12

An Algebraic Approach to the Specification of Stochastic Systems (Extended Abstract)

P. R. D'Argenio[1]*, *J.-P. Katoen*[2], *and E. Brinksma*[1]

[1]*Dept. of Computer Science. University of Twente.*
P.O.Box 217. 7500 AE Enschede. The Netherlands.
{dargenio,brinksma}@cs.utwente.nl

[2]*Lehrstuhl für Informatik VII. University of Erlangen-Nürnberg.*
Martensstrasse 3. D-91058 Erlangen. Germany.
katoen@informatik.uni-erlangen.de

Abstract

We introduce a framework to study stochastic systems, i.e. systems in which the time of occurrence of activities is a general random variable. We introduce and discuss in depth a stochastic process algebra (named ♤) adequate to specify and analyse those systems. In order to give semantics to ♤, we also introduce a model that is an extension of traditional automata with clocks which are basically random variables: the stochastic automata model. We show that this model and ♤ are equally expressive. Although stochastic automata are adequate to analyse systems since they are finite objects, they are still too coarse to serve as concrete semantic objects. Therefore, we introduce a type of probabilistic transition system that can deal with arbitrary probability spaces. In addition, we give a finite axiomatisation for ♤ that is sound for the several semantic notions we deal with, and complete for the finest of them. Moreover, an expansion law is straightforwardly derived.

Keywords

Stochastic process algebras, stochastic automata, probabilistic transition systems, probabilistic bisimulations, real-time systems.

1 INTRODUCTION

In the world of performance modelling, many models have been defined to analyse and simulate systems such as queuing networks, stochastic Petri-nets, or generalised semi-Markov processes. It has been argued many times that,

*Supported by the NWO/SION project 612-33-006.

with these models, the difficulty of the design and analysis of a system grows rapidly with the size and complexity of the system itself.

In the last few years, this phenomenon has drawn the attention of many researchers into extending process algebras with stochastic and real-time features [16, 11, 13, 4, 6, 5, ...]. The so called *stochastic process algebras* considerably simplify the tractability of complex systems because, in this framework, systems do not need to be modelled as a whole, but as a composition of small subsystems. Another advantage is that stochastic process algebras not only allow to study the performance of a system, but also its functionality.

In this article, we have a three-folded purpose: we discuss a probabilistic transition system model based on general distributions, we introduce a stochastic automata model which borrows ideas from both timed automata [2, 14] and generalised semi-Markov processes (GSMP, for short) [26, 10], and finally we introduce and discuss in depth a stochastic process algebra.

Probabilistic transition systems (PTS, for short) have been widely studied in the context of discrete probabilities [25, 17, 12, 19, 23, 9, ...]. However, the case with general distributions has received scant attention [13, 22]. In the first part of our paper we define probabilistic transition systems that deal with any kind of probabilistic spaces, including thus discrete, continuous, and singular. This generality allows the specification of real-time systems in which time constraints are not necessarily deterministic but dependent on random variables. Our definition is basically a generalisation and formalisation of [13].

Although PTSs are an adequate framework for the understanding of processes with stochastic behaviour, they are highly infinite which makes them too difficult to deal with. Therefore, we also introduce the so-called *stochastic automata*. A stochastic automaton is an automaton extended with clocks. Clocks are variables which take some random value which is set according to a given probability distribution. Once set, clocks count down, and when they reach value zero, they may enable certain transitions in the automaton. We define the semantics of stochastic automata in terms of PTSs. In fact, we define two different kinds of semantics: one when the stochastic automaton is regarded as a *closed* system, i.e., when the system is complete by itself and no external interaction is required, and the other when it is regarded as an *open* system, that is, a system that cooperates with the environment or is intended to be part of a larger system. Interpretation of stochastic automata as closed systems is adequate for the final analysis of the system, e.g. to study the performance or to verify the whole system. Instead, the interpretation as open systems is appropriate to study compositionality and to analyse how systems behave in contexts.

Compositionality is a major drawback in many existing models for performance analysis such as queuing networks, stochastic Petri nets, or GSMPs, specially, in non-Markovian models. On the contrary, stochastic automata offer an appropriate framework to straightforwardly compose systems. In fact, because of its simplicity, we use stochastic automata as the underlying semantics of a stochastic process algebra that allows to express general distributions. Actually, the stochastic automata model and the process algebra turn out to be equally expressive. In this way, the process algebra can be regarded as a language to describe stochastic automata. This result closely follows the methodology of [7] where a process calculus for timed automata was introduced. Since a stochastic automaton can be executed using discrete event simulation techniques, the process algebra is called SPADES standing for *stochastic process algebra for discrete event simulation*, but we just write ♤.

Usually, the semantics of stochastic process algebras such as TIPP [11, 15], PEPA [16], and EMPA [4], is defined in terms of extended transition systems, which basically associate a distribution function to each transition. However, the inherent interleaving characteristic of transition systems demands a careful treatment of the definition of parallel composition. In traditional interleaving process algebras like CCS [18] the expansion law plays an important role: it says how parallel composition can be decomposed in terms of more primitive operations, namely, prefixing and non-deterministic choice. Stochastic process algebras extend prefixing into $a_F; P$ where F is a distribution function which determines the probability of the random delay after which the action a can happen. In this setting, the expansion law does no longer hold in general. To face this problem, the community has come up with different solutions.

A first proposal, and the most widely accepted, has been to restrict the attention to exponential distributions. Their memoryless property restores the expansion law [16, 15, 4]. Others have faced the general case [11, 13, 20] but the underlying semantic object usually becomes cumbersome and infinite, which makes it intractable. An alternative solution is to drop the expansion law by moving to true concurrency models [6], but for simple recursive processes, their semantic representations are infinite.

We propose a more elegant solution for ♤. We separate the stochastic information from the action name. (We remark that a similar approach has been used in [13].) Instead of writing $a_F; P$, we write $\{\!|x_F|\!\}(\{x_F\}\mapsto a; P)$. The operator $\{\!|x_F|\!\}\ldots$ sets the clock x_F according to the distribution function F, and the operation $\{x_F\}\mapsto\ldots$ prevents the prefixing $a; P$ to happen until clock x_F has expired (i.e., reached value 0). This separation of concerns gives as a result a straightforward expansion law, and moreover, it introduces more expressive power. We observe that in principle *any* kind of (continuous, discrete, ...) distribution function is allowed in this model, while we maintain a finite semantic object in a reasonable way (comparable to regular processes in CCS).

The paper is organised as follows. Section 2 discusses probabilistic transition systems and probabilistic bisimilarity for general probability spaces. In Section 3, we define the stochastic automata model and study its semantics. In Section 4, we discuss ♤ in depth including its semantics and axiomatisation. We discuss related work and further research in Section 5.

The complete report of this article, including proofs, rigorous definitions, and detailed technicalities, is given in [8].

2 PROBABILISTIC TRANSITION SYSTEMS

In this section, we introduce the notion of probabilistic transition systems and probabilistic bisimulation.

Preliminaries. Let $\mathbb{N}$ be the set of non-negative integers. Let $\mathbb{R}$ be the set of real numbers and $\mathbb{R}_{\geq 0}$ the set of non-negative reals. For $n \in \mathbb{N}$, let $\mathbb{R}^n$ denote the nth Cartesian product of $\mathbb{R}$. In particular, $\mathbb{R}^0 \stackrel{\text{def}}{=} \{\emptyset\}$.

A *probability space* is a structure $(\Omega, \mathcal{F}, P)$ where Ω is a *sample space*, $\mathcal{F}$ is a *σ-algebra* on Ω, and P is a *probability measure* on $\mathcal{F}$. In this work, we consider only probability spaces isomorphic to some Borel space defined in a real hyperspace, whose coordinates come from independent random variables. We denote by $\mathcal{R}(F_1, \ldots, F_n)$ the probability space $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n), P_n)$ where $\mathcal{B}(\mathbb{R}^n)$ is the Borel algebra on $\mathbb{R}^n$ and P_n is the unique probability measure obtained from $F_1, \ldots, F_n$, a given family of distribution functions. In particular, if $n = 0$, $\mathcal{R}()$ is the trivial probability space $(\{\emptyset\}, \{\emptyset, \{\emptyset\}\}, P_0)$ with P_0 in the obvious way. We refer to [24] for further reading.

Let $\mathcal{P} = (\Omega, \mathcal{F}, P)$ be a probability space. Let $\mathcal{D} : \Omega \to \Omega'$ be an injective function. We lift $\mathcal{D}$ to subsets of Ω as usual: $\mathcal{D}(A) \stackrel{\text{def}}{=} \{\mathcal{D}(a) \mid a \in A\}$ and define $\mathcal{F}' \stackrel{\text{def}}{=} \{\mathcal{D}(A) \mid A \in \mathcal{F}\}$. Now, it is clear that, $\mathcal{D}(\mathcal{P}) \stackrel{\text{def}}{=} (\mathcal{D}(\Omega), \mathcal{F}', P \circ \mathcal{D}^{-1})$ is also a probability space. Since $\mathcal{D}(\mathcal{P})$ is basically the same probability space as $\mathcal{P}$, we say that $\mathcal{D}$ is a *decoration* and we refer to $\mathcal{D}(\mathcal{P})$ as the *decoration of $\mathcal{P}$ according to $\mathcal{D}$*. Decoration functions are a key concept in the probabilistic part of the stochastic automata semantics.

