Stochastic Simulation of Event Structures

Joost-Pieter Katoen[a], Ed Brinksma[a], Diego Latella[b] and Rom Langerak[a]

[a]Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
e-mail: {katoen, brinksma, langerak}@cs.utwente.nl

[b]CNUCE Istituto del CNR, Via Santa Maria 36, 56126 Pisa, Italy
e-mail: d.latella@cnr.cnuce.it

Abstract

Currently the semantics of stochastic process algebras are defined using (an extension) of labelled transition systems. This usually results in a semantics based on the interleaving of causally independent actions. The advantage is that the structure of transition systems closely resembles that of Markov chains, enabling the use of standard solution techniques for analytical and numerical performance assessment of formal specifications. The main drawback is that distributions are restricted to be exponential. In [2] we proposed to use a partial-order semantics for stochastic process algebras. This allows the support of non-exponential distributions in the process algebra in a perspicuous way, but the direct resemblance with Markov chains is lost. This paper proposes to exploit discrete-event simulation techniques for analyzing our partial-order model, called stochastic event structures. The key idea is to obtain from event structures so-called (time-homogeneous) generalized semi-Markov processes. Such processes allow for the generation of simulation runs and are amenable to fast simulation techniques.

1 Introduction

Nowadays, performance analysis has a well-recognized position in the design of complex distributed systems. Usually, performance models like queueing networks and Markov chains are developed by abstracting from the system specification that is being used for the qualitative analysis and functional design. In this way, obtaining performance models from system specifications is largely based on human ingenuity and experience. The increasing complexity and magnitude of systems complicates this task considerably. Therefore, it is suggested to obtain performance models in a *compositional* way by exploiting the structure of the system specification at hand. Since process algebras are typically characterized by the presence of a number of powerful composition operators that facilitate the development of well-structured specifications, these needs resulted in the investigation of *stochastic process algebras.* These formalisms are extensions of standard process algebras, such as CSP, CCS, and LOTOS, where the time of occurrence of actions is determined by stochastic variables.

Traditionally, the semantics of process algebras is based on the interleaving of causally independent actions. This means that parallel composition (denoted $|||$) is interpreted as a combination of alternative composition (denoted $+$) and sequencing (denoted $;$), e.g.,

$$a\,;\,P\,|||\,b\,;\,Q = a\,;\,(P\,|||\,b\,;\,Q) + b\,;\,(a\,;\,P\,|||\,Q) \tag{1}$$

where a and b denote actions and P and Q processes. This principle—in its full form known as the expansion theorem—is naturally supported by labelled transition systems, the most popular semantical model for interleaving semantics.

The semantics of stochastic—usually Markovian—process algebras such as PEPA [8], MTIPP [4, 7], and EMPA [1] are defined using an extension of labelled transition systems. The structure of transition systems closely resembles that of standard Markov chains, which is an advantage when trying to obtain a performance model directly from the formal specification. In this way the standard techniques and tools for obtaining e.g. steady state probabilities for ergodic Markov chains can be adopted for performance assessment of the formal specification. In addition, the elegant—memoryless—property of exponential distributions enables a smooth incorporation of such distributions into an interleaving setting, since

$$a_\lambda\,;\,P\,|||\,b_\mu\,;\,Q = a_\lambda\,;\,(P\,|||\,b_\mu\,;\,Q) + b_\mu\,;\,(a_\lambda\,;\,P\,|||\,Q) \tag{2}$$

Here, a_λ denotes that a occurs after a delay determined by an exponentially distributed random variable with rate λ (and similar for b). The reason that this law holds is that the time until the occurrence of b after the occurrence of a is still distributed according to μ irrespective of how much time has elapsed until a occurred. (By symmetry, a similar reasoning applies when b occurs before a.)

The interleaving of causally independent actions, however, complicates the incorporation of more general (non-memoryless) distributions in transition systems considerably. If we, for instance, generalize the above idea by equipping actions with generally distributed random variables (denoted a_U for random variable U) we obtain that

$$a_U\,;\,P\,|||\,b_V\,;\,Q \neq a_U\,;\,(P\,|||\,b_V\,;\,Q) + b_V\,;\,(a_U\,;\,P\,|||\,Q) \tag{3}$$

In case the memoryless property is not satisfied the residual lifetime of V after the occurrence of a must be computed in order to correctly deduce the time until b's occurrence (and, by symmetry, an analogous procedure must be carried out for the residual lifetime of U if b occurs first).

We may thus conclude that exponential distributions and interleaving semantics fit well together, but that general distributions and interleaving semantics do not. From a practical point of view, however, the incorporation of general distributions is considered to be essential in order to faithfully model, for instance, high-speed communication networks and workflow management systems.

In [2] we showed that using an alternative—*noninterleaving*—semantics general distributions can be incorporated in a perspicuous and elegant way. For this purpose we used *stochastic event structures*, a generalization of bundle event structures where events and bundles, i.e., causal relations between events, are decorated with random variables. The (non-severe) condition obtained from our approach is that the class of distribution

functions must be closed under product and have a unit element for this operation. For example, phase-type distributions [14] and distributions of exponential polynomial form [16] satisfy these conditions. Both classes include frequently used distributions such as hyper- and hypo-exponential, Erlang and Coxian distributions and are widely used in the performance analysis community.

Unfortunately, while moving to a noninterleaving semantics the convenient direct relationship between the semantical model of stochastic process algebras and Markov-like models (such as semi-Markov chains where we would be able to deal with arbitrary distributions) is lost. This paper, therefore, proposes a recipe to construct discrete-event systems from stochastic event structures, such that *discrete-event simulation* techniques can be exploited for performance assessment. Discrete-event systems are characterized by the fact that the state of the model only changes at discrete moments in time. Moreover, the behaviour of such model is completely determined by the state changes and the points at which they occur.

This paper presents a recipe for obtaining (time–homogeneous) *generalized semi-Markov processes* (GSMPs) starting from stochastic event structures. GSMPs constitute a mathematical framework for the study of discrete-event systems [3, 19]. A mapping of stochastic event structures onto GSMPs enables the generation of simulation runs from which—using the standard simulation techniques for discrete-event systems—performance results can be obtained. A large class of time-homogeneous GSMPs is amenable to fast simulation techniques, in particular so-called regenerative simulation.

This paper is organized as follows. In Section 2 we recapture the notion of stochastic event structures, the noninterleaving semantical model that we proposed in [2]. Generalized semi-Markov processes are briefly explained in Section 3. The core of the paper is Section 4 that presents our recipe for mapping stochastic event structures onto GSMPs and that presents a simulation algorithm based on this mapping. Section 5 is a prospective view on using regenerative simulation techniques for infinite stochastic event structures. Section 6 introduces our (non-Markovian) stochastic process algebra and shows by means of example how to obtain event structures for process algebra expressions in a compositional way. Section 7 shows which implications this semantical mapping has for the generation of GSMPs. Section 8 concludes the paper and proposes some ideas for further research.

2 Stochastic event structures

Event structures are a prominent noninterleaving model for concurrency. We use Langerak's *bundle event structures* [12], an adaptation of Winskel's event structures [22] to fit the specific requirements of multi-party synchronization. Bundle event structures (or, simply: event structures) consist of *events* labelled with actions—an event modelling the occurrence of its action—together with relations of causality and conflict between events. System runs can be modelled as partial orders of events satisfying certain constraints posed by the causality and conflict relations between the events.

Conflict is a symmetric binary relation, denoted #, between events. The intended meaning of $e \# e'$ is that either e or e' can appear in a system run but not both.[1] *Causality* is represented by a relation, denoted $\mapsto$, between a set X of events, that are pairwise in conflict, and an event e. The interpretation is that if e happens in a system run, exactly one event in X has happened before (and caused e). This enables us to uniquely define a causal ordering between the events in a system run. When there is neither a conflict nor a causal relation between events they are independent.

