Tutte le algebre insieme:

Concepts, Discussions and Relations of Stochastic Process Algebras with General Distributions

Mario Bravetti[1] and Pedro R. D'Argenio[2]*

[1] Dip. di Scienze dell'Informazione, Università di Bologna,
Mura Anteo Zamboni 7, 40127 Bologna, Italy
bravetti@cs.unibo.it

[2] CONICET – FaMAF, Universidad Nacional de Córdoba,
Ciudad Universitaria, 5000 Córdoba, Argentina
dargenio@famaf.unc.edu.ar

Abstract. We report on the state of the art in the formal specification and analysis of concurrent systems whose activity duration depends on general probability distributions. First of all the basic notions and results introduced in the literature are explained and, on this basis, a conceptual classification of the different approaches is presented. We observe that most of the approaches agree on the fact that the specification of systems with general distributions has a three level structure: the process algebra level, the level of symbolic semantics and the level of concrete semantics. Based on such observations, a new very expressive model is introduced for representing timed systems with general distributions. We show that many of the approaches in the literature can be mapped into this model establishing therefore a formal framework to compare these approaches.

1 Introduction

The research community has widely recognized the importance of time aspects in the specification and analysis of concurrent systems and communication protocols (see e.g. [1, 3, 19] and the references therein). There are fundamentally two reasons behind this recognition: first of all, the (correct) behavior of certain systems/protocols often depends on real-time aspects; second, expressing time duration of system activities makes it possible to estimate system performance. In this paper we report on the state of the art in the formal specification and analysis of concurrent systems whose activity durations are random variables.

The random duration of an activity is represented by probability distribution functions. For example Fig. 1 depicts the probability of the duration of an activity. This can be any time value between two and four time units.

An important special case of duration distribution is the exponential distribution. Due to its nice mathematical properties (the so called "memoryless" property), the problem of specifying systems which only make use of exponential

* Partially supported by the NWO visiting grant B-61-519

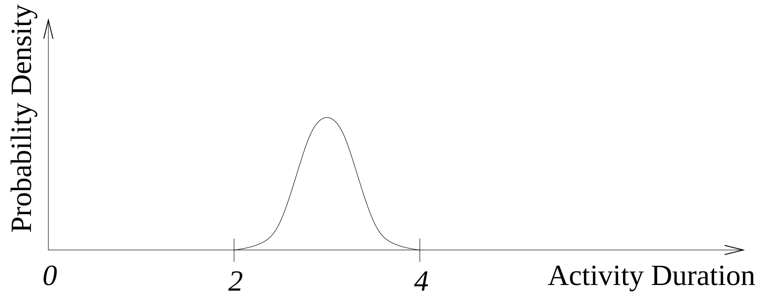


Fig. 1. Activity with a probabilistic duration

distributions is easier than the general case and has been successfully studied (see e.g. [20, 19, 3, 5]). The approach obtained in this restricted setting is often referred to as the "Markovian" approach, in that system behavior turns out to be expressible by simple continuous time Markov chains (CTMCs). In spite of its advantages, the Markovian approach imposes a considerable limitation on the modeling of time aspects. Therefore, it is important to address the general case in which probability distributions can be arbitrary.

In order to better understand the power of general distributions, let us look again at the example of Fig. 1. The distribution of Fig. 1 expresses both *time bounds* for the activity (i.e. it will certainly last between 2 and 4 time units) and a *probabilistic quantification* over the possible duration values for the activity. This shows that, compared to the Markovian case where time bounds are not expressible, general distributions make it possible to express both the system aspects typical of real time modeling (where time bounds are represented) and the system aspects typical of stochastic modeling (where a probabilistic quantification of time values is expressed) in an integrated fashion. In particular, given a system specification with general distributions, it is possible both to validate its real time properties via, e.g., model checking, and to evaluate its performance.

The problem of specifying and analyzing systems with general distributions has been studied, e.g., in [11, 9, 36, 17, 32, 6, 4]. This paper focuses on such a general problem and presents the main concepts and results obtained in the literature. More precisely we proceed as follows. First of all we explain basic notions about the formal specification and analysis of systems with general distributions and, on this basis, we present a conceptual classification of the different approaches in the literature.

A result of such a conceptual step is the observation that much of the approaches in the literature are based on the common idea that the specification of systems with general distributions has a three level structure: the process algebra level, the level of symbolic semantics (where system timed behavior is represented in a symbolic way by clocks), and the level of concrete semantics (where system timed behavior is represented explicitly by numerical timed transitions). Based on such an observation, we introduce a new model to represent timed systems with general distributions. We show that many of the approaches in the literature can be mapped into this model establishing therefore a formal framework to compare such approaches.

The paper is organized as follows. In Sect. 2 we present the basic concepts about the formal specification and analysis of concurrent systems with stochastic time on the basis of the several approaches in the literature. In Sect. 3 we introduce “prioritized stochastic automata (PSA)”: the common symbolic model for the representation of stochastic timed systems on which we will map such approaches. Sect. 4 introduces “probabilistic timed transition systems (PTTSs)”: the concrete timed model used to define the semantics of PSA. Actually we consider two kind of semantics based on so-called “residual” (see [11]) and “spent” (see [4]) clock lifetimes. Sect. 5 defines symbolic bisimulation equivalence (over PSA) and concrete bisimulation equivalence (over PTTSs). In Sect. 6 we provide embeddings of the stochastic models presented in [4, 11, 9] into PSA. In Sect. 7, semantics for the different process algebras in [4, 11, 9] are given directly in terms of PSA and shown to be consistent with their original semantics. In Sect. 8 we report some notes and discussions concerning a detailed comparison of the approaches in [11, 9, 36, 17, 32, 6, 4]. Sect. 9 concludes the paper.