Probabilistic transition systems. We introduce a transition system with probabilistic information. We allow any kind of probability spaces, including continuous distributions. The definition of our model is inspired by [12] and [13], although we do not consider explicit timed transitions.

Definition 1 Let *Prob*(H) denote the set of probability spaces $(\Omega, \mathcal{F}, P)$ such that $\Omega \subseteq H$. A *probabilistic transition system* (PTS for short) is a structure $\mathcal{T} = (\Sigma, \Sigma', \sigma_0, \mathcal{L}, T, \longrightarrow)$ where

1. Σ and Σ' are two disjoint sets of *states*, with the *initial state* $\sigma_0 \in \Sigma$. States in Σ are called *probabilistic* and states in Σ' are *non-deterministic*.
2. $\mathcal{L}$ is a set of *labels*.
3. $T : \Sigma \to \mathit{Prob}(\Sigma')$ is the *probabilistic transition relation*.
4. $\longrightarrow \subseteq \Sigma' \times \mathcal{L} \times \Sigma$ is the *labelled (or non-deterministic) transition relation*. We denote $\sigma' \xrightarrow{\ell} \sigma$ for $\langle \sigma', \ell, \sigma \rangle \in \longrightarrow$, and $\sigma' \not\xrightarrow{\ell}$ for $\neg \exists \sigma . \sigma' \xrightarrow{\ell} \sigma$. □

Since T is defined as a (total) function, each probabilistic state has exactly one outgoing transition. It can be shown that if *Prob*(Σ') contains only discrete probability spaces, PTSs are as expressive as the simple probabilistic automata of [23] and strictly more expressive than the class of reactive PTSs [17, 9].

Since our interest is to deal with time information using PTSs, the set of labels we will use is $\mathcal{L} = \mathbf{A} \times \mathbb{R}_{\geq 0}$, where $\mathbf{A}$ is a set of action names and $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers, which are intended to denote the (relative) time at which an action takes place. We usually denote $a(d)$ instead of (a, d) whenever $(a, d) \in \mathcal{L}$ and it means "action a occurs right after the system has been idle for d time units".

Probabilistic bisimulation. Probabilistic bisimulation was introduced in [17] for a class of PTSs dealing only with discrete probability spaces. This definition has been adapted in [9, 12, 23] for several variants of PTSs, all of them in a discrete probabilistic setting. Bisimulations have also been defined in settings where exponential distributions are involved [16, 15, 4]. [13] has defined bisimulation in a continuous setting and [22] used a coalgebraic approach for the general setting. In essence, our definition coincides with the one in [13].

Definition 2 Let $(\Sigma, \Sigma', \sigma_0, \mathcal{L}, T, \longrightarrow)$ be a PTS. We define the function $\mu : \Sigma \times \wp(\Sigma') \to [0,1]$ by $\mu(\sigma, S) \overset{\text{def}}{=}$ **if** $S \cap \Omega \in \mathcal{F}$ **then** $P(S \cap \Omega)$ **else** 0, provided that $T(\sigma) = (\Omega, \mathcal{F}, P)$.

Let R be an equivalence relation on $\Sigma \cup \Sigma'$ such that if $\sigma_1 R \sigma_2$ then either $\sigma_1, \sigma_2 \in \Sigma$ or $\sigma_1, \sigma_2 \in \Sigma'$. Let Σ'/R be the set of equivalence classes in Σ' induced by R. Then R is a *(probabilistic) bisimulation* if, whenever $\sigma_1 R \sigma_2$, for all $S \subseteq \Sigma'/R$ and $\ell \in \mathcal{L}$, the following transfer properties hold

1. $\mu(\sigma_1, \bigcup S) = \mu(\sigma_2, \bigcup S)$, if $\sigma_1, \sigma_2 \in \Sigma$; and
2. $\sigma_1 \xrightarrow{\ell} \sigma_1'$ implies $\sigma_2 \xrightarrow{\ell} \sigma_2'$ and $\sigma_1' R \sigma_2'$, for some $\sigma_2' \in \Sigma$, if $\sigma_1, \sigma_2 \in \Sigma'$.

Two states σ_1 and σ_2 are *(probabilistically) bisimilar*, notation $\sigma_1 \underline{\leftrightarrow} \sigma_2$, if there exists a probabilistic bisimulation R with $\sigma_1 R \sigma_2$. Two PTSs $\mathcal{T}_1$ and $\mathcal{T}_2$ are *bisimilar*, notation $\mathcal{T}_1 \underline{\leftrightarrow} \mathcal{T}_2$, if their respective initial states are bisimilar on the disjoint union of $\mathcal{T}_1$ and $\mathcal{T}_2$. □

It can be proven that $\underline{\leftrightarrow}$ is the largest probabilistic bisimulation, and hence, that it is an equivalence relation.

Although, the definition of probabilistic bisimulation coincides with the traditional definitions in the discrete case, e.g. [17, 12, 23], we remark a necessary difference. In the discrete case, instead of property 1. above, it suffices to insist that $\mu(\sigma_1, S) = \mu(\sigma_2, S)$ where $S \in \Sigma'/R$, i.e., S is an equivalence class instead of a set of equivalence classes. In our case, this would have been too weak due to the allowance of, for instance, continuous distribution function. For example, consider the PTSs $\mathcal{T}_i = (\{\sigma\}, \mathbb{R}, \sigma, \mathbb{R}, T_i, \longrightarrow)$, $i \in \{1, 2\}$, where $d \xrightarrow{d} \sigma$, and $T_1(\sigma)$ and $T_2(\sigma)$ are the probability spaces for a uniform distribution on $[0,1]$ and $[1,2]$, respectively. According to Definition 2, $\mathcal{T}_1$ and $\mathcal{T}_2$ are not bisimilar, since they do not agree in their probabilities. However, the weaker property of the discrete case would have induced that the identity relation is a probabilistic bisimulation since the probability of a point in a continuous probability space is always zero.

3 THE STOCHASTIC AUTOMATON MODEL

In this section, we introduce a new automaton model that allows us to represent processes with stochastic information. The basic idea is borrowed from timed automata [2, 14] by combining it with ideas of discrete event systems, in particular GSMPs [10, 26]. Besides, we study two different semantic models for stochastic automata.

Stochastic Automata. We first enumerate all the ingredients of a stochastic automaton and then give an example to explain the intuition behind the definition.

Definition 3 A *stochastic automaton* is a structure $(\mathcal{S}, s_0, \mathcal{C}, \mathbf{A}, \longrightarrow, \kappa, F)$ where:

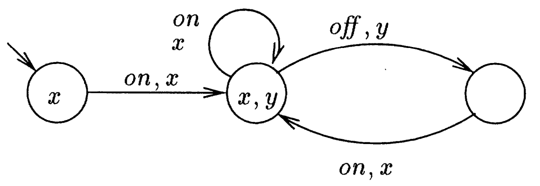
- $\mathcal{S}$ is a set of *locations* with $s_0 \in \mathcal{S}$ being the *initial location.*
- $\mathcal{C}$ is a set of *clocks.*
- $\mathbf{A}$ is a set of *actions*
- $\longrightarrow \;\subseteq \mathcal{S} \times (\mathbf{A} \times \wp_{\mathrm{fin}}(\mathcal{C})) \times \mathcal{S}$ is the set of *edges*. We denote the edge $(s, a, C, s') \in\longrightarrow$ by $s \xrightarrow{a,C} s'$ and we say that C is its *trigger set.*
- $\kappa : \mathcal{S} \to \wp_{\mathrm{fin}}(\mathcal{C})$ is the *clock setting function.*
- $F : \mathcal{C} \to (\mathbb{R} \to [0, 1])$ assigns to each clock a *distribution function* such that $F(x)(t) = 0$ for $t < 0$; we write F_x instead of $F(x)$.

Notice that each clock $x \in \mathcal{C}$ is a random variable with distribution F_x. □

As in [7], the information of which clock should be set is related to the locations. Clocks are randomly set according to a certain associated distribution function and they count down. A clock expires if it has reached the value 0. The occurrence of an action is controlled by the expiration of clocks. Thus, whenever $s \xrightarrow{a,C} s'$ and the system is in location s, a happens as soon as all the clocks in the trigger set C have expired. Immediately afterwards all clocks in $\kappa(s')$ are randomly set according to their respective distributions.