1. Definition. (*Bundle event structure*)

A *bundle event structure* $\mathcal{E}$ is a quadruple $(E,\#,\mapsto,l)$ with E a set of *events*, $\# \subseteq E \times E$ the (irreflexive and symmetric) *conflict* relation, $\mapsto \subseteq \mathcal{P}(E) \times E$ the *causality* relation, and $l : E \to L$ the *action-labelling* function, where L is a set of action labels, such that $\mathcal{E}$ satisfies $\forall\, X \subseteq E, e \in E$:

$$X \mapsto e \;\Rightarrow\; (\forall\, e_i,e_j \in X : e_i \neq e_j \;\Rightarrow\; e_i \# e_j) \quad .$$

□

We usually denote bundle (X,e) by $X \mapsto e$ and an event labelled a by e_a. The constraint specifies that for bundle $X \mapsto e$ all events in X are in mutual conflict. Bundle event structures are graphically represented in the following way. Events are denoted as dots; near the dot the action label is given. Conflicts are indicated by dotted lines between representations of events. Bundle $X \mapsto e$ is indicated by drawing an arrow from each event in X to e and connecting all arrows by small lines.

2. Example. Figure 1(a) has bundles $\{ e_a \} \mapsto e_c$, $\{ e_b \} \mapsto e_c$, $\{ e_b \} \mapsto e_d$, and a conflict between e_c and e_d. In Figure 1(b) we have $\{ e_a,e'_a \} \mapsto e_b$, $\{ e_a \} \mapsto e_x$ and $\{ e'_a \} \mapsto e_y$. In Figure 1(a) event e_c can happen after both e_a and e_b have occurred and if e_d has not yet appeared. e_d is enabled once e_b occurs. In Figure 1(b) e_b is enabled if either e_a or e'_a has appeared. □

Figure 1 Some example event structures.

Stochastic event structures are obtained from event structures by decorating events and bundles with random variables. To specify the relative delay between causally dependent

[1] In order to support disruption (denoted [>) in LOTOS the symmetric conflict can be considered being constructed from an asymmetric conflict relation. It is shown in [9, Chapter 8] how to deal with this construct in the stochastic case; for simplicity we consider # in this paper.

events a random variable is associated to a bundle, and in order to facilitate the specification of timing constraints on events that have no bundle pointing to them (i.e., the initial events), a random variable is also associated to an event. Though it seems sufficient to only have random variables for initial events, synchronization of events makes it necessary to allow for equipping all events with random variables, including the non-initial ones (see [2, 9]).

We assume mappings $\mathcal{A}$ and $\mathcal{R}$ to associate a random variable to events and bundles, respectively. A bundle $X \mapsto e$ with $\mathcal{R}((X,e)) = U$ is denoted by $X \overset{U}{\mapsto} e$; its interpretation is that if an event in X has happened at a certain time t, then the time of enabling of e is determined by $t+U$. E.g., the interpretation of $\{ e_a \} \overset{U}{\mapsto} e_b$ is that if e_a has happened at time t_a then the time at which e_b is enabled is determined by t_a+U. If more than one bundle points to an event the following interpretation is chosen. E.g., let $\{ e_a \} \overset{U}{\mapsto} e_c$ and $\{ e_b \} \overset{V}{\mapsto} e_c$. Now, if e_a happens at t_a and e_b at t_b then the time of enabling of e_c is determined by the random variable $max(t_a+U,t_b+V)$. This corresponds to the principle that an event can happen once all timing constraints on it have been met.

The interpretation of event e with $\mathcal{A}(e) = U$ is that e can happen at any t from the beginning of the system—usually assumed to be time 0—where t is a realization of U. E.g., in case $\{ e_a \} \overset{U}{\mapsto} e_b$ with $\mathcal{A}(e_b) = V$ we have that the random variable $max(V,t_a+U)$ determines the time of enabling of e_b given that e_a happens at time t_a.

So, the random variables associated to bundles specify the relative delay between causally dependent events, while the event random variables specify the absolute delay of events (i.e., the delay relative to the start of the system).

In the sequel we assume that RV is a class of random variables that is closed under *max*, i.e., for $U,V \in \mathsf{RV}$ we have $max(U,V) \in \mathsf{RV}$, and that has a unit element, $\mathbf{I}$ say, for *max*, i.e., $max(U,\mathbf{I}) = max(\mathbf{I},U) = U$, for all $U \in \mathsf{RV}$. Later on we will justify these requirements on RV.

3. Definition. (*Stochastic event structure*)

A *stochastic* event structure Σ over RV is a triple $\langle \mathcal{E},\mathcal{A},\mathcal{R} \rangle$ with $\mathcal{E}$ a bundle event structure $(E,\#,\mapsto,l)$, and $\mathcal{A} : E \longrightarrow \mathsf{RV}$ and $\mathcal{R} : \mapsto \longrightarrow \mathsf{RV}$, associating a random variable of class RV to events and bundles, respectively. □

For depicting stochastic event structures we use the same conventions as for event structures; the random variable associated with a bundle (event) is depicted near to the bundle (event). Random variables equal to $\mathbf{I}$ are usually omitted.

4. Definition. (*Stochastic determinism*)

A stochastic event structure $\langle \mathcal{E},\mathcal{A},\mathcal{R} \rangle$ is said to be *stochastically deterministic* (s-deterministic) if (i) $\mathcal{R}$ is injective, (ii) $ran(\mathcal{A}) \cap ran(\mathcal{R}) = \varnothing$, and

$$\text{(iii)}\ \forall e,e' \in E : (\mathcal{A}(e) \neq \mathbf{I} \ \wedge\ \mathcal{A}(e') \neq \mathbf{I} \ \wedge\ \mathcal{A}(e) = \mathcal{A}(e')) \ \Rightarrow\ e = e'$$

□

Here, for function $f : A \longrightarrow B$ let $ran(f)$ denote $\{ f(a) \mid a \in A \}$. The constraints ensure that a random variable (apart from $\mathbf{I}$) can be assigned to a single event or bundle only.

Since stochastic variants of Petri nets are widely used in the performance community we conclude this section by briefly addressing the relationship between stochastic event structures and Petri nets. Event structures were originally intended to better understand the (partial-order) behaviour of Petri nets. Events correspond to transitions (the main difference being that transitions can fire more than once, while an event can occur at most once) and bundles correspond—roughly speaking—to flow relations. The random variables associated to events and bundles can be interpreted as being attached to places. The interpretation of place p with random variable U is that when a token arrives in p at time t then this token is available for consumption by one of p's output transitions after a delay (since t) determined by U. It thus appears that stochastic event structures resemble a stochastic generalization of timed-place Petri nets by Sifakis [20]. Figure 2 illustrates this for some small examples.

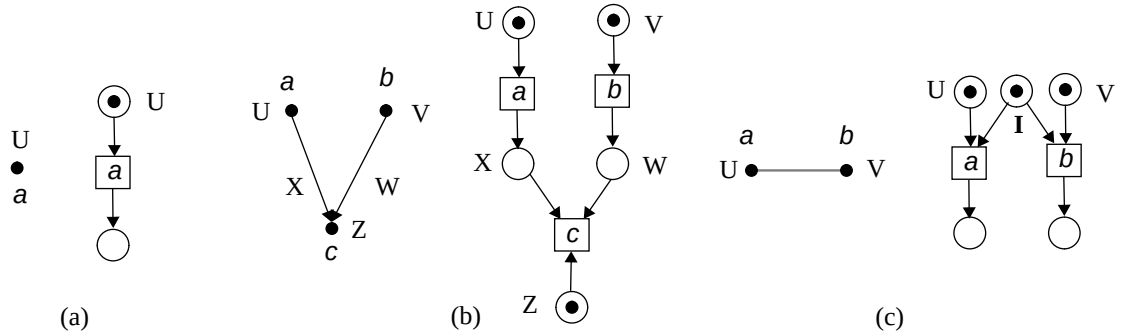

Figure 2 Stochastic event structures versus stochastic timed-place Petri nets.