2 Concurrent systems with stochastic time

In this section we explain how to formally represent and reason about concurrent systems whose behavior is specified by means of activities with random duration. As explained in Sect. 1, we focus on the general case, i.e., the expressiveness of our specification paradigms is not limited to “memoryless” exponential distributions. As we will see, abandoning such limitation changes the nature of the problem and increases its complexity. In order to better understand this it is important to glance some details of the Markovian approach.

2.1 Exponential distributions make things easy

Restricting to exponentially distributed durations gives the advantage of having activities for which the memoryless property holds. This property basically says that at each time point in which an activity has started but not terminated yet, the residual duration of the activity is still distributed as the entire duration of the activity. Such a property makes it possible to represent “timed” behavior of systems by a continuous time Markov chain (CTMC), i.e. a simple continuous time stochastic process where in each time point the future behavior of the process is completely independent of its past behavior and depends on its current state only (Markov property). In fact the memoryless nature of time in the Markovian approach makes it possible to avoid the explicit representation of time passage in system specifications. For instance, consider a simple example of two exponentially timed activities with rates (the parameters of the exponential distribution) λ and μ executed in parallel. The resulting CTMC is the one in

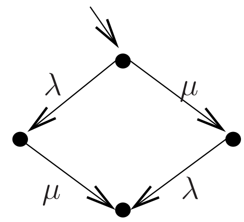


Fig. 2. Parallel of exponential delays

Fig. 2. Transitions in a CTMC represent exponentially distributed delays and choices in a state of a CTMC are resolved via the "race policy", i.e. the delays represented by the outgoing transitions are executed in parallel and the first delay that terminates determines the transition to be performed. Therefore, in the example, both delays count synchronously from the initial state and, when one of them terminates, the corresponding transition is executed. Such a transition leads to a state where the other delay counts its residual duration until it terminates as well. Note that, because of the memoryless property, the residual duration of the delay is also exponentially distributed and with the same rate. Hence the CTMC of Fig. 2 is an adequate representation of the parallel execution of the two considered activities.

If system behaviors in the form of CTMCs are obtained from system specifications expressed with a process algebra (see, e.g., [20, 19, 3, 5]), the "+" operator will be the natural choice to express the same kind of alternative given by transitions leaving a state of a CTMC. For example, $\lambda + \mu$ represents a choice between exponentially timed delays λ and μ and it is solved via the race policy explained above. As far as the "$||$" operator is concerned, the example of Fig. 2, should make clear that the simple standard interleaving semantics can be adopted: the semantics of $\lambda \,||\, \mu$ is just that of $\lambda.\mu + \mu.\lambda$. This example also shows that having a race policy interpretation of operator "+" and an interleaving semantics for operator "$||$" yields an expansion law like $\lambda \,||\, \mu = \lambda.\mu + \mu.\lambda$, which is crucial for building complete axiomatizations of process equivalence notions.

Regarding equivalences, it turns out that standard bisimulation equivalence (as opposed to other notions of equivalence) can be easily extended by following an approach similar to that of [26] for discrete probabilistic choices.

2.2 A symbolic model with clocks

When the restriction to exponential distributions is abandoned, the behavior in a global system state *does* depend on the time partially spent by activities currently under execution. Therefore, we are forced to explicitly represent the passage of time. For instance, if the two activities in the example of Fig. 2 are *not* exponentially distributed, the time spent in a state reached after the termination of the first activity effectively depends on the time the second activity has already (partially) spent in execution.

A simple way of representing the execution time of an activity is to adopt a model with clocks in a similar manner timed automata do [1]. A timed automaton represents the behavior of a system in terms of a fixed set of clocks c_1, c_2, c_3,.... During the execution of the automaton, each clock has an associated time value. When the automaton sojourns in a state, clock time values increase in a synchronous manner. The transitions of the automaton are executed instantaneously: their execution may depend on some condition on clocks (e.g. $c_1 \geq t$ for some time value t) and may cause some clock to be set to some value (e.g. c_3:=0). More precisely, labels of timed automata transitions are of three kinds:

- *actions*, used to express the occurrence of events and to synchronize system components (when composing in parallel several timed automata),
- *guards*, expressing a condition on clocks, and
- *clock setting* events.

Note that the representation of time passage in a timed automata is *symbolic* in the sense that the temporal behavior of the system is expressed by means of events like clock setting and clock constraints instead of *concrete* (real-valued) timed transitions. This makes it possible have a finite representation of system behavior, hence the chance of analyzing some of its properties in a computable way, even if the time domain is assumed to be (continuously) infinite. On the other hand the semantics of timed automata (the meaning of the symbolic representation) is usually defined in terms of an inherently infinite *concrete* semantic model which makes use of real-valued timed transitions.

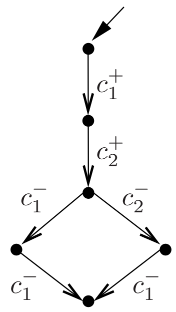
It is easy to see that a simple probabilistic extension of the clocks of a timed automaton gives the possibility of representing generally distributed time.

Introducing stochastic clocks

In order to express generally distributed time it is sufficient to consider a variant of timed automata where:

- each clock has an associated *probability distribution* expressing its *duration*: $c_1 \mapsto f$, $c_2 \mapsto g$, $c_3 \mapsto h, \ldots$;
- the three kinds of labels of a transitions are the following ones:
  - *actions* (represented with a, $b, \ldots$), used to express the occurrence of events and to synchronize system components (as in timed automata),
  - *guards*, requiring all clocks in a certain set to be *terminated*, and
  - *clock setting* events representing the *start* of all clocks in a certain set;
- we possibly may express probabilistic and/or prioritized choices.