Example 1 Figure 1 represents a switch that controls a light. In the picture, circles represent locations, variables enumerated in each location are the clocks that are to be set according to the function κ, and edges are represented by the arrows. The initial location is represented by a small ingoing arrow. The distribution function of each clock is given beside the picture.

The switch may be turned on at any time according to an exponential distribution with average of 30 minutes, even if the light is still on. It switches automatically off exactly 2 minutes after the most recent time the light was switched on. Since we considered that exactly 2 minutes must pass before the light is turned off, y is a random variable that takes value 2 with probability 1. Notice that we can easily change the system to consider that clock y is not



$$F_x(t) = 1 - e^{-\frac{1}{30}t}$$

$$F_y(t) = \begin{cases} 0 & \text{if } t < 2 \\ 1 & \text{if } t \geq 2 \end{cases}$$

Figure 1 The switch

precise and has a drift of ϵ units of time. If, for instance, we assume that such a drift is uniformly distributed, then y would become a random variable with a uniform distribution in $[2-\epsilon, 2+\epsilon]$. □

Actual behaviour. In this subsection, we define the semantics of stochastic automata when it is regarded as a closed system. A *closed system* is a system which is considered complete by itself and no external interaction is needed. In this kind of system one not only models the components of the intended system but also the environment with which it interacts. In this way, the activity of the whole system can take place as soon as it becomes ready to be executed since there is no external agent that may delay its execution. That is, closed systems respond to the *maximal progress* property. We refer to this interpretation as the *actual behaviour*.

First, we introduce some background concepts, then we state which are the probabilistic spaces that we use, and finally we define the actual behaviour of stochastic automata.

A *valuation* is a function $v : \mathcal{C} \to \mathbb{R}$. Let $\mathcal{V}$ be the set of all valuations. If $d \in \mathbb{R}_{\geq 0}$, we define $v - d$ by $\forall x \in \mathcal{C}.\ (v-d)(x) \overset{\text{def}}{=} v(x) - d$. For simplicity, assume the set $\mathcal{C}$ of clock is totally ordered. Thus, if $C \subseteq \mathcal{C}$, we write $\overrightarrow{C}$ for the ordered form of C and $\overrightarrow{C}(i)$ for its i-th element. Let $C \subseteq \mathcal{C}$, $n = \#C$, and $\overrightarrow{D} \in \mathbb{R}^n$. We define $v[\overrightarrow{C} \hookleftarrow \overrightarrow{D}]$ by

$$v[\overrightarrow{C} \hookleftarrow \overrightarrow{D}](y) \overset{\text{def}}{=} \begin{cases} \overrightarrow{D}(i) & \text{if } y = \overrightarrow{C}(i), \text{ for some } i \in \{1, \ldots, n\} \\ v(y) & \text{otherwise} \end{cases}$$

Let $(\mathcal{S}, s_0, \mathcal{C}, \mathbf{A}, \longrightarrow\!\!\!\!\blacktriangleright, \kappa, F)$ be a stochastic automaton. Let s be a location in $\mathcal{S}$ and let $n = \#\kappa(s)$. Let v be a valuation in $\mathcal{V}$. Define $\mathcal{D}^s_v : \mathbb{R}^n \to \{s\} \times \mathcal{V} \times \{1\}$ by $\mathcal{D}^s_v(\overrightarrow{D}) \overset{\text{def}}{=} (s, v[\overrightarrow{\kappa(s)} \hookleftarrow \overrightarrow{D}], 1)$. Notice that $\mathcal{D}^s_v$ is injective. In the next definition we will use the probability space $\mathcal{R}(F_1, \ldots, F_n)$ decorated according to some $\mathcal{D}^s_v$.

Definition 4 Let $SA = (\mathcal{S}, s_0, \mathcal{C}, \mathbf{A}, \longrightarrow\!\!\!\!\blacktriangleright, \kappa, F)$ be a stochastic automaton. The *interpretation* (or the *actual behaviour*) of SA in a valuation v_0 is given by the PTS $I^A_{v_0}(SA) \overset{\text{def}}{=} ((\mathcal{S} \times \mathcal{V} \times \{0\}), (\mathcal{S} \times \mathcal{V} \times \{1\}), (s_0, v_0, 0), \mathbf{A} \times \mathbb{R}_{\geq 0}, T, \longrightarrow)$ with T and $\longrightarrow$ defined as follows

$$\textbf{Prob}\ \frac{\overrightarrow{\kappa(s)} = (x_1, \ldots, x_n)}{T(s, v, 0) \ = \ \mathcal{D}^s_v(\mathcal{R}(F_{x_1}, \ldots, F_{x_n}))}$$

$$\textbf{Act}\ \frac{s \xrightarrow{a,C} s' \qquad d \in \mathbb{R}_{\geq 0} \qquad \forall x \in C.\,(v-d)(x) \leq 0 \qquad \forall d' \in [0,d).\ \forall s \xrightarrow{b,C'}.\ \exists y \in C'.\ (v-d')(y) > 0}{(s,v,1) \xrightarrow{a(d)} (s',(v-d),0)}$$

We say that an edge $s \xrightarrow{a,C} s'$ is enabled in a valuation v if it induces a non-deterministic transition outgoing from $(s,v,1)$. In particular, notice that $s \xrightarrow{a,\emptyset} s'$ is enabled for any valuation v. □

Notice that, according to Definition 4, for each location s and valuation v there is exactly one probabilistic transition since $\mathcal{D}^s_v$ is injective. So, for any stochastic automaton SA and any valuation v_0, $I^A_{v_0}(SA)$ is indeed a PTS.

Rule **Prob** considers the setting of the clocks. Since the values of the clocks are assigned randomly, a probabilistic transition corresponds to this step. Notice that this definition relies on the definition of $\mathcal{D}^s_v$ on probability spaces. Rule **Act** explains the case of triggering an edge. So, for the occurrence of an action a at time d according to an edge $s \xrightarrow{a,C} s'$, we check that all the clocks in the trigger set C have already expired at time d. This part is considered by the satisfaction of the predicate $\forall x \in C.\ (v-d)(x) \leq 0$. Moreover, it should be the case that no edge was enabled before. That is, any edge must have an active (i.e. positive) clock at any valuation "previous" to $v-d$. In this way, the edge is forced to occur as soon as it becomes enabled. So, the maximal progress is checked by the formula $\forall d' \in [0,d).\ \forall s \xrightarrow{b,C'}.\ \exists y \in C'.\ (v-d')(y) > 0$. For the reader familiar with timed automata [2, 14], we may say that the first constraint corresponds to the guard of the edge $s \xrightarrow{a,C} s'$, and the second constraint is the invariant of location s.

Example 2. To understand the formal semantics, we consider a simple example. Figure 2 represents an alarm bell that rings randomly between 10 and 11 seconds according to a uniform distribution. We define clock x to be a random variable with a uniform distribution function F_x in the interval $[10, 11]$. If s is the only location of the alarm bell, its PTS is given by

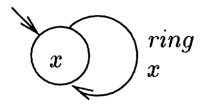


Figure 2 The alarm bell

$$\Sigma = \{(s, x{\leftarrow\!\dashv}d, 0) \mid d \in \mathbb{R}\} \qquad \Sigma' = \{(s, x{\leftarrow\!\dashv}d, 1) \mid d \in \mathbb{R}\}$$

$$T(s,v,0) = \mathcal{D}^s_v(\mathcal{R}(F_x)) \qquad (s, x:=d, 1) \xrightarrow{ring(d)} (s, x:=d, 0) \ \ (\text{if } d \geq 0) \quad \square$$

We can extend the definition of probabilistic bisimulation to stochastic automata as follows.

Definition 5 Two stochastic automata SA_1 and SA_2 are *(probabilistically) bisimilar*, notation $SA_1 \leftrightarrow SA_2$, if, for every valuation v, their interpretations are bisimilar, i.e., $I^A_v(SA_1) \leftrightarrow I^A_v(SA_2)$. □

Potential behaviour. In this subsection, we define the behaviour of a stochastic automaton as an open system. An *open system* is a system that interacts with its environment. The environment can be a user or another system. Basically, an open system is a component of a larger system. When a stochastic automaton describes an open system, the semantics given in Definition 4 does not suffice. In an open system, an action that is enabled may not be executed until the environment is also ready to execute such an action. Therefore, an activity may not take place as soon as it is enabled. This kind of behaviour is appropriate to study compositionality. In fact, it turns out that probabilistic bisimilarity is not a congruence for some basic operations on stochastic automata, such as parallel composition. This has to do with the race condition on the branches of the stochastic automata. Fastest branches (i.e. branches which are enabled) may be disallowed or slowed down when the system is embedded in some context, and therefore, slower branches, which could not be executed in isolation, may become enabled in the composed stochastic automata. For a discussion of this phenomenon, we refer to Example 4.

Therefore, we need to consider not only the actual behaviour of a stochastic automaton, but also its *potential behaviour*. The potential behaviour is in principle the actual behaviour with a larger non-deterministic transition relation. A non-deterministic transition in the potential behaviour represents the fact that an edge is potentially executable at any time after it becomes enabled.