3 Discrete-event simulation

In this section we introduce the basic notions of discrete-event simulation. In particular we will focus our attention on a mathematical framework for the study of discrete-event systems, the so-called generalized semi-Markov processes (GSMPs). While doing so, we restrict our attention to those notions that are relevant and sufficient for our purposes; thorough treatments of discrete-event simulation and GSMPs are, for instance, given by Shedler [19] and Glynn [3]. The introduction in Nicola *et. al* [13] is more intuitive.

A discrete-event system consists of a (countable) *output state space* S and a finite set of *stochastic events* (s-events) Ev. Usually these s-events are simply called events, but in order to distinguish between these events and the events occurring in event structures, we refer to them as s-events. Let $Ev = \{ \xi_1, \ldots, \xi_M \}$ be the set of s-events. For output state $s \in S$, $Ev(s)$ is the set of scheduled s-events, i.e., $Ev() : S \longrightarrow \mathcal{P}(Ev)$. $Ev(s)$ is the set of possible s-events that can happen in output state s. If $\xi_i \in Ev(s)$ then ξ_i is said to be *active* in state s, otherwise it is called *inactive*. The system moves from s to

s' by the execution of a single s-event $\xi \in Ev(s)$. $Ev(s)$ is thus the set of s-events that can trigger a state transition outgoing from s. (In general, the next state of a GSMP is probabilistically determined and may depend on the entire history of the system; for the sake of convenience we do not consider this possibility here). What is considered to be state and s-event depends heavily on the application at hand and the amount of detail that one wants to consider, see e.g., the examples in [19].

Each s-event ξ_i is accompanied by a *clock* c_i that indicates the amount of time to elapse until ξ_i can happen. That is to say, c_i indicates the "remaining lifetime" of ξ_i. For $\xi_i \notin Ev(s)$ we have $c_i = \infty$. An *internal state* of the discrete-event system is determined by its output state s, the clock values of the s-events in $Ev(s)$, and the current time t since the start of the system. Let c be a vector of clock values, for each s-event ξ_i a clock value c_i. The set of possible clock vectors of s, denoted $C(s)$, has to satisfy

$$\{\, (c_1, \ldots, c_M) \mid \forall\, i : (c_i = \infty \Leftrightarrow \xi_i \notin Ev(s)) \ \wedge\ (\forall\, j : c_i \neq \infty \ \wedge\ c_j \neq \infty \ \Rightarrow\ c_i \neq c_j) \,\}$$

The conditions ensure that (i) the clock of s-event ξ_i has the value ∞ iff ξ_i is inactive (in s), and (ii) that active s-events have different clock values. The internal state space of a GSMP is now given by

$$\{\, (s,c,t) \mid s \in S \ \wedge\ c \in C(s) \ \wedge\ t \in \mathbb{R}^+ \,\}$$

where t denotes the time at which the GSMP is considered, assuming that the system starts at time 0.

The value of s is changed explicitly by the state transitions of the system, i.e., by executing one of its active s-events. The value of each clock decreases implicitly as time advances. In this paper we assume that all clocks decrease their value at the same rate, although in the general theory of GSMPs different clock rates can be dealt with. Notice that the value of s does remain the same inbetween state transitions. Given that the discrete-event system is in state (s,c,t) it moves to (s',c',t') with $s \neq s'$ by the execution of the active s-event $\xi^* = \xi_j \in Ev(s)$ with minimal clock value, i.e., $c_j = \mathit{Min}\{\, c_i \mid \xi_i \in Ev(s) \,\}$. According to constraint (ii) of above this s-event is always uniquely defined. t' equals $t + c_j$. The clock values c' are now determined as follows:

$$c'_i \triangleq \begin{cases} c_i - c_j & \text{if } \xi_i \in Ev(s') \cap (Ev(s) - \{\, \xi^* \,\}) \\ F_i(\cdot) & \text{if } \xi_i \in Ev(s') - (Ev(s) - \{\, \xi^* \,\}) \\ \infty & \text{otherwise} \end{cases}$$

For each active s-event ξ_i in s that remains to be active in s' the clock is equal to c_i, the value of the clock at the previous transition instant, minus the elapsed time c_j. Notice that for these s-events the clocks remain running and are not reset. For each newly active s-event ξ_i the clock value is determined by a *clock-setting distribution function* F_i. (In general, this distribution function may depend on the entire history of the GSMP up to s' and on ξ^*, the trigger s-event). This means that a sample is taken from a stochastic variable with distribution F_i to generate the value c_i. Finally, for all other s-events the clock is set to ∞. Notice that this includes the clocks of active s-events in s that have become inactive in s'.

The reader should keep in mind that transitions from (s,c,t) to (s',c',t') with $s' = s$, $c' \neq c$ and $t' \neq t$ are not considered since time flows implicitly. The only (relevant) transitions a GSMP can take are for $s' \neq s$.

Since the states s of the discrete-event system do only change at transition instants it suffices to consider the state sequence

$$(s^0,c^0,t^0),(s^1,c^1,t^1),\ldots,(s^n,c^n,t^n)$$

where (s^i,c^i,t^i) denotes the internal state of the GSMP at the i-th transition ($i \geqslant 0$), where t^i denotes the time at which the i-th transition takes place ($t^0 = 0$). For our purposes it suffices to assume that there is a unique initial state. The next internal state $(s^{i+1},c^{i+1},t^{i+1})$ may depend on the complete history of the system $(s^0,c^0,t^0),\ldots,(s^i,c^i,t^i)$; not only on the current state (as in Markov chains).

The GSMPs considered in this paper are *time-homogeneous*, since the clock-setting distribution functions do only depend on the current state s, and not on the entire history of the system at hand, and since we do not consider state transition probabilities (and so, these probabilities do not depend on the history of the system). This entails that the internal state sequence $(s^0,c^0,t^0),\ldots,(s^n,c^n,t^n)$ is a time-homogeneous semi-Markov chain [3]. (If, in addition, all clock-setting distribution functions are exponential, then the internal state sequence is a continuous-time Markov chain.)

In the second part of this section we briefly address how we can calculate, for instance, the likelihood that a particular state sequence (or, sample path) is generated. Let $T^i = t^{i+1} - t^i$ be the time between the i-th and $(i+1)$-th transition and let W^i_j be the stochastic variable that determines the remaining lifetime of s-event ξ_j in internal state (s^i,c^i,t^i) with $\xi_j \in Ev(s^i)$. Define for $\xi_j \in Ev(s^{i+1}) \cap (Ev(s^i) - \xi^*)$

$$F^{i+1}_j(x) \triangleq Pr\{\, W^{i+1}_j \leqslant x \mid W^i_j \geqslant T^i \,\}$$

and let $\overline{F}^{i+1}_j = 1 - F^{i+1}_j$, the complementary distribution function of W^{i+1}_j. The superscript $i+1$ indicates the dependency of F_j on the history of the system $(s^0,c^0,t^0),\ldots,(s^i,c^i,t^i)$. Let f^i_j denote the probability density function of F^i_j. The following lemma stems from [13]:

5. Lemma. For all $\xi_j \in Ev(s^{i+1}) \cap (Ev(s^i) - \xi^*)$ we have:

1. $\overline{F}^{i+1}_j(x) = \overline{F}^i_j(x + T^i)/\overline{F}^i_j(T^i)$ and
2. $f^i_j(x) = f^{i+1}_j(x + T^i)/\overline{F}^i_j(T^i)$.

