 Generalized Semi Markovian Process Algebra

In this section we give a description of the main ingredients of GSMPA. For a complete and formal presentation see [3].

2.1 Specification of a Concurrent System

In GSMPA the specification of a concurrent system is composed of the specification of its behavior (a process of GSMPA) and the specification of the duration of the actions it may execute. The separation of the two specifications reflects the same separation at the semantic level stemming from the structure of GSMPs.

2.1.1 Specification of System Behavior

The specification of system behavior consists of a process term of GSMPA and a set of constant definitions of the form $A \triangleq E$ where A is a constant and E is a term. In the following we present the syntax and informal semantics of GSMPA process terms.

In GSMPA an *action* is represented as $<a^v, w>$ and consists of a *type* a, a *visibility* v and a *weight* w. The set of action *types* is $AType$, ranged over by $a, b, c, \ldots$ The *visibility* v of an action specifies if the action must be executed internally or may synchronize with other actions. We define the set $AVis$ of *action visibilities*, ranged over by v, by $AVis = \{o, h\}$ where o stands for *observable* and h stands for *hidden*. The *weight* $w \in \mathbb{R}_+$ of an action is used for choosing among alternative actions. When multiple actions are executable (due e.g. to an alternative composition operator "+") a probabilistic choice is made by giving each action a probability proportional to its weight.

Let $Const$ be a set of *constants* ranged over by $A, B, \ldots$, and let $ARFun = \{\varphi : AType \longrightarrow AType\}$ be a set of *action relabeling functions*. The set of *process terms* of GSMPA is generated by the following syntax:

$$E ::= \underline{0} \mid <a^v, w>.E \mid E/L \mid E[\varphi] \mid E + E \mid E \parallel_S E \mid A$$

where $a \in AType$; $v \in AVis$; $w \in \mathbb{R}_+$; $L, S \subseteq AType$; and $A \in Const$. "$\underline{0}$" is the terminated process; ".", "+" and "$[\varphi]$" are the prefix, choice and relabeling operators of CCS; "$/L$" and "$\parallel_S$" are the hiding and parallel (where the actions in "S" are required to synchronize) operators of CSP. Some differences are that: "+" expresses a probabilistic choice which is resolved according to the the weights of the actions executable by the two terms; "$[\varphi]$" modifies also the type of hidden actions; "$/L$" turns the visibility of the actions whose type belong to L into h; and "$\parallel_S$" assigns to an action obtained by synchronization the *product* of the weights of the actions that synchronize, so that in each side of "$\parallel_S$" an independent choice is made for choosing the synchronizing action.

2.1.2 Specification of Durations

Durations of actions are specified by a function $PDF_{pid} : AType \longrightarrow Dist$ assigning to each local action of a sequential process (identified by specifying its *location*, i.e. the position with respect to the parallel composition operators) a duration distribution. The syntax of process identifiers is:

$$pid ::= \epsilon \mid \swarrow pid \mid \searrow pid$$

where $\swarrow$ denotes a process to the left of a "$\parallel_S$" operator and similarly for $\searrow$. For instance in $(E_1 \parallel_S E_2) \parallel_{S'} E_3$, the process identifier $\swarrow\searrow$ denotes the process E_2. For actions obtained by synchronization, the duration is described by specifying a composition rule $PDFSynch_a : (Dist \times Dist) \longrightarrow Dist$ for evaluating the duration distribution of an action derived by the synchronization of two actions of type a.

2.2 Semantics for GSMPA

The semantics of a GSMPA specification parallel its structure and consist of a labeled transition system representing the system behavior and a function associating with each action of the whole system (including actions obtained by synchronization) a duration distribution. Actions are identified at the semantic level through their location, i.e. the position with respect to the parallel composition operator. We express the location of an action obtained by synchronization by composing the locations of the synchronizing actions. The syntax of *action locations* is:

$$loc ::= \epsilon \mid \swarrow loc \mid \searrow loc \mid \langle loc | loc \rangle$$

where $\langle loc|loc \rangle$ represents action synchronization. An *action identifier* is composed by a type and a location. For example in $(a.E_1 \parallel_\emptyset b.E_2) \parallel_{\{b\}} b.E_3$, the identifier of the action formed by the synchronization of the two actions of type b is $b_{\langle \searrow | \rangle}$.