By using an approach like this, two parallel activities with random duration can be easily represented even if they are not exponentially distributed. Suppose that the duration of two activities are generally distributed according to distributions f and g. The behavior of $f \,||\, g$ can be represented as in Fig. 3. In this figure, the execution of these activities is represented by clocks c_1 and c_2, where their associated distribution functions are f and g, respectively. A c^+ label means that the clock c is set to be started, while a c^- label represents the clock c to be



$$Dist(c_1) = f, \qquad Dist(c_2) = g$$

Fig. 3. Parallel of generally distributed delays

terminated. In this event-based representation, we assume that whenever a clock is not explicitly restarted it continues counting. Therefore, Fig. 3 is a correct representation of the parallel execution $f \,||\, g$. (Compare to the exponential case in Fig. 2, in which a clock does not need to be explicitly started.)

Models to represent systems like this, which may execute generally distributed timed activities in parallel, are also used in probability theory. In particular the class of generalized semi-Markov processes (GSMPs) exploits a similar event-based symbolic representation where clocks are also called "elements". GSMPs also provide the capability of expressing probabilistic choices.

2.3 A concrete semantics

Similarly as for classic timed automata, the meaning of the symbolic representation given by an automata with probabilistic clocks can be formally defined in terms of a concrete model with real-valued timed transitions.

A concrete probabilistic timed model

In the case that a continuous time domain is considered (as we do in this paper), the concrete model is given by a transition system on an uncountably large state space where time passage is represented explicitly via transitions labeled with real numbers. More precisely, the concrete model must have at least the following three kinds of transitions:

- *actions* transitions,
- *time* transitions, labeled with a time $t \in \mathbb{R}_{\geq 0}$,
- *continuous probabilistic* transitions represented by probability spaces.

In particular, continuous probabilistic choices are used to represent the *sampling of time values* from distributions on durations.

The easiest way to understand the concrete model is to see it as a representation of the behavior of the symbolic probabilistic automata when it is "executed" (i.e. when actual time values are sampled from duration distributions). In the literature, there are two techniques to represent this execution (i.e. to define the semantics of the symbolic automata):

– keeping track of *residual lifetime* of clocks, i.e. the amount of time that a clock has to spend in execution from the current time instant until its termination.
– keeping track of *spent lifetime* of clocks, i.e. the amount of time that a clock has already spent in execution since it was started until the current time instant.

While the residual lifetime of a clock decreases as time passes, the spent lifetime of a clock increases. This last case is similar to the way clock values are treated in the semantics of timed automata.

Recording residual lifetimes

The technique based on clock residual lifetimes (see e.g. [11]) is the simplest one: the duration of a clock is decided once and for all when a clock is started (by sampling from its associated distribution) and the duration time is decreased until it gets to zero.

More precisely the concrete semantic model is derived from the stochastic automata as follows:

– When a clock is *started*, its residual lifetime is determined by sampling a value from its associated duration distribution.
– When a clock is under execution, its time to termination t_{TERM} is given by its residual lifetime t_R ($t_{TERM} = t_R$). This means that if we consider the situation of the system after a time period t (i.e., a t transition is performed):
  - if $t < t_{TERM}$ then the residual lifetime of the clock becomes $t_R - t$,
  - if $t \geq t_{TERM}$ then the clock is *terminated*.

The main advantage of this technique, which is commonly used in discrete event simulation, is its simplicity and applicability (see e.g. [27]). However, it may be argued that this approach is not adequate if non-deterministic choices need to be resolved by adversaries. This is due to the fact that the eventual duration of a clock is decided when it starts. Hence, an adversary can base its decisions on the knowledge of the future behavior of the system and therefore, play a more angelic (or more demonic) role than it is desired. For instance, if three clocks are being concurrently executed in a symbolic state, an adversary a priory may know not only which one of the three clocks will terminate first in such a state, but also which clock will be the next one to terminate, thus obtaining information about the future behavior of the system.

Recording spent lifetimes

The technique based on clock spent lifetimes (see [4]) is more complex but similar to that of timed automata: the spent lifetime of a clock is set to zero when it starts and it increases as time passes. At every (concrete) state the distribution of the time to termination of the clock is resampled from its associated duration distribution conditioned to the spent lifetime.

More precisely the concrete semantic model is derived from the automata with stochastic clocks as follows:

- When a clock is *started* its spent lifetime is set to 0.
- When a clock is under execution, its time to termination t_{TERM} is determined by sampling a value from its *residual duration distribution*. This is computed from:
  - the spent lifetime t_S of the clock and
  - the duration distribution associated to the clock.

  This means that if we consider the situation of the system after a time period t (i.e., a t transition is performed):
  - if $t < t_{TERM}$ then the spent lifetime of the clock becomes $t_S + t$,
  - if $t \geq t_{TERM}$ then the clock is *terminated*.

This technique presents an appropriate context for the resolution of non-determinism by adversaries. However, it cannot easily be taken into practice by means of discrete event simulation. This would require to compute the residual duration distribution of every clock in execution and sample a value from it, and this repeated at every state traversed by the automata.

2.4 Bisimulation equivalences

Similarly to the Markovian case, we want to find some extension of standard bisimulation equivalence which makes it possible to reason about equivalence of systems with generally distributed durations.