PROOF. We only consider the first part of the lemma.

$$\begin{aligned} & F^{i+1}_j(x) \\ = \;& \{ \text{ definition of } F_j \} \\ & Pr\{\, W^{i+1}_j \leqslant x \mid W^i_j \geqslant T^i \,\} \\ = \;& \{\, W^i = W^{i+1} + T^i \,\} \\ & Pr\{\, W^i_j \leqslant x + T^i \mid W^i_j \geqslant T^i \,\} \end{aligned}$$

$=$ { definition conditional probability; calculus }
$(Pr\{ W_j^i \leqslant x + T^i \} - Pr\{ W_j^i \leqslant T^i \})/Pr\{ W_j^i \geqslant T^i \}$
$=$ { calculus }
$(1 - Pr\{ W_j^i \geqslant x + T^i \} - (1 - Pr\{ W_j^i \geqslant T^i \}))/Pr\{ W_j^i \geqslant T^i \}$
$=$ { calculus }
$1 - Pr\{ W_j^i \geqslant x + T^i \}/Pr\{ W_j^i \geqslant T^i \}$
$=$ { definition of $\overline{F}_j$; $\xi_j \in Ev(s^{i+1}) \cap (Ev(s^i) - \xi^*)$ }
$1 - \overline{F}_j^i(x + T^i)/\overline{F}_j^i(T^i)$. □

Notice that this lemma is not needed to determine the clock values c^{i+1}. Instead, this lemma is useful in characterizing the probability density of a sample path of the GSMP, see below.

The probability density $P_{i,i+1}(\xi_j)$ of the discrete-event system going from internal state (s^i,c^i,t^i) to $(s^{i+1},c^{i+1},t^{i+1})$ by executing ξ_j is given by:

$$P_{i,i+1}(\xi_j) \triangleq f_j^i(T^i) \cdot \prod_{\xi_k \in Ev(s^i) - \{ \xi_j \}} (1 - F_k^i(T^i))$$

$P_{i,i+1}(\xi_j)$ is the probability that c_j has the minimum clock value of all active s-events in s^i. The likelihood $P_{0,n}$ of a sample path $(s^0,c^0,t^0),\ldots,(s^n,c^n,t^n)$ now equals $\prod_{i=0}^{n-1} P_{i,i+1}$.

4 Event structures and GSMPs

This section describes a recipe for generating a discrete-event system (in particular, a GSMP) starting from an s-deterministic stochastic event structure. This enables the generation of sample paths for this class of stochastic event structures, and constitutes the starting-point for performing simulation runs. If, in addition, the resulting GSMP satisfies certain restrictions such that it reduces to a (semi-) Markov chain (see previous section) standard analytical techniques for assessing such models can be applied.

How do we obtain a GSMP from an (s-deterministic) stochastic event structure? The basic idea is to let stochastic events (s-events) correspond to *bundles* and to let output states be stochastic event structures. Consider, for example, $X \mapsto e$. This bundle is 'activated' as soon as an event e_i in the set X appears. Suppose that e_i happens at time t_i. At that moment we schedule a new s-event that models the satisfaction of $X \mapsto e$, and initialize its clock value with a sample of $\mathcal{R}((X,e))$, the random variable associated with $X \mapsto e$. Random variables associated with bundles thus act like clock-setting distributions of the GSMP. Once the s-event happens, bundle $X \mapsto e$ is satisfied, and is discarded from the event structure at hand. (If it happens to be the only bundle pointing to e, then in addition, e is recorded to have happened.) In this way, stochastic event structures evolve in time by executing bundles (and events).

We now describe the recipe for obtaining a GSMP from a stochastic event structure by discussing the components of the GSMP—output states, initial state, stochastic events, and state transition relation—one by one.

Output states

An output state consists of three components: an event structure $\mathcal{E} = (E,\#,\mapsto,l)$, a function $\mathcal{R}$ that associates to each bundle in $\mathcal{E}$ a random variable, and a set $\mathcal{H} \subseteq E$ that records the events that have happened so far. Remark that we do not have a function associating a random variable to events in E.

Initial state

For s-deterministic stochastic event structure $\Sigma = \langle (E,\#,\mapsto,l),\mathcal{A},\mathcal{R}\rangle$ the initial state $s^0 = (\mathcal{E}^0,\mathcal{R}^0,\mathcal{H}^0)$ of the discrete-event system is defined as follows: $\mathcal{E}^0 = (E^0,\#^0,\mapsto^0,l^0)$ with $E^0 = E \cup \{\alpha\}$, $\#^0 = \#$, $\mapsto^0 = \mapsto \cup \{(\{\alpha\},e) \mid e \in E \wedge \mathcal{A}(e) \neq \mathbf{I}\}$, $l^0 = l \cup \{(\alpha,\text{start})\}$, $\mathcal{R}^0 = \mathcal{R} \cup \{((\{\alpha\},e),\mathcal{A}(e)) \mid e \in E \wedge \mathcal{A}(e) \neq \mathbf{I}\}$, and $\mathcal{H}^0 = \{\alpha\}$. Stated in words, a fictitious *start event* α is introduced modelling the start of the system, from which a bundle is pointing to any event e in E with a stochastic variable different from $\mathbf{I}$. (In our case this includes all initial events in Σ, since these events have been assigned a random variable different from $\mathbf{I}$ at language level.) $\mathcal{A}(e)$, the random variable of e in Σ, is associated with this new bundle $\{\alpha\} \mapsto e$. As an effect, all (non-$\mathbf{I}$) event random variables are turned into bundle random variables. We assume that the system is initially just started, so $\mathcal{H}^0 = \{\alpha\}$.

Since Σ is s-deterministic it follows immediately that $\mathcal{R}^0$ is injective.

6. Example. Consider Σ as depicted in Figure 3(a). The corresponding initial state of its discrete-event system is depicted in Figure 3(b). (We adopt the convention that happened events are indicated as open dots, whereas events that have not yet happened are drawn as closed dots, like before.) □

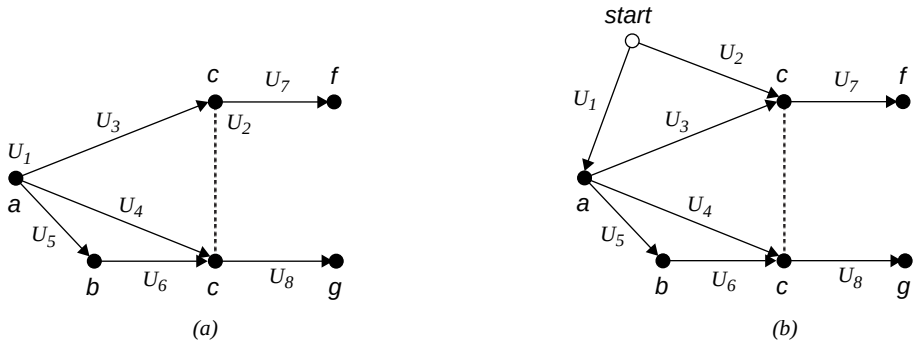

Figure 3 A (a) stochastic event structure Σ and (b) its initial GSMP state $(\mathcal{E}^0,\mathcal{R}^0,\mathcal{H}^0)$.

Stochastic events

The s-events of the discrete-event system are bundles (rather than events!). Since the set of s-events needs to be finite we require the stochastic event structure to have a finite set of bundles. (This means that for recursive processes we use a finite approximation of

the fixpoint semantics, in case this is infinite; see Section 6.)

For $s = (\mathcal{E},\mathcal{R},\mathcal{H})$ the set of activated s-events $Ev(s)$ is defined as follows:

$$Ev(s) \triangleq \{\, (X,e) \in \mapsto \mid X \cap \mathcal{H} \neq \varnothing \ \wedge \ \neg\, (\exists\, e' \in \mathcal{H} : e \# e') \,\}$$

So, bundle (X,e) is 'activated' in output state s, if some event in X has appeared, and if it is not pointing to event e which is permanently disabled (due to some other conflicting event that has already appeared). Remark that we could equally well omit the last conjunct, allowing all bundles to expire. It is, however, not necessary to consider those bundles that point to events that can never happen anyway.