2.2.1 Representation of System Behavior

The labeled transition system (LTS) is obtained through operational structured semantics whose inference rules here are omitted (see [3]). Conceptually at the semantic level the execution of an action gives rise to the two events of *action start* and *action termination* that are represented by two type of transitions in the LTS. The event of an action termination is uniquely related to the corresponding event of action start through the action location. The states of the LTS are pairs $\langle E, Exec \rangle$, consisting of a GSMPA term E and a label $Exec$. $Exec$ is the set of identifiers of the actions in execution, i.e. actions that have already started, but have not terminated yet. This makes the semantics of GSMPA very similar to a GSMP where states are labeled with activities in progress.

In order to obtain a semantics adherent to a GSMP we have to extend the ST semantics by describing the system behavior also in the case of contemporaneous termination of actions. This can be done very naturally by defining *termination transitions* as in the step semantics [13].

The last enhancement of the ST semantics we have to make regards probabilistic choices. In GSMPA choices are solved through the *preselection policy*. Each transition representing the event of an action start is labeled with the *weight* of the action and is called a *choice transition*. Choices among such transitions are carried out by giving each of them a probability proportional to its weight.

2.2.2 Durations of Actions

The function that assigns to each action identifier its distribution of duration is determined as follows. The duration distribution of an action of type a obtained by synchronization is computed starting from the distribution of synchronizing actions, (given at the level of the sequential process pid by $PDF_{pid}(a)$ defined in the specification) and the composition rule $PDFSynch_a$ also defined in the specification (see [3] for a formal definition).

2.3 Equivalence for GSMPA

A notion of equivalence has been defined for GSMPA which combines a bisimulation for ST semantics in the form of that presented in [2] with the notion of *probabilistic bisimulation* introduced in [10]. See [3] for a formal definition. We have proven the equivalence to be a congruence with respect to all the the operators of GSMPA, including recursion.

3 Analyzing Methodologies for GSMPA

In this section we will introduce some methodologies for analyzing concurrent systems which exploit the capability of GSMPA to express both real-time and probabilistic-time aspects. We want to point out that the equivalence notion for GSMPA provides already the language with a methodology to analyze both real-time and probabilistic-time aspects. This because through the method of equivalence checking it is possible to verify whether a concurrent system meets a specification in the case when the specification is a process term as well.

3.1 Probabilistic Real-Time Testing

First of all we can study the behavior of a system with respect to both aspects of time with the standard approach of testing [8]. With this approach the behavior of a system P is observed through a test, consisting of another process Q. The test Q includes a special action ω representing the success of the test. A test is applied to a process P by performing $P \|_S Q$ where S, the set of actions that must synchronize, includes all actions except ω. We have that: P *may* satisfy the test if $P \|_S Q$ has a system run where ω is performed, P *must* satisfy the test if $P \|_S Q$ performs ω in all system runs. With this approach we can compare systems by means of the set of tests they satisfy. We can define a preorder among systems such that $P \leq R$ if and only if for both "must" and "may" notion of satisfaction, all test satisfied by P are satisfied also by R. The definition of a preorder leads to a methodology for the analysis of concurrent systems called preorder checking. Through the method of preorder checking it is possible to verify whether a concurrent system R meets a specification P, which is a process term as well. This is done by checking if $P \leq R$, where P is treated as the minimal requirement to be met and may contain do not care points. Moreover the testing approach leads also to a notion of equivalence: we have that P is equivalent to R if $P \leq R$ and $R \leq P$, i.e. if they satisfy the same tests.

In [14] a timed version of the testing approach is developed which introduces the notion of test as a pair composed of an environment Q and a time bound l. For a formalism capable of expressing (non-deterministic) time, a test (Q, l) is satisfied by P if success is reached after at most time l. The resulting preorder can be seen as a relation of efficiency among processes.

In [7] a probabilistic version of the testing approach is developed which introduces the notion of probability of a system P passing a test Q. If P satisfies Q with probability 1 then it reaches the success in all system runs, and this corresponds to the notion of "must" satisfaction. If, instead, Q is satisfied with probability less than 1 then success is reached only in some cases and this corresponds to the notion of "may" satisfaction. In this context $P \leq R$ is defined by requiring that for all tests, the probability of P satisfying the test is not greater than the probability of R satisfying the test.