Definition 6 Let $SA = (\mathcal{S}, s_0, \mathcal{C}, \mathbf{A}, \longrightarrow, \kappa, F)$ be a stochastic automaton. The *potential behaviour* of SA in a valuation v_0 is defined by the PTS $I^P_{v_0}(SA) \stackrel{\text{def}}{=} ((\mathcal{S} \times \mathcal{V} \times \{0\}), (\mathcal{S} \times \mathcal{V} \times \{1\}), (s_0, v_0, 0), \mathbf{A} \times \mathbb{R}_{\geq 0}, T, \longmapsto)$, where T is defined by rule **Prob** as in Definition 4 and $\longmapsto$ is defined as follows

$$\textbf{Pot}\ \frac{s \xrightarrow{a,C} s' \qquad d \in \mathbb{R}_{\geq 0} \qquad \forall x \in C.\ (v-d)(x) \leq 0}{(s, v, 1) \overset{a(d)}{\longmapsto} (s', (v-d), 0)} \qquad \square$$

The difference between the actual and the potential behaviour relies on rules **Act** and **Pot**. To be precise, **Pot** is the same as rule **Act** where the constraint of maximal progress has been omitted.

Definition 7 Two stochastic automata SA_1 and SA_2 are *potentially bisimilar*, notation $SA_1 \leftrightarrow_P SA_2$, if, for every valuation v, their potential behaviours are probabilistically bisimilar, i.e., $I^P_v(SA_1) \leftrightarrow I^P_v(SA_2)$. $\square$

The following theorem states that it is always possible to recover the actual behaviour from the potential behaviour.

Theorem 8 *Let SA be a stochastic automaton and let $I^P_{v_0}(SA)$ and $I^A_{v_0}(SA)$ be its potential and actual behaviour in $v_0 \in \mathcal{V}$, respectively. The two following statements are equivalent*

1. $(s,v,1) \overset{a(d)}{\longmapsto} (s',v',0)$ *and for all* $d' \in [0,d)$, $b \in \mathbf{A}$, $(s,v,1) \overset{b(d')}{\not\longmapsto}$
2. $(s,v,1) \xrightarrow{a(d)} (s',v',0)$

As a consequence, we have that potential bisimulation is strictly finer than probabilistic bisimulation. That is, for two stochastic automata SA_1 *and* SA_2, $SA_1 \underline{\leftrightarrow}_P SA_2$ *implies* $SA_1 \underline{\leftrightarrow} SA_2$.

Structural bisimulation. Often, we can check if two stochastic automata are equivalent just by inspecting their structure, without the need to study their actual or potential behaviour. Thus, we define a stronger notion of equivalence which we call structural bisimulation. We also state that this relation is finer than potential bisimulation.

Definition 9 Let $(\mathcal{S}, s_0, \mathcal{C}, \mathbf{A}, \longrightarrow, \kappa, F)$ be a stochastic automaton. A relation $R \subseteq \mathcal{S} \times \mathcal{S}$ is a *structural bisimulation* if R is symmetric and whenever $s_1 R s_2$, for all $a \in \mathbf{A}$, $C \subseteq \mathcal{C}$, the following transfer properties hold:

1. $s_1 \xrightarrow{a,C} s_1'$ implies $\exists s_2'.\ s_2 \xrightarrow{a,C} s_2'$ and $s_1' R s_2'$;
2. $\kappa(s_1) = \kappa(s_2)$

If R is a structural bisimulation such that $s_1 R s_2$, we denote $s_1 \underset{\sim}{\leftrightarrow} s_2$ and we say that s_1 and s_2 are structurally bisimilar. Two stochastic automata SA_1 and SA_2 are *structurally bisimilar*, notation $SA_1 \underset{\sim}{\leftrightarrow} SA_2$, if their respective initial locations are structurally bisimilar on the disjoint union of SA_1 and SA_2. □

Following standard results on bisimulation, we can prove that $\underset{\sim}{\leftrightarrow}$ is the largest structural bisimulation, and moreover, that it is an equivalence relation.

It is clear that two stochastic automata may be potentially bisimilar but not structurally bisimilar. Instead, structural bisimulation implies potential bisimulation, and hence probabilistic bisimulation, too.

Theorem 10 *Let* SA_1 *and* SA_2 *be two stochastic automata. If* $SA_1 \underset{\sim}{\leftrightarrow} SA_2$ *then* $SA_1 \underline{\leftrightarrow}_P SA_2$.

4 SPADES

In the following we introduce SPADES, denoted by ♤ and standing for *stochastic process algebra for discrete event simulation*. The methodology that we follow to define the syntax and the semantics is close to results in [7] where a process algebra for timed automata was introduced.

Syntax. Let $\mathbf{A}$ be a set of *actions*. Let $\mathcal{CN}$ be a set of clock names and $\mathcal{DF}$ a set of distribution functions. We define $\mathcal{C} \subseteq \mathcal{CN} \times \mathcal{DF}$ to be the set of clocks. We denote x_G for $(x,G) \in \mathcal{C}$. We define the distribution assignment function $F : \mathcal{C} \to (\mathbb{R} \to [0,1])$ by the second projection, i.e., $F(x_G) \overset{\text{def}}{=} G$.

Definition 11 Let $\mathbf{V}$ be a set of *process variables*. The syntax of $\spadesuit$ is defined according to the following grammar:

$$\begin{array}{lll} p & ::= & \mathbf{stop} \mid a;p \mid C \mapsto p \mid p+p \mid \{\!|C|\!\}p \mid \\ & & p\,||_A\,p \mid p\,\lfloor\!\lfloor_A\,p \mid p\,|_A\,p \mid p[f] \mid X \end{array}$$

where $C \subseteq \mathcal{C}$ is finite, $a \in \mathbf{A}$, $A \subseteq \mathbf{A}$, $f : \mathbf{A} \to \mathbf{A}$, and $X \in \mathbf{V}$. A *recursive specification* E is a set of *recursive equations* having the form $X = p$ for each $X \in \mathbf{V}$, where $p \in \spadesuit$. Every recursive specification has a distinguished process variable called *root*. □

Process **stop** represents inaction; it is the process that cannot perform any action. The intended meaning of $a;p$ (named *(action-)prefixing*) is that action a is immediately enabled and once it is performed the behaviour of p is exhibit. $C \mapsto p$ is the *triggering condition*; process p becomes enabled as soon as all the clocks in C expire. $p + q$ is the *choice*; it behaves either as p or q, but not both. We remark that the passage of time does not resolve the choice if the process is regarded as an open system; if instead it is regarded as a closed system, the fastest process is the one to be executed. This last case is known as the race condition. The *clock setting operation* $\{\!|C|\!\}p$ sets the clocks in C according to their respective distribution function. We choose a LOTOS-like parallel composition. Thus, $p||_A q$ executes p and q in parallel, and they are synchronised by actions in A. We should remark that synchronisation may happen if both processes are ready to do it. We also introduce the operators $\lfloor\!\lfloor_A$ and $|_A$ (named *left* and *communication merge* respectively) in order to finitely axiomatise the parallel composition. Finally, the *renaming* operation $p[f]$ is a process that behaves like p except that actions are renamed by f.

Example 3 As a simple example, we give the specification of the switch described in Example 1.

$$\begin{array}{lcl} \mathit{Arrival} & = & \{\!|x_G|\!\}\{x_G\} \mapsto \mathit{on}; \mathit{Arrival} \\ \mathit{Switch}_{\mathit{off}} & = & \mathit{on}; \mathit{Switch}_{\mathit{on}} \\ \mathit{Switch}_{\mathit{on}} & = & \mathit{on}; \mathit{Switch}_{\mathit{on}} + \{\!|y_K|\!\}\{y_K\} \mapsto \mathit{off}; \mathit{Switch}_{\mathit{off}} \\ \mathit{System} & = & \mathit{Arrival} \,||_{\{\mathit{on}\}}\, \mathit{Switch}_{\mathit{off}} \end{array}$$

In this case G is an exponential distribution with rate $\frac{1}{30}$ and K is the distribution function that gives probability 1 to the value 2. Process *Arrival* models the arrival of people which occurs exponentially distributed with average of 30 minutes. *Switch* models the switch itself which initially is *off*. Notice that the switch is always enabled to accept an "*on*" and hence no clock controls this activity on the switch part of the system. Process *System* describes the whole system, allowing people to turn on the switch, i.e., process *Arrival* and *Switch* should synchronise on the action *on*. □