In case a new s-event $\xi_i = X \mapsto e$ becomes activated in output state s then c_i is assigned a randomly drawn realization (i.e., a sample) of $\mathcal{R}((X,e))$. In this way, the distribution functions of the random variables associated to bundles in s act like clock-setting distribution functions. Notice that clocks of activated s-events can never be identical in our case, since component $\mathcal{R}$ of s is injective. This entails that all continuous random variables in s are unique and statistically independent, so the probability that two clocks have the same value is 0.

7. EXAMPLE. Let s be the output state of Figure 3(b) and let s-event ξ_i correspond to the bundle with distribution U_i. Then $Ev(s) = \{\, \xi_1,\xi_2 \,\}$, with $c_1 = F_{U_1}(\cdot)$ and $c_2 = F_{U_2}(\cdot)$, i.e., the clocks of ξ_1 and ξ_2 are assigned a random sample of U_1 and U_2, respectively. The clocks of all other s-events equal ∞. (In the sequel we omit mentioning clocks equal to ∞.) □

State transitions

Suppose the discrete-event system is in state (s,c,t) and moves to (s',c',t') by executing s-event $\xi^* \in Ev(s)$. Let $s = (\mathcal{E},\mathcal{R},\mathcal{H})$ and $s' = (\mathcal{E}',\mathcal{R}',\mathcal{H}')$. Assume that ξ^* corresponds to bundle $X \mapsto e$ in $\mathcal{E}$. Then $\mathcal{E}'$ is determined as follows: $E' = E$, $\#' = \#$, $l' = l$ and $\mapsto' = \mapsto \setminus \{\, (X,e) \,\}$. Bundle $X \mapsto e$ is eliminated since this bundle 'appeared'. The random variables associated with the retained bundles are unchanged, i.e., $\mathcal{R}' = \mathcal{R} \restriction \mapsto'$. Finally, if $X \mapsto e$ is the only bundle pointing to e then e is added to the set of happened events, since all bundles pointed to it have disappeared and their corresponding timings have elapsed. So, we obtain

$$\mathcal{H}' \triangleq \mathcal{H} \cup \{\, e \mid \forall\, Y : Y \mapsto e \ \Rightarrow \ Y = X \,\}$$

This completely defines the next state s'. The clock values c' in the next state are determined according to the recipe in Section 3. The time t' equals t plus the clock value of the s-event ξ^* that triggered the transition from s to s'. It is not difficult to check that the above described discrete-event system for stochastic event structure Σ is a time-homogeneous GSMP.

The complete simulation algorithm for stochastic event structure $\langle \mathcal{E},\mathcal{A},\mathcal{R} \rangle$ with $\mathcal{E} = (E,\#,\mapsto,l)$ is presented in Table 1. For random variable U with distribution F_U let $F_U(\cdot)$ denote a sample of U; output state $s^i = (\mathcal{E}^i,\mathcal{R}^i,\mathcal{H}^i)$, for $i \geqslant 0$. It is assumed that the function $Ev()$ is given.

$\mathcal{E}^0 := (E \cup \{\alpha\}, \#, \mapsto \cup \{(\{\alpha\}, e) \mid e \in E \wedge \mathcal{A}(e) \neq \mathbf{I}\}, l \cup \{(\alpha, \text{start})\})$;
$\mathcal{R}^0 := \mathcal{R} \cup \{((\{\alpha\}, e), \mathcal{A}(e)) \mid e \in E \wedge \mathcal{A}(e) \neq \mathbf{I}\}$;
$\mathcal{H}^0 := \{\alpha\}$;
for all $X_k \mapsto e_k \in \mapsto^0$
do if $X_k \mapsto e_k \in Ev(s^0) \longrightarrow c_k^0 := F_{\mathcal{R}^0((X_k, e_k))}(\cdot)$
[] $X_k \mapsto e_k \notin Ev(s^0) \longrightarrow c_k^0 := \infty$
fi od;
$t^0 := 0$;
$i := 0$;
while $Ev(s^i) \neq \varnothing$
do choose $X_j \mapsto e_j \in Ev(s^i)$ such that $c_j^i = Min\{c_k^i \mid U_k \in Ev(s^i)\}$;
$\mathcal{E}^{i+1} := (E^i, \#^i, \mapsto^i \setminus \{(X_j, e_j)\}, l^i)$;
$\mathcal{R}^{i+1} := \mathcal{R}^i \restriction \mapsto^{i+1}$;
$\mathcal{H}^{i+1} := \mathcal{H}^i \cup \{e \mid \forall Y : Y \mapsto^i e \Rightarrow X_j = Y)\}$;
for all $X_k \mapsto e_k \in \mapsto^{i+1}$
do case
[] $X_k \mapsto e_k \in Ev(s^{i+1}) \cap (Ev(s^i) - \{(X_j, e_j)\}) \longrightarrow c_k^{i+1} := c_k^i - c_j^i$
[] $X_k \mapsto e_k \in Ev(s^{i+1}) - (Ev(s^i) - \{(X_j, e_j)\}) \longrightarrow c_k^{i+1} := F_{\mathcal{R}^{i+1}((X_k, e_k))}(\cdot)$
[] $X_k \mapsto e_k \notin Ev(s^{i+1}) \longrightarrow c_k^{i+1} := \infty$
esac od;
$t^{i+1} := t^i + c_j^i$;
$i := i + 1$;
od

Table 1 Discrete-event simulation algorithm for stochastic event structures.

8. EXAMPLE. Consider our running example of this section. Figure 4 illustrates a possible state sequence, starting from (0) s^0 to (5) s^5. E.g., consider s^0 with $t^0 = 0$. Recall that its active s-events are ξ_1 and ξ_2, with clocks $c_1^0 = F_{U_1}(\cdot)$ and $c_2^0 = F_{U_2}(\cdot)$. Assume $c_1^0 < c_2^0$. Then ξ_1 will trigger a state transition from s^0 to s^1, with $c_2^1 = c_2^0 - c_1^0 = F_{U_2}(\cdot) - F_{U_1}(\cdot)$ and $t^1 = t^0 + c_1^0 = F_{U_1}(\cdot)$. The new s-events ξ_3, ξ_4 and ξ_5 are scheduled in s^1 with $c_3^1 = F_{U_3}(\cdot)$, $c_4^1 = F_{U_4}(\cdot)$ and $c_5^1 = F_{U_5}(\cdot)$. Assuming that c_2^1 is the minimum clock value in s^1 the system moves to s^2 with $t^2 = t^1 + c_2^1 = F_{U_2}(\cdot)$. In s^2 no new s-events are scheduled, and the clocks of ξ_3, ξ_4 and ξ_5 remain running. Notice that in s^2 s-event ξ_2 has appeared, but event e_c has not happened, because ξ_3 has not yet appeared. The other steps are left to the reader. It should be noted that $\xi_4 \notin Ev(s^3)$, since this bundle points to e_c, which is disabled by a happened event. □

Remark that it is not difficult to adapt the above definitions such that events that have happened finally disappear from the evolving event structure: remove events from E once they are happened and do not belong to some bundle set X, $X \mapsto e'$, and introduce an empty bundle $\varnothing \mapsto e'$ to all events e' for which $e' \# e$, once e happens. These empty bundles reflect the fact that e' is disabled (by e).