In order to develop a theory of testing for GSMPA we integrate the timed and probabilistic approach so that we can compare systems by accounting for both aspects. As in [14] we define a test as a pair (Q, l) composed of a GSMPA process Q and a time bound l. Then as in [7] we consider the probability that a system P satisfies the test (Q, l). This is done as follows. Consider the stochastic process $X = \{X(t) \mid t \in \mathbb{R}^+\}$ representing the probabilistic behavior of $P \|_S Q$ (see Sect. 3.3). For each $t \in \mathbb{R}^+$, $X(t)$ is a stochastic variable denoting the state reached by $P \|_S Q$ at time t. Now we denote with T the stochastic variable representing the time spent by $P \|_S Q$ for reaching the success, where T is ∞ if success is not reached. We have $T = min\{t \mid \text{in X(t) } \omega \text{ starts execution}\}$. The probability of P satisfying the test (Q, l) is then $P(T \leq l)$, hence the probability distribution of T evaluated in l. In this way we obtain a probabilistic quantification of the "may" satisfaction of a timed test (Q, l) and we have a "must" satisfaction when the probability is 1. Finally the efficiency preorder is determined by resorting to the probabilistic definition of preorder given in [7]: $P \leq R$ if for all tests (Q, l), the probability of P satisfying (Q, l) is not greater than the probability of R satisfying (Q, l). Note that in this context the methodology of preorder checking is still valid because in $P \leq R$ the process P represents the minimal requirement to be met both in term of system behavior and in term of system efficiency.

3.2 Probabilistic Real-Time Logic

A second methodology for analyzing systems specified with GSMPA is based on a generalization of the standard approach of real-time of validating a system against a specification expressed by formulas of some temporal logic. For example let us consider the real-time temporal logic TCTL [1], an extension of the temporal logic CTL [6] where temporal operators have subscripts that limit their scope in time. The formulas of TCTL are defined by the following grammar:

$$\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \, \exists\mathcal{U}_{\# l} \, \psi \mid \psi \, \forall\mathcal{U}_{\# l} \, \psi$$

where p ranges over atomic propositions on system states and $\#$ stands for one of the binary relations $<$, $\leq$, $>$, $\geq$ or $=$. The binary temporal operator $\psi_1 \, \exists\mathcal{U}_{\# l} \, \psi_2$ means that for some computation there exist a finite prefix such that ψ_1 holds at all the prefix intermediate states and ψ_2 holds at the final state at a time t satisfying $t \# l$. $\psi_1 \, \forall\mathcal{U}_{\# l} \, \psi_2$ means that for every computation the above property holds.

TCTL only take into consideration the mere possibility of computations, and abstract away from the actual probability with which a possible computation may take place. In order to be able to express a property requiring a certain behavior to be performed with (at least) a certain probability we must extend the expressiveness of the temporal logic (as done in [10] in the pure probabilistic case). We propose a probabilistic version of TCTL with the $\exists\mathcal{U}_{\# l}$ and $\forall\mathcal{U}_{\# l}$ being replaced by operators of the form $\exists\mathcal{U}_{p,\# l}$, where p is a probability. $\exists\mathcal{U}_{p,\# l}$ denotes that a computation composed of a finite prefix such that ψ_1 holds at all the intermediate states and ψ_2 holds at the final state at a time t satisfying $t \# l$ may be performed at least with probability p. We take $\exists\mathcal{U}_{0,\# l}$ as a special case denoting simply the existence of a computation as for the old $\exists\mathcal{U}_{\# l}$. Moreover we have that the old $\psi_1 \, \forall\mathcal{U}_{\# l} \, \psi_2$ can be obtained with $(\psi_1 \, \exists\mathcal{U}_{1,\# l} \, \psi_2) \vee \neg(\psi_1 \, \exists\mathcal{U}_{0,\# l} \, \psi_2)$.

Having introduced probabilities in system models has also a strong implication in the verification of system properties. Whilst in the standard approach of real-time (symbolic) *model checking* techniques must be applied [1], the fact of dealing with probabilistic models gives the opportunity of verifying properties with an approach based on testing. In [10] is shown that, in the context of a pure probabilistic language, for any given (testable) property and (low) probability p, it is possible to determine a test that is successful: with probability $1-p$ if the property holds, with probability p if the property does not hold. In this way by simply running the test and determining if we reach success we have an evaluation of the validity of the property within a certain confidence level, and this can be done for any desired degree of confidence.

3.3 Real-Time Semantics and Performance Semantics

Besides the possibility of performing combined analysis, with our approach it is possible to formally derive from the same model of a concurrent system, a pure real-time model and a pure performance model which are guaranteed to be consistent by construction. Such models can, then, be analyzed by employing existing techniques and tools.