In the sequel, we need the notion of free and bound clock variables. Let

Table 1 Stochastic automata for $\spadesuit$ $(X = p \in E)$

$\kappa(\mathbf{stop}) = \emptyset$	$\kappa(\{\!\|C\|\!\}p) = C \cup \kappa(p)$	$\kappa(p+q) = \kappa(p) \cup \kappa(q)$
$\kappa(a;p) = \emptyset$	$\kappa(C \mapsto p) = \kappa(p)$	$\kappa(p\|\|_A q) = \kappa(p) \cup \kappa(q)$
	$\kappa(p[f]) = \kappa(p)$	$\kappa(p \lfloor\!\lfloor_A q) = \kappa(p) \cup \kappa(q)$
$\kappa(\overline{\mathsf{ck}}(p)) = \emptyset$	$\kappa(X) = \kappa(p)$	$\kappa(p\|_A q) = \kappa(p) \cup \kappa(q)$
$a;p \xrightarrow{a,\emptyset} p$	$\dfrac{p \xrightarrow{a,C} p'}{\begin{array}{c} p+q \xrightarrow{a,C} p' \\ q+p \xrightarrow{a,C} p' \end{array}}$	$\dfrac{p \xrightarrow{a,C} p'}{\begin{array}{c} p\|\|_A q \xrightarrow{a,C} p'\|\|_A \overline{\mathsf{ck}}(q) \\ q\|\|_A p \xrightarrow{a,C} \overline{\mathsf{ck}}(q)\|\|_A p' \\ p\lfloor\!\lfloor_A q \xrightarrow{a,C} p'\|\|_A \overline{\mathsf{ck}}(q) \end{array}}\ a \notin A$
$\dfrac{p \xrightarrow{a,C'} p'}{\{\!\|C\|\!\}p \xrightarrow{a,C'} p'}$		
$\dfrac{p \xrightarrow{a,C'} p'}{C \mapsto p \xrightarrow{a,C\cup C'} p'}$	$\dfrac{p \xrightarrow{a,C} p'}{p[f] \xrightarrow{f(a),C} p'[f]}$	$\dfrac{p \xrightarrow{a,C} p' \quad q \xrightarrow{a,C'} q'}{\begin{array}{c} p\|\|_A q \xrightarrow{a,C\cup C'} p'\|\|_A q' \\ p\|_A q \xrightarrow{a,C\cup C'} p'\|\|_A q' \end{array}}\ a \in A$
$\dfrac{p \xrightarrow{a,C} p'}{X \xrightarrow{a,C} p'}$	$\dfrac{p \xrightarrow{a,C} p'}{\overline{\mathsf{ck}}(p) \xrightarrow{a,C} p'}$	

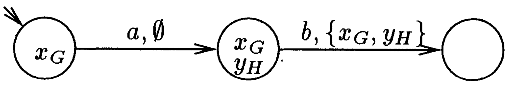
$p \in \spadesuit$. A clock x is free in p if p has a subterm $C \mapsto q$ with $x \in C$ that does not appear in a context $\{\!|C'|\!\}\ldots$ with $x \in C'$. A clock x is bound in p if p has a subterm $\{\!|C|\!\}q$ such that $x \in C$. We denote by $fv(p)$ and $bv(p)$ the sets of free and bound clock variables respectively.

Semantics. As we already said, compositionality is a major drawback in many models for performance analysis, specially in those with the generality of stochastic automata. Instead, stochastic automata can be composed straightforwardly. In fact, we use stochastic automata to give semantics to $\spadesuit$ in a structured operational (i.e., SOS) manner. In order to define the automaton associated to a parallel composition, we need to consider the additional operation $\overline{\mathsf{ck}}$. $\overline{\mathsf{ck}}(p)$ is a process that behaves like p except that no clock is set at the very beginning. We denote this extended language by $\spadesuit^{\mathsf{ck}}$. The sets of free and bounded variables for $\overline{\mathsf{ck}}(p)$ are defined by $fv(\overline{\mathsf{ck}}(p)) = fv(p) \cup \kappa(p)$ and $bv(\overline{\mathsf{ck}}(p)) = bv(p)$, where κ is defined in Table 1.

To associate a stochastic automaton to a given term, we need to define the different parts of the stochastic automaton. We start by defining the clock setting function κ and the set of edges $\longrightarrow$ as the least relations satisfying the rules in Table 1. However, not all the processes can have a straightforward stochastic automaton as a semantic interpretation. To do so, clock names must be considered with care as we see as follows. Consider the process

$$p \equiv \{\!|x_G|\!\}(a; \{x_G\} \mapsto (\{\!|x_G, y_H|\!\}\{y_H\} \mapsto b; \mathbf{stop})) \qquad (1)$$

The second occurrence of x_G is intended to be bound to the outermost clock setting as shown by the grey arrow. Using the rules in Table 1, the following stochastic automaton would be obtained



In this sense, x_G would be captured by the innermost clock setting as shown by the black arrow in (1). Therefore, we consider that clocks are different if they are set in different places, although they may have the same name. Clock capture may also occur in contexts with summations and parallel composition.

Capture of variables is a well known problem in languages with variables that can be solved by considering terms modulo α-congruence. It is indeed the solution that we adopt, although for recursive terms special care is needed. However, we would like to characterise processes which have conflict of variables since it is also relevant for the axiomatisation. In fact, we will see that the axiomatisation is sound and complete for structural bisimulation, and hence it becomes important that the scope and binding of clocks is correct since this relation considers clock names.

A first approach to characterise processes with conflict of variables could be purely syntactic. However, this notion turns out to be too strong. Although process p above is problematic, process $p||_{\{a\}}\mathbf{stop}$ does not introduce any problem since its associated stochastic automaton will not have any outgoing edge. In fact, it is evidently equivalent to $\{\!|x_G|\!\}\mathbf{stop}$.

Therefore, we need a dynamic characterisation of processes which do not have conflict of variables. A process p does not have *conflict of variables* if no clock is illegally captured, that is, for every path $p = p_0 \xrightarrow{a_1, C_1} p_1 \xrightarrow{a_2, C_2} p_2 \cdots p_{n-1} \xrightarrow{a_n, C_n} p_n$, for every subterm q of p_i, $i \in \{0, \ldots, n\}$, which is not in the scope of a prefix, the following conditions holds:

1. $q \equiv C \mapsto q'$ implies $C \cap \kappa(q') = \emptyset$
2. $q \equiv q' + q''$ implies $\kappa(q') \cap \kappa(q'') = fv(q') \cap \kappa(q'') = \kappa(q') \cap fv(q'') = \emptyset$
3. $q \equiv q'||_A q''$, $q' \lfloor\!\lfloor_A q''$, or $q'|_A q''$ implies $bv(q') \cap var(q'') = var(q') \cap bv(q'') = \emptyset$

Definition 12 Let p be a process without conflict of variables. The *stochastic automaton associated to* p is defined by $[\![p]\!] \stackrel{\text{def}}{=} (\spadesuit^{\overline{\mathsf{ck}}}, p, C, \mathbf{A}, \longrightarrow, \kappa, F)$, where $\longrightarrow$ and κ are defined in Table 1, and C, $\mathbf{A}$ and F are defined as for the syntax of $\spadesuit$. □

The reader is invited to check that the processes of the switch system defined in Example 3 do not have conflict of variables, and that the stochastic automaton associated to the process *System* is the one depicted in Figure 1 modulo the identification of $\overline{\mathsf{ck}}(\overline{\mathsf{ck}}(p))$ and $\overline{\mathsf{ck}}(p)$, for all p.

As we said, the restriction to processes which do not have conflict of variables is not an actual problem, since we can always properly rename clocks in any (guardedly defined) process to obtain another process which does not have conflict of variables. With "properly" we mean that the distribution function associated to the clock must be preserved. For instance, p can be α-converted into $\{\!|x_G|\!\}(a; \{x_G\} \mapsto (\{\!|z_G, y_H|\!\}\{y_H\} \mapsto b; \mathbf{stop}))$.

Relating stochastic automata and terms. In the following we study the connection between stochastic automata and recursive specifications. We show that guarded recursive specifications and finitely branching stochastic automata are equally expressive. In order to do so, we need to define the notion of guarded specification and finitely branching. A process variable is *guarded* if all its occurrences appear in a context of a prefix. A recursive specification E is *guarded* if $X = p \in E$ implies that all variables in p are guarded. A stochastic automaton is *finitely branching* if for every location s, its set of outgoing arrows $\{s \xrightarrow{a,C} s' \mid a \in \mathbf{A}, C \in \mathcal{C}, s' \in \mathcal{S}\}$ is finite. Now we can state:

Proposition 13 *Let E be a guarded recursive specification with root X. Assume E does not have conflict of variables. Then $[\![X]\!]$ is finitely branching.*

♤ has the property of expressing any (finitely branching) stochastic automaton. The proof of Theorem 14 follows closely the ideas of a similar theorem in [7].