It is important to have a definition of bisimulation equivalence both at the symbolic model and at the concrete model level. Equivalences at the symbolic level make it possible to actually decide the equivalence of two systems and to minimize the state space of a system in a computable way. Equivalences at the concrete level show that equivalence at the symbolic level is correct with respect to the adopted concrete semantics. Note that in general, at the concrete level, systems can be compared on the basis of actual time delays and probability spaces, while at the symbolic level we can only check correspondence of clock-events and correspondence of probability distributions. Therefore equivalence at the concrete level represents, somehow, the coarsest notion of equivalence that we can hope to gain with a symbolic equivalence (in fact the symbolic equivalence is typically much finer than the concrete one).

Symbolic bisimulation

Symbolic bisimulation is an equivalence defined on the symbolic stochastic timed model. It is defined in such a way that:

- *action* transitions are matched as in standard bisimulation,
- *clock start* and *clock termination* events are matched if they refer to clocks with the same duration distribution.

Moreover if the symbolic model includes probabilistic choices they are matched as in standard probabilistic bisimulation [26].

Two approaches may be followed in order to match clock start and termination. The simplest one is to only match clocks if they have the same name, therefore ensuring that duration distributions are the same. The advantage of this approach is its simplicity; its disadvantage is that it hardly provides any stochastic insight.

The other approach relies on a *clock name association*, given by a function or a relation. This relation ensure that start and termination events of clocks with different names but with the same duration distribution are properly matched. This second type of symbolic bisimulation is coarser than the first one.

Examples of the first case appear in [14, 11]; examples of the second one can be found in [6, 11, 4].

A particularly distinct case appears in [8, 4] where symbolic models can be restricted to canonical one (in the sense that clocks are canonically named) for which building clock name associations is not needed (it works just like the first type of equivalence) and nonetheless it equates symbolic automata just like the second kind.

Concrete bisimulation

Concrete bisimulation is an equivalence defined on the concrete timed model. In particular it is defined in such a way that:

- *action* transitions and *timed* transitions are matched as in standard bisimulation,
- *continuous probabilistic* transitions are matched according to an extension of probabilistic bisimulation [26] to continuous probability spaces.

2.5 Dealing with composition: process algebra

The essence of designing a process algebra for modeling systems with generally distributed activities can be captured by simply considering the extension of a standard process algebra with a new prefix or guard "$f.P$". This new operation represents the execution of P after a random delay (sampled from the general distribution f) took place.

In order to obtain a symbolic model with clocks just like those previously described, the semantics of $f.P$ should take into account the following:

– it should represent the execution of the delay as the combination of a *start* and a *termination* event (e.g. denoted by f^+ and f^-, respectively).
– it must generate a (clock) *name* for the delay which keeps it distinguished from other delays being executed at the same time (e.g. in $f \mid\mid f$).

If we see f as being the "type" of the delay, a semantics which operates in this way is exactly the classical ST semantics of [15, 7]. Therefore, conceptually, the essence of the problem of representing symbolically general distributions is just like the classical problem of expressing ST semantics for actions in algebraic terms.

As in the standard case of ST semantics (see [7]), we can employ different kind of techniques for generating clock names from delay prefixes. In particular such techniques can be classified into *static* and *dynamic* techniques. Note that it is also possible (see [11]) to define the semantics of the process algebra by abstracting from the particular name generation mechanism. We can just say that names are generated by *arbitrary* α*-conversion*: any name can be chosen for a delay provided that it is not in use by another clock in execution. Terms are then considered to be equivalent up to (distribution preserving) clock associations.

On the other hand, choosing a particular technique (static or dynamic) may lead to smaller models or may make it easier to check symbolic bisimulation equivalence at the price of introducing some (often quite complex) unique name generation mechanism.

In static techniques, clock names are generated statically according to the *syntactic structure* of the process algebra term. For instance, in [6] clocks are named according to the syntactic position of the delay w.r.t. the parallel operators in the term (the *location* of the delay). A simple rule like this is enough to guarantee that two delays that may be executed at the same time always get different names. The main advantage of the static approach is simplicity and the size of the symbolic state space which is smaller than the one of the dynamic approach, due to the fact that a delay always gets the same name independently on when it is executed. The main drawback is that clock names must be explicitly associated in the symbolic bisimulation (see, e.g., [6, 11]) in order to capture equivalence of systems just based on duration distributions.

Example 1. In figure 4 we depict the behavior of "$f \mid\mid f$" (only the phase of clock starts) according to a static technique like that of [6] which assigns names to delays according to their position w.r.t. parallel operators. In the example of figure 4 the delay f to the left of the parallel operator is named "f_l", the delay f to the right "f_r". Note that, since names of delays are determined by their syntactical position in the current term, after an event of starting of a delay f, we just have to record that this happened by turning f prefix into f^- in the term (we do not have to record names assigned to starting delays).

In dynamic techniques, clock names are generated dynamically (at run-time) according to the order of delay execution, with a fixed rule. For instance, in the approach of [8] the clock name generated for a delay f is f_i, where i is the least $i \in \mathbb{N}$ which is not used in the name of any clock with the same distribution f

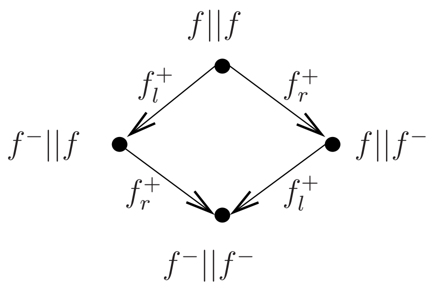


Fig. 4. Parallel of delays with the static technique (starting phase)

already in execution. The main advantage of the dynamic approach is the fact that clock names do not need to be explicitly associated in the symbolic bisimulation. This is due to the fact that the method to compute new names is fixed: processes that perform equivalent computations generate the same "canonical" names for clocks. The main drawback is the complex mechanism to compositionally generate canonical names in operational semantics (we have to perform a so-called level wise renaming [8, 4]) and the size of the state space which is larger than the one of the static approach since the same delay may get different names depending on the moment in which it is executed with respect to other delays.