The real-time model is obtained by simply abstracting from the probabilistic quantification of non-deterministic choices and non-deterministic durations. This implies eliminating weights from choice transitions and by turning probability distributions of actions into sets of possible time values, i.e. time values that are given probability (density) greater than 0. As a consequence the information related to the probabilistic-time behavior is lost. The model obtained can be simply turned into a timed automata [1] by associating a clock with each action. Clocks start from 0 and possible durations for an action are expressed as conditions on the associated clock in the transitions representing the action termination.

The performance model, a stochastic process representing the probabilistic behavior of the system, is a GSMP which is formally obtained by abstracting from functional information, i.e. from action types and locations. See [4] for more details. An automatizable method to analyze mathematically a GSMP for deriving performance measures, consists in approximating the general distributions with combinations of exponential distributions. A consequence of this approximation is that all duration values for an activity get probability greater than 0. Therefore the information about time constraints (related to the real-time behavior of the system) is lost.

4 Conclusion

Much work has still to be done in order to fully realize the integration of real-time and probabilistic-time process algebras. In fact GSMPA represents only a core calculus which includes the technical and conceptual machinery that allows to express both aspects of time. First of all an operator for representing the interruption of actions has to be introduced in order to improve the expressiveness of the language. Besides we must investigate whether our notion of testing preorder is a precongruence and we must find a characterization that allows to actually decide relationships between processes. Moreover a technique for the symbolic model checking of properties expressed with the probabilistic extension of TCTL must be developed and the alternative possibility of verification through testing, which sounds very promising, must be investigated.

References

[1] R. Alur, C. Courcoubetis, D. Dill *"Model-Checking in Dense Real-Time"*, in Information and Computation 104:2-34, 1993

[2] L. Aceto, M. Hennessy, *"Adding Action Refinement to a Finite Process Algebra"*, in Information and Computation 115:179-247, 1994

[3] M. Bravetti, M. Bernardo, R. Gorrieri, *"Towards Performance Evaluation with General Distributions in Process Algebras"*, in Proc. of the *9th Int. Conf. on Concurrency Theory (CONCUR '98)*, LNCS 1466:405-422, Nice (France), 1998

[4] M. Bravetti, M. Bernardo, R. Gorrieri, *"Generalized Semi Markovian Process Algebra"*, Technical Report UBLCS-97-9, University of Bologna (Italy), October 1997

[5] M. Bernardo, R. Gorrieri, *"A Tutorial on EMPA: A Theory of Concurrent Processes with Nondeterminism, Priorities, Probabilities and Time"*, in Theoretical Computer Science 202:1-54, 1998

[6] E. M. Clarke, E. A. Emerson, A. P. Sistla, *"Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications"*, in ACM Trans. on Programming Languages and Systems 8, 244-263, 1986

[7] R. Cleaveland, S. A. Smolka, A. E. Zwarico, *"Testing Preorders for Probabilistic Processes"*, in Proc. of the *19th Int. Colloquium on Automata, Languages and Programming (ICALP '92)*, LNCS 623:708-719, Vienna (Austria), 1992

[8] R. De Nicola, M. C. B. Hennessy, *"Testing Equivalences for Processes"*, in Theoretical Computer Science 34:83-133, 1984

[9] R.J. van Glabbeek, F.W. Vaandrager, *"Petri Net Models for Algebraic Theories of Concurrency"*, in Proc. of the *Conf. on Parallel Architectures and Languages Europe (PARLE)*, LNCS 259:224-242, Eindhoven (The Netherlands), 1987

[10] K.G. Larsen, A. Skou, *"Bisimulation through Probabilistic Testing"*, in Information and Computation 94:1-28, 1991

[11] K. Matthes, *"Zur Theorie der Bedienungsprozesse"*, in Trans. of the *3rd Prague Conf. on Information Theory, Stat. Dec. Fns. and Random Processes*, pp. 513-528, 1962

[12] X. Nicollin, J. Sifakis, *"The Algebra of Timed Processes, ATP: Theory and Application"*, in Information and Computation 114:131-178, 1994

[13] M. Nielsen, P.S. Thiagarajan, *"Degrees of Nondeterminism and Concurrency"*, in Proc. of the *4th Conf. on Foundations of Software Technologies and Theoretical Computer Science*, LNCS 181:89-117, 1984

[14] W. Vogler, *"Timed Testing of Concurrent Systems"*, in Information and Computation 121:149-171, 1995