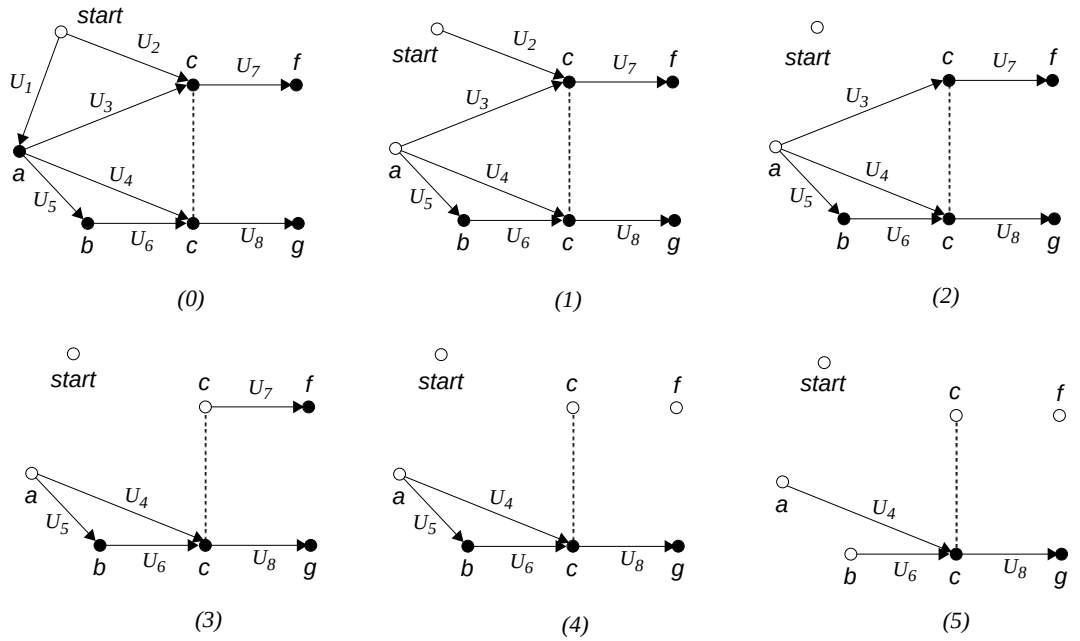

Figure 4 Possible state sequence $s^0, s^1, \ldots, s^5$ for GSMP of Σ.

5 Recursion and regenerative simulation

The mapping of stochastic event structures onto GSMPs presented above enables the generation of simulation runs (often called sample paths). As a next step, of course, simulation data should be gathered and analyzed. In a nutshell this works in general as follows. Suppose we are interested in the (unknown) value β of the desired performance measure. Since β is obtained as a function of stochastic variables, β is a realization (or sample) of a stochastic variable, U say. For a simulation run random variable U_i—called an observation—is defined that is supposed to 'represent' U. Given a set of observations $U_1, \ldots, U_n$, an *estimator* $\hat{U} = f(U_1, \ldots, U_n)$ is defined. Given the samples $\beta_1, \ldots, \beta_n$ of $U_1, \ldots, U_n$ this estimator obtains an estimate $\hat{\beta}$ of β. In principle different estimators can be defined, but not all functions are equally suitable. Interesting estimators are, e.g., those that are unbiased, i.e., for which the expected value of $\hat{U}$ equals $E[U]$, or that are strongly consistent, i.e., $\hat{\beta}$ reaches β for n approaching ∞.

Now one obtains an estimate $\hat{\beta}$ of the performance result from the samples of the stochastic variables $U_1, \ldots, U_n$. These samples are obtained from simulation runs of the system. Finally, the reliability of the obtained estimate is assessed. This is usually done by constructing *confidence intervals*. A confidence interval is an interval around the estimate in which the actual value $E[U]$ lies, with a certain—and preferably high—probability. There are different ways to obtain the right estimators and corresponding (narrow) confidence intervals, see [21].

Usually, different simulation runs have to be carried out in order to obtain results of a sufficient confidence level. However, for a specific class of models, the so-called regenera-

tive models, a single simulation run may be sufficient to obtain confident statistics. The evolution of a *regenerative stochastic process* can be divided into cycles, where between any two successive regeneration points, the behaviour of the process is a probabilistic replica of the process in any other cycle.

Under certain conditions (cf. [19]) a GSMP can be considered as a regenerative stochastic process. In particular this is the case when the GSMP is (1) irreducible, i.e., for each s,$s' \in S$ state s' should be 'reachable' from s, (2) the state space S and s-event set Ev are finite, (3) all clock-setting distribution functions have finite means and a density function that is continuous and positive on [0,∞), and (4) there exists a pair of states s,$s' \in S$ such that s can go to s' for which $Ev(s) \cap Ev(s') = \varnothing$, and $Ev(s')$ and the clock-setting distributions in s' are independent of the trigger event for entering s'. Notice that (4) requires that only new s-events are active in state s'. Under these conditions the GSMP probabilistically restarts when going from s to s', and the expected time between regeneration points is finite.

Given that a GSMP is regenerative there are standard ways in which to obtain (strongly consistent) estimators, and (asymptotic) confidence intervals by observing a finite part of a *single* simulation run of the GSMP, see for details [19, Chapter 3]. An interesting question therefore is whether stochastic event structures can give rise to regenerative GSMPs. As will be indicated in Section 6 recursive processes give rise to infinite stochastic event structures. Under a regularity assumption, which applies e.g., for tail recursion, such infinite event structures can be finitely represented and can considered to be 'regenerative', since they exhibit repetitive behaviour. For example, the event structure of Figure 7(c) satisfies such regularity principle: after the occurrence of an event labelled a the event structure repeats its behaviour. For such processes regenerative simulation is considered to be a useful technique.

6 A non-Markovian stochastic process algebra

In this paper we use a stochastic process algebra in which the delay of actions is specified by continuous stochastic variables:

9. Definition. (*Syntax of stochastic process algebra*)

$$B ::= \mathbf{0} \mid a_U \,;\, B \mid B + B \mid B \,||_G\, B \mid B[H] \mid B \setminus G \mid P$$

□

We assume a given set of observable actions Act and an additional *invisible action* τ; $\tau \notin \mathsf{Act}$. $\mathbf{0}$ denotes *inaction*, i.e., a process that cannot perform any action. $a_U \,;\, B$ denotes the *action-prefix* of $a \in \mathsf{Act} \cup \{\tau\}$ and B where the time from which a is enabled is determined by a continuous random variable U, $U \in \mathsf{RV}$ with $U \neq \mathbf{I}$. (This constraint avoids to have pure non-deterministic choices like $a_I + a_I$.) All actions are assumed to have zero duration; i.e., they occur instantaneously. The *choice* between B_1 and B_2 is denoted $B_1 + B_2$. $B_1 \,||_G\, B_2$ denotes *parallel composition* where actions in G

($G \subseteq \mathsf{Act}$) are synchronization actions; B_1 and B_2 can perform actions that are not in G independently of each other. $|||$ abbreviates $||_{\varnothing}$, i.e., parallel composition without synchronization. $B[H]$ denotes the *relabelling* of B according to H where $H(\tau) = \tau$, and $H(a) \neq \tau$ for $a \in \mathsf{Act}$. $B \setminus G$ denotes *hiding*; in $B \setminus G$ all actions in G ($G \subseteq \mathsf{Act}$) are turned into invisible actions. Finally, P denotes a *process instantiation* where a behaviour is considered in the context of a set of process definitions of the form $P := B$ where B possibly contains occurrences of P.

The precedences of the composition operators are, in decreasing binding order: ; , +, $||$, $\setminus$ and $[\,]$. Trailing **0**s are usually omitted.

The synchronization principle is that an action can only occur when all participants are ready to engage in it. Thus, for instance, in $a_W \,;\, b_U \,||_{\{a,b\}}\, a_Z \,;\, b_V$, the enabling time of a is determined by $max(W,Z)$ assuming that the system starts at time 0. t_a+$max(U,V)$ determines the enabling time of b given that a occurs at time t_a.