Example 2. In figure 5 we depict the behavior of "$f \,||\, f$" (only the phase of clock starts) according to a dynamic technique like that of [8] which assigns names to delays (indexes $i \in \mathbb{N}$) according to the rule of the minimum index not currently in use by delays with the same distribution. In the example of figure 5, depending on the order in which delays are started, either the lefthand one gets name 1 and the righthand one gets name 2, or vice-versa. Note that, since names of delays are determined by the order in which they are executed, after an f_i^+ event, we have to record, not only that this happened by turning f prefix into f^- in the term, but also the name i assigned to the delay f at the moment of delay start. As a consequence we produce an additional state w.r.t. the static naming case.

It could be said that [9, 23] also follow a dynamic approach. However, as event names are arbitrarily chosen, it does not have the advantages of using "canonically" chosen names. Other approaches do not generate a symbolic model as an intermediate semantics, therefore they are not concerned with clock name generation.

Basic operators in the case of pure ST semantics

As far as the choice operator "+" is concerned, due to the generation of start and termination events f^+ and f^- from delays f operated by ST semantics, we have that (intuitively)

$$f + g = f^+.f^- + g^+.g^-$$

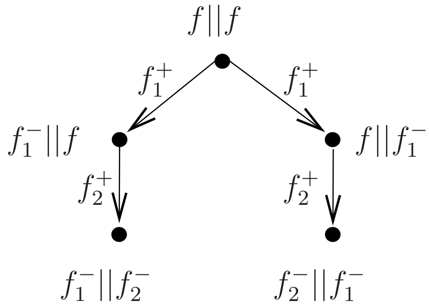


Fig. 5. Parallel of delays with the dynamic technique (starting phase)

i.e. choices between delays become *choices between starting events* in semantic models. Since start events are executed immediately this means that choices are solved by a *preselection policy*, where first we choose the delay to be performed and then we execute it, instead of the *race policy* used with exponential distributions. A possible justification of this fact (given in [8, 4]) is the following one. Preselection policy can be claimed to be more natural than race policy when the restriction to exponential distributions (justifying race policy) is abandoned. This because it causes $f + g$ to represent a real choice and not a form of parallel execution. In particular in the approach of [8, 4] probability information is attached to delays (by using "weight" values) and the choice between start events is resolved probabilistically. For instance, $<f, 1>.P + <g, 2>.Q$ means that $1/3$ of the time the delay f is chosen, after which P is executed, while the other $2/3$ the delay g followed by process Q is executed. This choice allows for a representation of the full GSMP model [10] (with the only restriction that we assume "decay rate" of delays to be always 1). On the other hand, adopting preselection policy for generally distributed delays is absolutely not mandatory (it is just a matter of taste) and as we will see, there are alternative technical solutions (e.g. that of [11]) which allow race policy for the "+" operator to be used.

As far as the parallel operator "||" is concerned, due to the generation of start and termination events f^+ and f^- from delays f operated by ST semantics, we have that (intuitively):

$$f \mid\mid g = f^+.f^- \mid\mid g^+.g^- = f^+.g^+.(f^-.g^- + g^-.f^-) + g^+.f^+.(f^-.g^- + g^-.f^-)$$

i.e. parallel of delays becomes *interleaving of events* in semantic models. [1] This shows that even in the case of general distributions it is possible to have an *expansion law at the level of events*. This makes it possible to produce complete axiomatizations for symbolic bisimulation equivalence (see [11, 4]). In the history of stochastic process algebra with general distributions obtaining an expansion

[1] For the sake of simplicity we assume delay starts to have precedence over delay terminations of another process as in [8, 4], hence we do not generate the complete interleaving of events.

law has been an important issue. First approaches failed to obtain expansion laws providing, at best, only partial decomposition (e.g. [17, 36, 32, 18]). Others pursued a true concurrency approach, dropping the idea of having an expansion law [9, 23]. Modern stochastic process algebras solved this problem and the solution depends on the splitting of the delay into start and termination *events*: parallel of delays becomes the *interleaving of events*.

Technical solutions different to ST semantics

The treatment of generally distributed durations in process algebra that we explained above is just intended to be conceptual and adheres completely to the approach of [8] only.

In the literature we can find approaches which essentially follows the concept that we explained, but adopt some different technical solution:

- We can have *event separation* already in the specification language [11]. In this case clock events C^+ and C^- are used directly in the initial system algebraic specification instead of delays f. This somehow gives more specification freedom, but forces the specifier to deal with clock names at the algebraic level.
- We can have that *clock start* events do *not resolve the choice* [11]. In this case the choice operator treats start events in a "special" way, i.e. in such a way that their execution does not resolve the choice. Therefore having that (intuitively):

$$(\{c_1^+\}.c_1^-) + (\{c_2^+\}.c_2^-) = \{c_1^+, c_2^+\}.(c_1^- + c_2^-)$$

  where $\{c_1^+, c_2^+, ...\}$ represents a set of clocks starting at the same time.
  In this way we can obtain *race policy* instead of *preselection policy* for choice even in the case of generally distributed durations.
- The possibility of dealing with *termination of multiple clocks* can be introduced [11, 6]. This basically means that we can specify systems behaving like this (intuitively):

$$\{c_1^+, c_2^+\}.(\{c_1^-\}.P + \{c_2^-\}.Q + \{c_1^-, c_2^-\}.R)$$

  where $\{c_1^-, c_2^-, \dots\}$ represents a set of terminated clocks.