Theorem 14 *For every finitely branching stochastic automaton SA there is a guarded recursive specification E with root X such that the reachable part of SA and the reachable part of $[\![X]\!]$ are isomorphic.*

Bisimulations in ♤. We extend the notion of probabilistic bisimulation, potential bisimulation and structural bisimulation to ♤ in the obvious way. Let $p, q \in$ ♤. We say that p and q are *probabilistically*, *potentially*, or *structurally bisimilar*, if their respective associated stochastic automata are. We use the notation $p \underline{\leftrightarrow} q$, $p \underline{\leftrightarrow}_P q$, and $p \overset{\leftrightarrow}{\sim} q$, respectively.

In Section 3, we have already anticipated that probabilistic bisimilarity is not a congruence. This is shown by the following example.

Example 4 *$\underline{\leftrightarrow}$ is not a congruence for parallel composition.* Processes $p_1 \equiv a; \mathbf{stop} + \{\!|x_G|\!\}\{x_G\}\mapsto b; \mathbf{stop}$ and $p_2 \equiv a; \mathbf{stop} + \{\!|x_G|\!\}\{x_G\}\mapsto c; \mathbf{stop}$ $(b \neq c)$ are probabilistically bisimilar if $G(0) = 0$, since in both cases, only the action a at time 0 can be performed. However, $p_1||_{\{a\}}\mathbf{stop}$ and $p_2||_{\{a\}}\mathbf{stop}$ are not bisimilar. In this context, the execution of action a is preempted since there is no possible synchronisation, and b or c may happen (at a certain time greater than 0). This example is depicted in Figure 3. The reader is invited to check that $\underline{\leftrightarrow}$ is neither a congruence for the triggering condition. □

This is precisely the kind of situations that occur when dealing with open systems, and hence they justify the introduction of potential bisimulation. The next theorem states that $\underline{\leftrightarrow}_P$ is a congruence for the operations in ♤.

Theorem 15 *Let $p, q \in$ ♤ such that $p \underline{\leftrightarrow}_P q$. For any context* $\mathsf{C}[\;]$ *containing the operations* $\mathbf{stop}$, $a;$, $C\mapsto$, $\{\!|C|\!\}$, $+$, $||_A$, $\lfloor\!\lfloor_A$, $|_A$, *or* $[f]$, *and such that* $\mathsf{C}[p]$ *and* $\mathsf{C}[q]$ *do not have conflict of variables, it holds that* $\mathsf{C}[p] \underline{\leftrightarrow}_P \mathsf{C}[q]$.

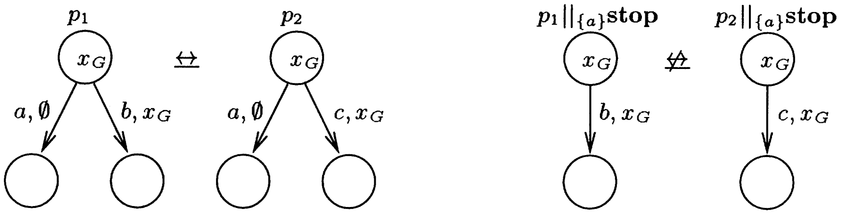


Figure 3 Bisimilarity is not a congruence

Besides, we have the result that structural bisimulation is a congruence for all the operations (including $\overline{\mathsf{ck}}$).

Theorem 16 *Let $p, q \in \spadesuit^{\overline{\mathsf{ck}}}$ such that $p \underset{\sim}{\leftrightarrow} q$. For any context* $\mathsf{C}[\ ]$ *containing the operations* **stop**, *$a;$, $C\mapsto$, $\{\!|C|\!\}$, $+$, $||_A$, $\lfloor\!\lfloor_A$, $|_A$, $[f]$, or $\overline{\mathsf{ck}}$, it holds that* $\mathsf{C}[p] \underset{\sim}{\leftrightarrow} \mathsf{C}[q]$.

The proof of Theorem 15 is quite involved since it has to be done in a traditional way: a relation is given for each case and it is proven to be a potential bisimulation (up to $\underline{\leftrightarrow}_P$). Instead, the proof that $\underset{\sim}{\leftrightarrow}$ is a congruence uses the results of [3] since rules in Table 1 can be easily rewritten into *path* format.

Another important result that we would like to highlight is that proper renaming of variables preserves potential bisimulation. It is important indeed because it justifies the fact that we can always properly rename clocks to obtain processes without conflict of variables as we claimed before.

Structural axioms. In this paragraph, we give a set of axioms for $\spadesuit$. We study the so-called structural axioms. These axioms preserve structural bisimulation. We show that they allow to rewrite any (closed) term into a basic or normal form. Moreover, we show that parallel composition and renaming can be eliminated in favour of the basic operations **stop**, $a;$, $C\mapsto$, $\{\!|C|\!\}$ and $+$. When convenient, we consider terms modulo α-conversion.

Axioms in Table 2 can be explained as follows. The choice is commutative (**A1**) and associative (**A2**). Axiom **A3** states a kind of idempotency of $+$ and **A4** states that **stop** is the neutral element for $+$. Axioms **T1**–**T5** show the way in which triggering conditions can be simplified. **T3** defines how to reduce nested triggering conditions into only one. Axioms **T4** and **T5** say how to move clock settings and summations out of the scope of a guard. **S1** says that it is irrelevant to set an empty set of clocks. **S2** gathers all the clocks settings in only one operation and **S3** moves clocks settings out of the scope of a summation.

Axioms **R1**–**R5** define the renaming operation. The way in which they operate is more or less standard in process algebra.

Axioms **PC1** and **PC2** move clock settings out of the scope of the parallel composition. This is necessary because when expanding parallel composition

Table 2 Structural axioms for ♤

A1	$p+q=q+p$	**R1**	$\mathbf{stop}[f]=\mathbf{stop}$
A2	$(p+q)+r=p+(q+r)$	**R2**	$(a;p)[f]=a;(p[f])$
A3	$a;p+a;p=a;p$	**R3**	$(C\mapsto p)[f]=C\mapsto p[f]$
A4	$p+\mathbf{stop}=p$	**R4**	$(\{\!\mid C\mid\!\}p)[f]=\{\!\mid C\mid\!\}p[f]$
T1	$C\mapsto\mathbf{stop}=\mathbf{stop}$	**R5**	$(p+q)[f]=p[f]+q[f]$
T2	$\emptyset\mapsto p=p$		
T3	$C\mapsto C'\mapsto p=C\cup C'\mapsto p$		
T4	$C\mapsto\{\!\mid C'\mid\!\}p=\{\!\mid C'\mid\!\}C\mapsto p$	if $C\cap C'=\emptyset$	
T5	$C\mapsto(p+q)=C\mapsto p+C\mapsto q$		
S1	$\{\!\mid\emptyset\mid\!\}p=p$		
S2	$\{\!\mid C\mid\!\}\{\!\mid C'\mid\!\}p=\{\!\mid C\cup C'\mid\!\}p$		
S3	$\{\!\mid C\mid\!\}p+\{\!\mid C'\mid\!\}q=\{\!\mid C\cup C'\mid\!\}(p+q)$	if $C\cap fv(q)=C'\cap fv(p)=\emptyset$	
PC1	$(\{\!\mid C\mid\!\}p)\|\|_A q=\{\!\mid C\mid\!\}(p\|\|_A q)$	if $C\cap \mathsf{var}(q)=\emptyset$	
PC2	$p\|\|_A(\{\!\mid C\mid\!\}q)=\{\!\mid C\mid\!\}(p\|\|_A q)$	if $C\cap \mathsf{var}(p)=\emptyset$	
PC3	$p\|\|_A q=p\lfloor\!\lfloor_A q+q\lfloor\!\lfloor_A p+p\|_A q$	if $\mathsf{B}'(p)\wedge\mathsf{B}'(q)$	
LM1	$\mathbf{stop}\lfloor\!\lfloor_A q=\mathbf{stop}$	if $\mathsf{B}'(q)$	
LM2	$a;p\lfloor\!\lfloor_A q=\mathbf{stop}$	if $\mathsf{B}'(q)\wedge a\in A$	
LM3	$a;p\lfloor\!\lfloor_A q=a;(p\|\|_A q)$	if $\mathsf{B}'(q)\wedge a\notin A$	
LM4	$(C\mapsto p)\lfloor\!\lfloor_A q=C\mapsto(p\lfloor\!\lfloor_A q)$		
LM5	$(\{\!\mid C\mid\!\}p)\lfloor\!\lfloor_A q=\{\!\mid C\mid\!\}(p\lfloor\!\lfloor_A q)$	if $C\cap \mathsf{var}(q)=\emptyset$	
LM6	$p\lfloor\!\lfloor_A(\{\!\mid C\mid\!\}q)=\{\!\mid C\mid\!\}(p\lfloor\!\lfloor_A q)$	if $C\cap \mathsf{var}(p)=\emptyset$	
LM7	$(p+q)\lfloor\!\lfloor_A r=(p\lfloor\!\lfloor_A r)+(q\lfloor\!\lfloor_A r)$	if $\mathsf{B}'(r)$	
CM1	$p\|_A q=q\|_A p$		
CM2	$\mathbf{stop}\|_A\mathbf{stop}=\mathbf{stop}$		
CM3	$\mathbf{stop}\|_A a;q=\mathbf{stop}$		
CM4	$a;p\|_A b;q=\mathbf{stop}$	if $a\notin A$	
CM5	$a;p\|_A a;q=a;(p\|\|_A q)$	if $a\in A$	
CM6	$(C\mapsto p)\|_A q=C\mapsto(p\|_A q)$		
CM7	$(\{\!\mid C\mid\!\}p)\|_A q=\{\!\mid C\mid\!\}(p\|_A q)$	if $C\cap \mathsf{var}(q)=\emptyset$	
CM8	$(p+q)\|_A r=(p\|_A r)+(q\|_A r)$	if $\mathsf{B}'(r)$	
UB1	$\mathsf{B}'(\mathsf{stop})$	**UB3**	$\dfrac{\mathsf{B}'(p)}{\mathsf{B}'(C\mapsto p)}$
UB2	$\mathsf{B}'(a;p)$	**UB4**	$\dfrac{\mathsf{B}'(p)\quad\mathsf{B}'(q)}{\mathsf{B}'(p+q)}$