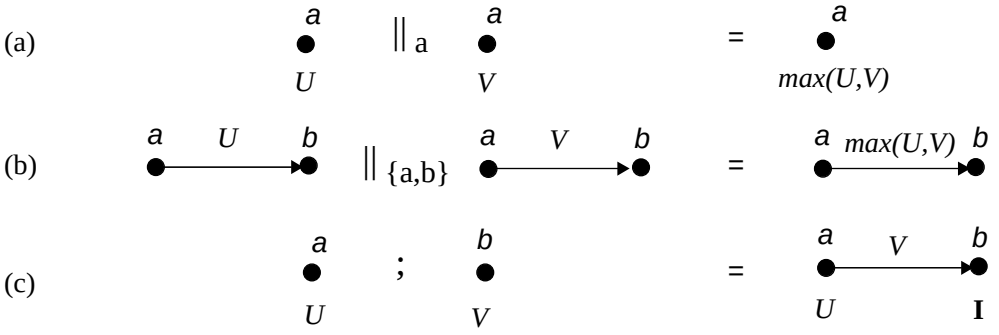

Figure 5 Examples of composing stochastic event structures.

The semantics of our process algebra is defined using stochastic event structures. A mapping $[\![\ ]\!]$ is defined that maps each expression B onto a stochastic event structure $[\![B]\!]$ in a compositional way. This mapping is completely described in [2, 9]. Let us start by considering some simple examples (cf. Figure 5). According to our synchronization principle the resulting random variable of e_a in Figure 5(a) equals $max(U,V)$. A similar reasoning applies to (b) (where, for simplicity, irrelevant random variables are omitted). Finally, in (c) the main issue is what the random variable, W say, associated with e_b will be. Intuitively, this random variable should not impose an additional constraint on the timing of e_b. Therefore, we let $W = \mathbf{I}$. Then e_b is enabled after the occurrence of e_a (at time t_a) at time $max(t_a{+}V,\mathbf{I}) = t_a{+}V$, what one would expect.

This motivates that we require the class RV of random variables to be closed under max and to have a unit element $\mathbf{I}$ for max, as presented on page 5. (Notice that for statistically independent random variables U and V the distribution function of $max(U,V)$ corresponds to the product of their distribution functions, i.e., $F_U \cdot F_V$.)

10. Example. For the purpose of this paper it suffices to present the semantics by example. The expressions corresponding to Figure 6 are as follows: (a) $(a_U \,;\, d_V + b_W \,;\, e_R) \,|||\, c_Z$, (b) $a_U \,;\, b_V \,||_b\, b_W$, and (c) $a_U \,;\, c_V \,||_c\, b_W \,;\, c_Z$. □

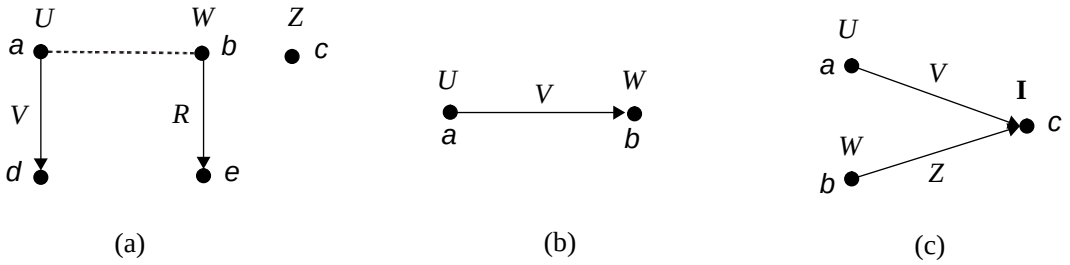

Figure 6 Some examples of noninterleaving semantics.

Finally we explain the semantics of process instantiation P, for $P := B$. $[\![P]\!]$ is defined in the following way by using standard fixed point theory [17]. A complete partial order (c.p.o.) $\trianglelefteq$ is defined on stochastic event structures with the empty event structure (i.e., $[\![\mathbf{0}]\!]$) as the least element $\perp$. Then for each definition $P := B$ a function $\mathcal{F}_B$ is defined that substitutes a stochastic event structure for each occurrence of P in B, interpreting all operators in B as operators on stochastic event structures. $\mathcal{F}_B$ is shown to be continuous, which means that $[\![P]\!]$ can be defined as the least upper bound (l.u.b.) of the chain (under $\trianglelefteq$) $\perp$, $\mathcal{F}_B(\perp)$, $\mathcal{F}_B(\mathcal{F}_B(\perp)), \ldots$. It is beyond the scope of this paper to treat all details here; [9, Chapter 10] treats the recursive case extensively.

11. EXAMPLE. As an example of a recursive process definition we consider

$$P := a_U \,;\, P \,|||\, b_V \quad .$$

The first approximation is $\perp$. The second and third approximations $\mathcal{F}_B(\perp)$ resp. $\mathcal{F}_B^2(\perp)$ are depicted in Figure 7(a) and (b), respectively. By repeated substitution we obtain the stochastic event structure of Figure 7(c). □

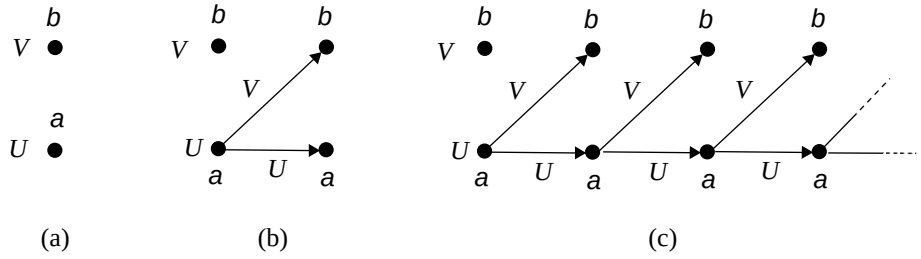

Figure 7 Example of semantics for a recursive process definition.

An interesting feature of $[\![\ ]\!]$ is that removing the parts concerning the random variables of events and bundles in the definition of $[\![\ ]\!]$ leads to the event structure semantics of ordinary LOTOS given in [12]; so, $[\![\ ]\!]$ is an *orthogonal extension* of the plain event structure semantics. That is, stochastic information is not affecting the other semantic components like causality and conflict.

7 Simulating behaviour expressions

In this section we show how the recipe of obtaining GSMPs can be applied to behaviour expressions. Recall that the recipe of Section 4 considers s-deterministic stochastic event structures. In order to avoid that several random variables occur more than once in an expression (and thus will lead to a non s-deterministic event structure) we start by decorating each random variable associated to an action-prefix with an arbitrary but unique occurrence identifier, denoted by a natural number. For instance, an expression like $a_U \,;\, b_V + a_U$ becomes $a_{U_1} \,;\, b_{V_2} + a_{U_3}$. To enforce that for recursive behaviours unique identifiers are generated we label each process occurrence with a globally unique id as well. Thus, $P := a_U \,;\, P$ becomes $P := a_{U_1} \,;\, P_2$. Each unfolding of P generates unique ids for random variables by prefixing with the id of the process instantiation. So, P contains random variables U_1,U_{21},U_{221}, ...

Unfortunately, when this procedure is applied to expression B this does not imply that $[\![B]\!]$ is s-deterministic. For instance, for $B = a \,;\, d_U \,||_d (b \,;\, d_V \,|||\, c \,;\, d_W)$ where irrelevant distributions are omitted we obtain two bundles in $[\![B]\!]$ with random variable U (see Figure 8). This is caused by the fact that there are two possible candidates with which d_U can synchronize—a form of pure nondeterminism—and this results in copying bundle $\{ e_a \} \overset{U}{\mapsto} e_d$.