2.6 Totally different approaches

In the following we briefly discuss approaches which deal with general distributions in a very different way with respect to the methodology above.

- The approach of [9] is based on a *truly concurrent* semantics for the process algebra. The semantics of process algebraic specifications is given in terms of *stochastic bundle event structures* instead of transitions over global system states. Therefore it does not represent "interleaved" execution of processes, but keeps instead a "local" representation of the behavior of each process. The advantage is that it does not need to represent residual duration distributions of generally distributed activities. However, it is not so clear how to use truly concurrent models for actual system analysis apart from discrete event simulation [24] and analysis of the first passage time of events [34] (a technique to approximate performance measures).

- The approach of [36, 18] is based on a *direct concrete semantics* for the process algebra. The semantics of process algebraic specifications is given directly in terms of a concrete model with explicit time. This is a fine approach if we only expect to perform discrete event simulation of systems. The main advantage is that it is easy to deal with very expressive process algebras: we do not have to worry about how to develop symbolical representations. However, concrete models are mostly uncountably large which makes them unsuitable for system analysis not based on simulation. In a recent work [22] a methodology for obtaining finite semantic models from the algebra of [36, 18] is defined, which is based on symbolic operational semantics. Such semantics generates symbolical transition systems which abstract from time values by representing operations on values as symbolic expressions. In this way, for systems belonging to a certain class, it is possible to derive a GSMP via a (quite involved) procedure.
- The approach of [29] defines a testing theory for semi-Markov processes. This is done by using a process algebra which is similar to IGSMP [8, 4], but where the parallel operator is left out. Tests are processes which do not include probabilistic delays (just actions and discrete probabilistic choices).

3 A common model for stochastic timed systems

The model we introduce in this section considers the ingredients discussed in the previous section: clock start, clock termination, execution of actions, and probabilistic jumps. All these ingredients are included in only one symbolic transition. The aim of this model is to encode many formalisms for modelling stochastic timed systems, hence having a common framework to formally compare existing approaches. In addition, this model also considers priorities because some frameworks implicitly include this feature and, moreover, they make it possible to represent maximal progress and urgency. This model is an extension of stochastic automata [14, 11] with priorities and probabilistic transitions.

With *PDF* we denote the set of all probability distributions functions and with $\mathit{Prob}(\Omega)$ we denote the set of all probability spaces with sample space in a subset of Ω. We let $\mathit{Prob}_d(\Omega)$ denote the subset of $\mathit{Prob}(\Omega)$ containing only discrete probability spaces. We use $\rho, \rho', \rho_i, \ldots$ to denote discrete probability spaces and $\pi, \pi', \pi_i, \ldots$ to denote probability spaces in general. In both cases, we will overload the notation and use these letters to represent the probability measure as well.

Definition 1. *A* prioritized stochastic automata *(PSA) is defined to be a structure* $(\mathit{St}, \mathit{Ck}, \mathit{Distr}, \mathit{Act}, \longrightarrow, s_0, C_0)$ *where*

– *St is a countable set of* control states *with* $s_0 \in St$ *being the* initial control state.
– *Ck is the set of* clock names *with* $C_0 \subseteq Ck$ *being the set of* clocks to be started at initialization. *For the sake of clarity in technical manipulation, we assume that Ck is totally ordered, and that if* $C \subseteq Ck$, $\vec{C}$ *is the vector induced by this order.*
– $\mathit{Distr} : Ck \to PDF$ *assigns a probability distribution function to each clock. If* $f \in PDF$, *we usually name a clock* $c_f \in Ck$ *to indicate that* $\mathit{Distr}(c_f) = f$.
– *Act is set of* actions *partitioned in the following sets:*
  • Act_d*, the set of* delayable *actions and*
  • Act_u*, the set of* urgent *actions*

  where Act_u *is ordered according to a* priority relation $\prec$ *which is a strict order with the* silent action $\tau \in Act_u$ *being the maximum element.*
– $\longrightarrow \subseteq St \times 2^{Ck} \times Act \times Prob_d(2^{Ck} \times St)$ *is the* control transition *relation.*

We write $s \xrightarrow{C,a} \rho$ if $(s, C, a, \rho) \in \longrightarrow$ and $s \xrightarrow{C,a}$ if there is a distribution ρ such that $s \xrightarrow{C,a} \rho$. If $\rho(C', s') = 1$ (i.e., ρ is trivial) then we write $s \xrightarrow{C,a,C'} s'$ instead of $s \xrightarrow{C,a} \rho$. The meaning of a control transition $s \xrightarrow{C,a} \rho$ is the following. To trigger the transition, all clocks in set C must terminate, that is, the transition cannot be executed as long as a clock in C is active. Transitions are labeled with actions. They can be delayable, meaning that they need to interact with the environment, or they can be urgent and they will not be allowed interact with the environment. Urgent actions impose maximal progress and therefore transitions labeled with actions of this type must be executed as soon as they are enabled. In addition, if a conflict between two enabled urgent transitions occurs, it may be solved according to the priority relation on the actions labelling the control transition provided it is defined (notice that the order may not be total). In addition, when a transition is executed, a probabilistic branching will take place according to the probability space ρ. $\rho(C', s')$ is the probability that all clocks in C' are started and the system reaches the control state s'.