in terms of summations, we do not want to duplicate clocks. Duplicating clocks would transform processes without conflict of variables into (semantically different!) processes with conflict of variables. **PC3** decomposes the parallel composition in terms of the left merge and the communication merge provided no clock setting is wrongly duplicated. **LM1**–**LM7** and **CM1**–**CM8** define the left merge and the communication merge respectively. The predicate B' defined by the rules **UB1**–**UB4** encodes information about $\overline{\mathsf{ck}}$. In fact, for all guarded processes such that $\mathsf{B}'(p)$ can be proven using axioms **UB1**–**UB4**, it holds that $\overline{\mathsf{ck}}(p)\mathrel{\underset{\sim}{\leftrightarrow}} p$. We do not want to have $\overline{\mathsf{ck}}$ in our axiomatisation since it does not preserve $\underline{\leftrightarrow}_P$.

We observe that idempotency is not generally true in ♤. Consider the process

$p \equiv \{\!|x_G|\!\}\{x_G\}\mapsto a; \mathbf{stop}$ where G is uniform on $[0,2]$. The probability that a occurs in the interval $[0,1]$ in process p is $\frac{1}{2}$, while in process $p+p$ such probability is $\frac{3}{4}$. It follows that $p \not\leftrightarrow_{\!\!\!\sim} p+p$ and so they are not related by finer bisimulations. Although axiom **A3** already states a notion of idempotency a more general property is $C\mapsto a;p = C\mapsto a;p + C\mapsto a;p$ which we call **A3'** and can be derived from the axioms **A3** and **T5**.

Axioms in Table 2 are sound for structural bisimulation. An immediate consequence is that they are also sound for potential and probabilistic bisimulation. Besides, it can be easily checked that the axioms preserve the property of non-conflict of variables. The side conditions in Table 2 are essential for this to hold.

Theorem 17 *Let $p, q \in \spadesuit$ such that $p = q$ can be proved from axioms in Table 2. Then we have,*

1. *p does not have conflict of variables if and only if neither q does; and*
2. *if they do not have conflict of variables, then $p \underset{\sim}{\leftrightarrow} q$.*

An interesting property that is derived from these axioms is that every term can be expressed in a normal form.

Definition 18 Define the set $B \subseteq \spadesuit$ of *basic terms* inductively as follows:

- $\mathbf{stop} \in B'$
- $p \in B$, $C \in \wp_{\text{fin}}(\mathcal{C})$ and $a \in \mathbf{A} \implies C\mapsto a;p \in B'$
- $p, q \in B' \implies p+q \in B'$
- $p \in B'$ and $C \in \wp_{\text{fin}}(\mathcal{C}) \implies \{\!|C|\!\}p \in B$

$B' \subseteq \spadesuit$ is the set of all terms whose clock settings are all within the scope of a prefix construction. (Notice that $p \in B'$ implies $\mathsf{B}'(p)$.) A basic term has the general form (modulo **A1**, **A2**, **A3'** and **A4**)

$p \equiv \{\!|C|\!\}\ (\sum_{i\in I} C_i\mapsto a_i; p_i)$

where each p_i is a basic term, and $\sum_{i\in I} q_i \stackrel{\text{def}}{=} q_1 + \cdots + q_n$ for $I = \{1,\ldots,n\}$. In particular, $\sum_{i\in\emptyset} q_i \stackrel{\text{def}}{=} \mathbf{stop}$. □

Theorem 19 *Let $\spadesuit^c$ ($\subseteq \spadesuit$) be the set of all* finite *(or* closed*) terms, i.e., terms which do not contain process variables. For every term $p \in \spadesuit^c$ there is a term $q \in B$ such that $p = q$ can be proven by means of the basic axioms and α-conversion.*

The set of axioms given in Table 2 is complete for structural bisimulation on the set $\spadesuit^c$. Theorem 19 is essential for the proof of completeness. Since α-conversion does not imply structural bisimulation, we must ensure that it is not used in the proof of Theorem 19. To do so, it is enough to restrict to terms without conflict of variables because of Theorem 17.

Theorem 20 *Let $p, q \in \spadesuit^c$ be two terms without conflict of variables. Suppose $p = q$ can be derived from the axioms in Table 2 (but not α-conversion!). Then $p \overset{\leftrightarrow}{\sim} q$.*

One of the reasons why many approaches to stochastic process algebras stick to only exponential distributions [16, 15, 4, ...] is that general distributions do not preserve Milner's expansion law in their models. In other cases, combining the expansion law with general distributions lead to infinite and sometimes quite complicated models [11, 13, 20]. In our case, the expansion law is inherent in the model and the way parallel composition is defined, and can be smoothly derived from the axioms as stated by the following theorem.

Theorem 21 (Expansion Law) *Let $p, q \in \spadesuit$ such that $p = \{\!|C|\!\}p'$ and $q = \{\!|C'|\!\}q'$ with $p' = \sum C_i \mapsto a_i; p_i$ and $q' = \sum C'_j \mapsto b_j; q_j$. Suppose $p||_A q$ does not have conflict of variables. From the axioms in Table 2 we can derive*

$$\begin{aligned} p||_A q \;=\; \{\!|C \cup C'|\!\} \Big(& \textstyle\sum_{a_i \notin A} C_i \mapsto a_i; (p_i||_A q') \;+\; \sum_{b_j \notin A} C'_j \mapsto b_j; (p'||_A q_j) \\ & + \textstyle\sum_{a_i = b_j \in A} (C_i \cup C'_j) \mapsto a_i; (p_i||_A q_j) \Big) \end{aligned}$$

For clarity, we did not include the renaming operation. This, however, could be done straightforwardly.

Example 5 The reader is invited to check that, using the axioms, the process *System* of Example 3 can be re-written into the following expression.

$$\begin{aligned} \mathit{System} &= \{\!|x_G|\!\}\{x_G\} \mapsto on; \mathit{Sys}_{on} \\ \mathit{Sys}_{on} &= \{\!|x_G, y_K|\!\}(\{x_G\} \mapsto on; \mathit{Sys}_{on} + \{y_K\} \mapsto \mathit{off}; \mathit{Sys}_{\mathit{off}}) \\ \mathit{Sys}_{\mathit{off}} &= \{x_G\} \mapsto on; \mathit{Sys}_{on} \end{aligned}$$

Its associated stochastic automaton is indeed the one depicted in Figure 1. □

5 FURTHER DISCUSSIONS

Related work. Apart from the Markovian process algebras [16, 15, 4, ...], some general stochastic process algebras have been introduced.

TIPP [11] is the earliest approach to the general case. Its syntax has the integrated prefix $a_F; p$ which in $\spadesuit$ corresponds to $\{\!|x_F|\!\}\{x_F\} \mapsto a; p$. Its semantics is based on labelled transition systems in which transitions are decorated with the associated distribution function and, to keep track of the execution of parallel processes, a number that indicates how many times an action has not been chosen to execute. This number introduces infinite semantic objects, even for simple regular processes. [20] has followed a similar approach to give semantics to a stochastic extension of the π-calculus. In this case transitions are decorated with locality information to keep track which process performed it.

In [13], a process algebra for discrete event simulation is introduced. The concerns of randomly setting a timer, expiration of such a timer, and actual activity are splitted in a rather similar way to ours. The semantic model is similar to our PTSs but with explicit time transitions, and hence semantic objects are usually highly infinite. The process algebra includes an urgent and a delayable prefixing, so its interpretation combines both views of closed and open system.

[6] studies a semantic for a process algebra similar to TIPP in terms of a stochastic extension of event structures. This model seems to be more natural to deal with general distributions since activities that are not causally dependent (i.e. concurrent activity) are not related in the model, contrarily of what occurs in interleaving based models. However, recursive processes always have associated an infinite semantic object.