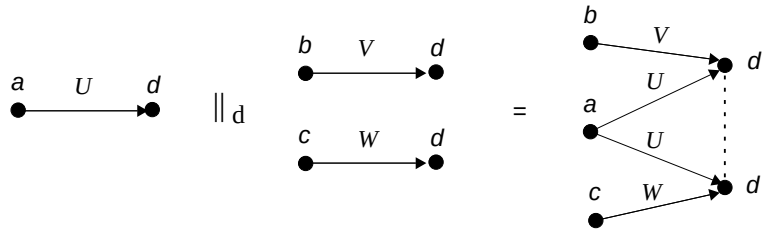

Figure 8 Copying of bundles (and random variables).

When we apply the recipe of Section 4 to $[\![B]\!]$ we obtain an output state (i.e., an event structure) in the GSMP with two 'activated' bundles equipped with the same random variable U. The use of this random variable as a clock-setting distribution function would now violate the clock constraint of GSMPs, see Section 3: we would obtain two s-events—one per bundle $\{ e_a \} \mapsto e_d$ —with the same clock value $F_U(\cdot)$ (different from ∞). A possible solution to avoid this form of pure nondeterminism (which is not a concept present in GSMPs) is to perform a renaming of random variables in $[\![B]\!]$, while keeping the same distributions of course, such that all random variables become unique and can be treated as statistical independent. An alternative to resolve this form of pure nondeterminism is to consider the process at hand in the context of an adversary, or scheduler. An adversary is an entity that schedules the next event (probably probabilistically) based on the past execution of the event structure so far. Such adversaries are, for instance, also used in [18].

8 Discussion and concluding remarks

The main contribution of this paper is an algorithm for obtaining generalized semi-Markov processes (GSMPs) from (s-deterministic) stochastic event structures, a causality-based semantical model suited for stochastic process algebras that include general distributions. In this way standard simulation techniques can be adopted to obtain performance statistics from (s-deterministic) stochastic event structures, and consequently, from a large class of expressions in our stochastic process algebra. To our knowledge this constitutes the first attempt to obtain stochastic simulation models from event structures and stochastic process algebras in a systematic way.

Apart from the fact that we allow general distributions our approach contrasts the interleaving approaches (such as [8, 4, 7, 1]) which obtain performance statistics either analytically or numerically by linking the semantical model to continuous-time Markov chains (CTMCs). Harrison and Strulo [6] use a stochastic process algebra to formally describe discrete-event simulation. In line with our work they do not restrict time delays to be specified by exponential random variables. They indicate (as we do in Section 7), however, that nondeterminism gives rise to invalid simulations. This is a serious problem in their approach since an interleaving semantics is used, i.e., parallelism is 'resolved' by nondeterminism. This type of 'artificial' nondeterminism is absent in our causality-based approach (and only pure nondeterminism causes some problems). Another important difference is that we obtain GSMPs from an arbitrary stochastic event structure, while Harrison and Strulo show that their formalism is powerful enough to describe the behaviour of GSMPs.

There are several paths along which we can extend our approach. Firstly, it would be interesting to identify a class of infinite stochastic event structures that can be finitely represented, since this would allow the use of regenerative simulation techniques (in a similar way as for stochastic Petri nets [5]). In order to support probabilistic branching at the language level it would be interesting to incorporate probabilities into our model. In particular, we plan to adjust our preliminary work on probabilistic extensions of non-stochastic event structures [10, 9] to the stochastic case, e.g., in accordance to the elegant proposal of Rettelbach [15] for the interleaving case. Concerning the mapping onto GSMPs we do not foresee serious problems, since state transitions in GSMPs are in general probabilistic. An alternative direction would be to investigate to what extent numerical techniques can be used to analyze stochastic event structures, for instance, by using discrete approximation of continuous distribution functions, see e.g., [11].

Acknowledgements The authors like to thank Victor Nicola for helpful discussions and suggestions. The work presented in this paper has been partially funded by C.N.R. - Progetto Bilaterale: Estensioni probabilistiche e temporali dell'algebra di processi LOTOS basate su strutture di eventi, per la specifica e analisi quantitative di sistemi distribuiti, C.N.R. - Progetto Coordinato: Strumenti per la specifica e verifica di proprieta' critiche di sistemi concorrenti e distribuiti, and by the EU as part of the ESPRIT BRA project 6021: Building Correct Reactive Systems (REACT).

References

[1] M. Bernardo and R. Gorrieri. Extended Markovian process algebra. In U. Montanari and V. Sassone, editors, *CONCUR'96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 315–330. Springer-Verlag, 1996.

[2] E. Brinksma and J-P. Katoen and R. Langerak and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, **38**(7):552–565, 1995.

[3] P.W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, **77**(1):14–23, 1989.

[4] N. Götz and U. Herzog and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In L. Donatiello and R. Nelson, editors, *Performance Evaluation of Computer and Communication Systems*, volume 729 of *Lecture Notes in Computer Science*, pages 121–146. Springer-Verlag, 1993.

[5] P.J. Haas and G.S. Shedler. Regenerative stochastic Petri nets. *Performance Evaluation*, **6**(3):189–204, 1986.

[6] P.G. Harrison and B. Strulo. Stochastic process algebra for discrete event simulation. In F. Baccelli, A. Jean-Marie and I. Mitrani, editors, *Quantitative Methods in Parallel Systems*, Springer, pages 18–37, 1995.

[7] H. Herrmanns and M. Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In U. Herzog and M. Rettelbach, editors, *Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling.* Technical Report **27**(4), Universität Erlangen-Nürnberg, pages 71–87, 1994.

[8] J. Hillston. *A Compositional Approach to Performance Modelling.* PhD thesis, University of Edinburgh, 1994. (also available as Technical Report CST-107-94).

[9] J-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures.* PhD thesis, University of Twente, CTIT Ph. D-thesis series No. 96-09, 1996.

[10] J-P. Katoen and R. Langerak and D. Latella. Modelling systems by probabilistic process algebra: An event structures approach. In R.L. Tenney, P.D. Amer, and M.Ü. Uyar, editors, *Formal Description Techniques VI*, volume C–22 of *IFIP Transactions*, pages 253–268. North-Holland, 1994.

[11] W. Kleinöder. *Stochastische Bewertung von Aufgabenstrukturen für hierarchische Mehrrechnersysteme.* PhD thesis, University of Erlangen-Nürnberg, 1982.

[12] R. Langerak. *Transformations and Semantics for LOTOS.* PhD thesis, University of Twente, 1992.

[13] V.F. Nicola and M.K. Nakayama and P. Heidelberger and A. Goyal. Fast simulation of highly dependable systems with general failure and repair processes. *IEEE Transactions on Computers*, **42**(12):1440–1452, 1993.

[14] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models–An Algorithmic Approach.* The Johns Hopkins University Press, 1981.

[15] M. Rettelbach. Probabilistic branching in Markovian process algebras. *The Computer Journal*, **38**(7):590–599, 1995.

[16] R.A. Sahner and K.S. Trivedi. Performance and reliability analysis using directed acyclic graphs. *IEEE Transactions on Software Engineering*, **13**(10):1105–1114, 1987.

[17] D.A. Schmidt. *Denotational Semantics: a methodology for language development.* Allyn and Bacon, 1986.

[18] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, **2**:250–273, 1995.
[19] G.S. Shedler. *Regenerative Stochastic Simulation.* Academic Press, Inc., 1993.
[20] J. Sifakis. Use of Petri nets for performance evaluation. In H. Beilner and E. Gelenbe, editors, *Measuring, Modelling and Evaluating Computer Systems.* North-Holland, pages 75–93, 1977.
[21] P.D. Welch. The statistical analysis of simulation results. In S.S. Lavenberg, editor, *The Performance Modelling Handbook.* Academic Press, pages 267–329, 1983.
[22] G. Winskel. An introduction to event structures. In J.W. de Bakker, W-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*, pages 364–397. Springer-Verlag, 1989.