Fig. 6 shows a system of three queues with two servers to process certain classes of job. Jobs arrive to the system according to an unknown rate and they queue in a buffer. Server A is intended to take jobs from this buffer a soon as it can and preprocess them. This preprocessing takes some (random) time. Moreover 1/3 of the outcome will result in a high priority job and the other 2/3 in a low priority one. As soon as the preprocessing finishes the job is queued in the high priority buffer or in the low priority one, depending whether it is high or low priority. Service B is intended to perform some postprocessing on these jobs. As soon as it can, it takes and process a job from the high priority queue provided there is any, otherwise it takes a job from the low priority queue. Processing takes a (random) time, and as soon as it finishes, the result is output. Notice that, since all ingredients above are present in control transitions $s \xrightarrow{C,a} \rho$, the PSA model allows for the modelling of systems like the queuing network depicted in Fig. 6.

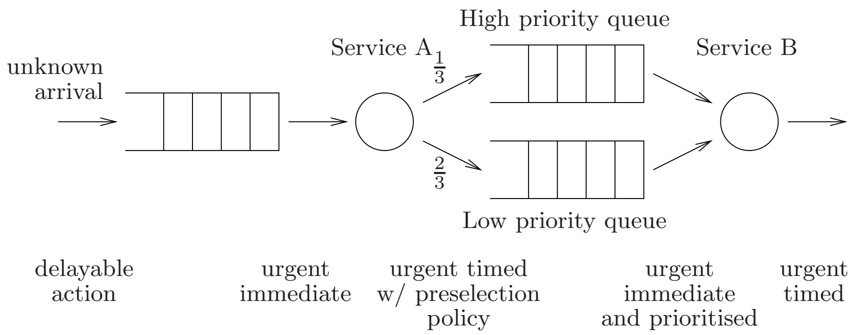


Fig. 6. A simple queuing network

Let *Jobs* be the possible set of jobs to be processed in the queuing network of Fig. 6. Suppose the serving time in A (resp. B) is distributed according to a distribution F_A (resp. F_B). Then, the queuing network can be modelled by the PSA defined as follows:

$$St = Jobs^* \times Jobs^* \times Jobs^* \times (Jobs \cup \{-\}) \times (Jobs \cup \{-\})$$
$$Ck = \{x_A, x_B\} \quad \text{with } Distr(x_A) = F_A \text{ and } Distr(x_B) = F_B$$
$$Act_d = \{input(j) \mid j \in Jobs\}$$
$$Act_u = \{get(j), put(j), getHigh(j), getLow(j), output(j) \mid j \in Jobs\}$$

where $getLow(j) \prec getHigh(j')$ for every $j, j' \in Jobs$ and the other actions in Act_u are not comparable.

A control state (q_I, q_H, q_L, A, B) saves in q_I, q_H, and q_L the contents of the input queue, the high priority queue, and the low priority queue, respectively, and A (resp. B) indicates if server A (resp. B) is processing a job $j \in Jobs$ or it is idling $(-)$. Initially, the system has all its queues empty and the servers are idling. Therefore $s_0 = (\epsilon, \epsilon, \epsilon, -, -)$. Moreover, initially the system is only waiting for a job to arrive so no timer needs to be set: $C_0 = \emptyset$. The control transitions are defined as follows (where $j \in Jobs$).

a job can be input at any time:

$$(q_I, q_H, q_L, A, B) \xrightarrow{\emptyset, input(j), \emptyset} (q_I j, q_H, q_L, A, B)$$

if server A is idling and q_I is not empty it must take a job and begin to process it (it starts clock x_A):

$$(j q_I, q_H, q_L, -, B) \xrightarrow{\emptyset, get(j), \{x_A\}} (q_I, q_H, q_L, j, B)$$

when clock x_A terminates, server A finishes processing its job and appends it to q_H with probability $1/3$, or to q_L with $2/3$:

$$(q_I, q_H, q_L, j, B) \xrightarrow{\{x_A\}, put(j)} \left\{ \begin{array}{l} \langle \emptyset, (q_I, q_H j, q_L, -, B) \rangle \mapsto \frac{1}{3}, \\ \langle \emptyset, (q_I, q_H, q_L j, -, B) \rangle \mapsto \frac{2}{3} \end{array} \right\}$$

if server B is idling and q_H is not empty it must take a job and begin to process it (it starts clock x_B), and similarly if q_L is not empty:

$$(q_I, jq_H, q_L, A, -) \xrightarrow{\emptyset, getHigh(j), \{x_B\}} (q_I, q_H, q_L, A, j)$$
$$(q_I, q_H, jq_L, A, -) \xrightarrow{\emptyset, getLow(j), \{x_B\}} (q_I, q_H, q_L, A, j)$$

when clock x_B terminates, server B finishes processing its job and outputs it:

$$(q_I, q_H, q_L, A, j) \xrightarrow{\{x_B\}, output(j), \emptyset} (q_I, q_H, q_L, A, -)$$

The server B prioritizes jobs from the high priority queue because $getLow(j) \prec getHigh(j')$ for every $j, j' \in Jobs$.

The PSA model is obtained by integrating the stochastic automata model [14, 11] with the IGSMP model [6, 8, 4] and it is based on the model used for the MoDeST language [13].

4 Semantics of PSA

The semantics of PSA is given in terms of probabilistic timed transition systems (PTTS). As discussed in Section 2.3, there are two possible interpretations. This is discussed in the following.

4.1 Probabilistic timed transition systems

The PTTS model is an extension of Segala's simple probabilistic automata [35] with continuous probability spaces and time labelled transitions.