A general semi-Markovian process algebra based on EMPA [4] is discussed in [5]. Terms in this process algebra have semantics in an interleaving based model. Finiteness of the associated semantic object is kept in a reasonable way. As a price to pay, the way to give semantics is quite cumbersome and not only the transitions are decorated with many information (as for instance the locality of the occurrence of an action) but also the states.

Among the above enumerated stochastic process algebras, [13] is the closest to ♤. As ♤, [13] also allows non-determinism. In all the other cases (including the Markovian process algebras), choice is always solved either probabilistically or by the race condition. We also mention that none of [11, 20, 6, 5] discusses an axiomatic theory for their respective stochastic process algebras.

Conclusions and further work. We introduced new models to analyse stochastic and real-time systems. We discussed in depth a stochastic process algebra whose expressivity is richer than existing ones. We showed that this process algebra and its underlying semantic model, the stochastic automata, are equally expressive. We gave an axiomatisation and we showed that the expansion law can be straightforwardly derived from them. Besides, we have defined a general probabilistic transition system. We have used it as semantic model of the stochastic automata. In fact, we gave two different ways of assigning a PTS to a stochastic automaton, so we may understand systems either as closed or open.

It is worthwhile to notice that ♤ is a conservative extension of the basic CCS (i.e. the sublanguage containing only prefixing, summation and stop) in both semantic and axiomatic sense. Besides, it can be proven that equivalences $\underline{\leftrightarrow}$, $\underline{\leftrightarrow}_P$, and $\underset{\sim}{\leftrightarrow}$ turn out to be the same in the non-stochastic sublanguage of ♤, that is, the set of all the terms in which operations $\{\!|C|\!\}$ and $C\mapsto$ do not occur.

We should remark a couple of works we have already done regarding stochastic automata and discrete event simulation. The first is that the actual behaviour of stochastic automata leads to an algorithm for discrete event simulation. We use the notion of adversaries or schedulers [25, 23] to resolve non-deterministic choices. Since parallel composition of stochastic automata can be easily defined (actually, it is the one of $\spadesuit$), the simulation algorithm can compose the complete stochastic automaton on the fly, which reduces the state space explosion problem. Secondly, we already know that stochastic automata properly contain a wide class of GSMPs. We will report in detail about these works in the near future.

From the results reported in this paper and the observations just mentioned, we believe that our models are quite suitable to specify and analyse stochastic systems and real-time systems. But to go further in this direction many things have still to be done. First of all the axiomatisation for $\spadesuit$ is not sufficient as it is. Axioms for potential bisimulation as well as laws for probabilistic bisimulation have to be introduced. A clear example of this need is that terms p, and $\{\!|C|\!\}p$ are potentially bisimilar provided $C \cap \mathit{fv}(p) = \emptyset$. However, equality $p = \{\!|C|\!\}p$ cannot be proved from the axioms in Table 2, which is reasonable because p and $\{\!|C|\!\}p$ are not necessarily structurally bisimilar.

As we pointed out we have a method to simulate stochastic automata (and hence terms in $\spadesuit$). However, analytical methods are far more effective to study the correctness of a system. Usually errors are events with low probability, so the use of simulation may not guarantee that they are not present or that their probability is low enough to be considered. Model checking has proven to be a powerful tool to verify timed systems. Some early papers like [1] have shown the possibility of borrowing ideas from model checking on timed automata and applying them to stochastic systems. Our work will also address the use of model checking on stochastic automata. Besides, we will investigate on the possibility of borrowing from analytical methods already used in the performance analysis community, so it can be applied to study analytically the performance of systems modelled in $\spadesuit$.

REFERENCES

1. R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems. In J. Leach Albert, B. Monien, and M. Rodríguez, eds., *Proceedings 18th ICALP,* Madrid, *LNCS* 510, pp 113–126. Springer, 1991.
2. R. Alur and D. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.
3. J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, ed., *Proceedings CONCUR 93,* Hildesheim, Germany, LNCS 715, pp 477–492. Springer, 1993.
4. M. Bernardo and R. Gorrieri. Extended Markovian process algebra. In U. Montanari and V. Sassone, eds., *Proceedings CONCUR 96,* Pisa, Italy, LNCS 1119, pp 314–330. Springer, 1996.
5. M. Bravetti, M. Bernardo, and R. Gorrieri. From EMPA to GSMPA: allowing for general distributions. In E. Brinksma and A. Nymeyer, eds., *Proceedings PAPM'97*, pp 17–33. University of Twente, June 1997.

6. E. Brinksma, J.-P. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, 38(6):552–565, 1995.
7. P.R. D'Argenio and E. Brinksma. A calculus for timed automata (Extended abstract). In B. Jonsson and J. Parrow, eds., *Proceedings FTRTFT'96,* Uppsala, Sweden, LNCS 1135, pp 110–129. Springer, 1996.
8. P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems. Technical Report CTIT-98-02. University of Twente, 1998.
9. R.J. van Glabbeek, S.A. Smolka, and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Infor. & Comput.*, 121:59–80, 1995.
10. P.W. Glynn. A GSMP formalism for discrete event simulation. *Proceedings of the IEEE*, 77(1):14–23, 1989.
11. N. Götz, U. Herzog, and M.Rettelbach. TIPP - Introduction and application to protocol performance analysis. In H. König, ed., *Formale Beschreibungstechniken für verteilte Systeme*, FOKUS series. Saur Publishers, 1993.
12. H.A. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings 11th IEEE Real-Time Systems Symposium*, pp 278–287, Lake Buena Vista, Florida, December 1990.
13. P. Harrison and B. Strulo. Stochastic process algebra for discrete event simulation. In F. Bacelli, A. Jean-Marie, and I. Mitrani, eds., *Quantitative Methods in Parallel Systems*, Esprit Basic Research Series, pp 18–37. Springer, 1995.
14. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Infor. & Comput.*, 111:193–244, 1994.
15. H. Hermanns and M.Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In *Proceedings PAPM'94*, pp 71–87. University of Erlangen, July 1994.
16. J. Hillston. *A Compositional Approach to Performance Modelling.* Distinguished Dissertation in Computer Science. Cambridge University Press, 1996.
17. K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Infor. & Comput.*, 94:1–28, 1991.
18. R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.
19. A. Pnueli and L.D. Zuck. Probabilistic verification. *Infor. & Comput.*, 103:1–29, 1993.
20. C. Priami. Stochastic π-calculus with general distributions. In [21], pp 41–57.
21. M. Ribaudo, ed. *Proceedings PAPM'96,* Torino, Italy. Università di Torino, 1996.
22. J.J.M.M. Rutten and E. de Vink. Bisimulation for probabilistic transition systems: a coalgebraic approach (extended abstract). In *Proceedings* 24th *ICALP,* Bologna, LNCS 1256, pp 460–470. Springer, 1997.
23. R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
24. A.N. Shiryaev. *Probability.* Springer, second edition, 1996.
25. M.Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proceedings* 26th *FOCS,* Portland, pp 327–338. IEEE Comp. Soc. Press, 1985.
26. W. Whitt. Continuity of generalized semi-Markov processes. *Math. Oper. Res.*, 5:494–501, 1980.

BIOGRAPHY

Pedro R. D'Argenio graduated as Computing Analyst and Licentiate in Computer Science from the National University of La Plata, Argentina, in 1993 and 1994, respectively. Until 1995, he held research and teaching positions at the Department of Computer Science of the National University of La Plata. Since 1995, he is a Ph.D. student at the Department of Computer Science of the University of Twente, The Netherlands. His current research subjects include specification, verification, and validation of real-time, stochastic, and distributed systems as well as formal methods applied to performance analysis.

Joost-Pieter Katoen received his M.Sc. degree (with honours) and Ph.D. degree in Computer Science from the University of Twente, The Netherlands, in 1987 and 1996, respectively. From 1988 to 1990 he was a postgraduate student at the Eindhoven University of Technology, The Netherlands. He joined Philips Research Laboratories Eindhoven from 1990 to 1992. Since 1997, he is assistant professor at the Faculty of Computer Science of the University of Erlangen-Nürnberg. His current research interests include specification and verification of real-time and probabilistic systems, semantics, and performance analysis based on formal methods.

Ed Brinksma received his M.Sc. degree (cum laude) in Mathematics from the University of Groningen, The Netherlands, in 1982. In 1982 he joined the Department of Computer Science at the University of Twente as an assistant professor where he got his Ph.D. in Computer Science in 1988. In the period 1983-1989 he was the chairman of the committee of the International Organisation for Standardisation (ISO) that was responsible for the definition of the formal specification technique LOTOS. Since 1991, he is a full professor, occupying the chair in Formal Methods and Tools. His main research interest is the application of formal methods to the design and analysis of distributed systems. His current research topics include the application of formal methods to testing, the relation between formal methods and performance analysis, the application of correctness preserving transformations to realistic designs, and tool-oriented design of specification formalisms.