Definition 2. *A* probabilistic timed transition system *(PTTS) is a structure* $(\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \pi_0)$ *where*

- Σ *is a set of* states*;*
- Act *is a set of* actions *like in PSA;*
- $\longrightarrow \subseteq \Sigma \times (Act \cup \mathbb{R}_{\geq 0}) \times Prob(\Sigma)$ *is the* transition relation*; and*
- $\pi_0 \in Prob(\Sigma)$ *is the probability space that indicates how to select a probable initial state.*

In addition, the following requirements must hold:

1. *maximal progress:* $\forall \sigma \in \Sigma, a \in Act_u.\ \sigma \xrightarrow{a} \implies \nexists t \in \mathbb{R}_{\geq 0}.\ \sigma \xrightarrow{t}$
2. *priority:* $\forall \sigma \in \Sigma, \{a, b\} \subseteq Act_u.\ (a \prec b \wedge \sigma \xrightarrow{a}) \implies \sigma \overset{b}{\not\longrightarrow}$

Above we use the following notation. $\sigma \xrightarrow{a} \pi$ *denotes* $(\sigma, a, \pi) \in \longrightarrow$*. If there is a probability space* π *such that* $\sigma \xrightarrow{a} \pi$*, we write* $\sigma \xrightarrow{a}$*; if such* π *does not exists, we write* $\sigma \overset{a}{\not\longrightarrow}$ *Moreover, we write* $\sigma \xrightarrow{a} \sigma'$ *if* $\sigma \xrightarrow{a} \pi$ *and* π *is a trivial probability space where the atom* $\{\sigma'\}$ *has measure 1.*

Transition $\sigma \xrightarrow{a} \pi$ means that when the system is in state σ it may perform an action a and then move to some state with a probability determined by the probability space π. π may be a continuous probability space.

As an example, consider a metronome. A metronome is a device that marks the tempo of a piece of music (i.e. the speed at which it should be played). Thus, a metronome is a clock that ticks once each given interval of time. Suppose our metronome always plays *andante*; this would be a tick per second. As any clock a metronome may not be precise. Our metronome may skew up to 1 millisecond according to a uniform distribution. A possible modelling of the metronome's behaviour in terms of PTTS could be as follows (time is measured in milliseconds).

$$
\begin{array}{ll}
\Sigma = \mathbb{R} & \\
Act_d = \emptyset & Act_u = \{tick\} \\
0 \xrightarrow{tick} \pi_U & \text{where } \pi_U \text{ is the probability space in } \mathbb{R} \text{ with} \\
 & \text{measure defined from a uniform in } [999, 1001] \\
t + t' \xrightarrow{t} t' & \text{with } t, t' > 0 \\
\pi_0 = \pi_U &
\end{array}
$$

Notice that the metronome satisfies maximal progress since $0 \stackrel{t}{\not\longrightarrow}$ for any real t. Maximal progress requires that when an urgent action becomes available, it must be executed without letting time pass. (See first requirement in Def. 2.)

To add an on/off button to the metronome, the previous PTTS should be modified as follows:

$$
\begin{array}{l}
\Sigma = \mathbb{R} \cup \{sys\_off\} \\
Act_d = \{on, off\} \\
\text{the transition relation is extended s.t. } sys\_off \xrightarrow{on} 0, \\
\qquad t \xrightarrow{off} sys\_off, \text{ and } sys\_off \xrightarrow{t} sys\_off \text{ for all } t \geq 0
\end{array}
$$

with the rest of the components as before.

Our metronome is not as sophisticated as to include a priority scheme, but if it did, no state could show a non-deterministic choice between a high-priority labelled transition and a low-priority one. This is stated by the second requirement in Def. 2

We give two different interpretations to PSA in terms of PTTSes, one that observe the residual lifetime of the clocks to decide the enabling of a transition, and the other that observe the spent lifetime.

4.2 Residual lifetime semantics

In the *residual lifetime semantics*, when a clock c is started, its termination time is sampled according to $Distr(c)$. The clock c is *active* as long as it does not reach this termination time. Otherwise we say it is *terminated*. Therefore, in this context, a control transition $s \xrightarrow{C,a} \rho$ becomes enabled as soon as all clocks in C are terminated in the sense above. To carry the time value and the termination value of a clock, we use valuations. A *valuation* is a partial function v from Ck in the set $\mathbb{R}_{\geq 0}$ of non-negative real numbers. Let Val be the set of all valuations, and $TVal$ be the set of all total valuations on Ck, i.e. valuations such that the underlying function is total.

A state in the semantics is a triple (s, v, e) where s is a control state in the PSA, v is the *time valuation*, and e is the *enabling valuation*: both v and e are total valuations. Therefore, given a clock c, $v(c)$ registers the time that has passed since c was started, and $e(c)$ registers its termination time which was sampled when it was started. Notice that c is active if $v(c) \leq e(c)$. The difference $e(c) - v(c)$ is c's residual lifetime. As a consequence, a control transition $s \xrightarrow{C,a} \rho$ is enabled in state (s, v, e) whenever $v(c) \geq e(c)$ for all clocks in C. This is denoted by the predicate $enabled(s \xrightarrow{C,a} \rho, v, e)$,

Let $\mathcal{R}(f_1, \ldots, f_k)$ be the probability space in the k-dimensional real space with the unique probability measure induced by the probability distribution functions $f_1, \ldots, f_k$. For $C = \{c^1_{f_1}, \ldots, c^k_{f_k}\} \subseteq Ck$, $\mathcal{R}(Distr(\vec{C}))$ denotes the probability space $\mathcal{R}(f_1, \ldots, f_k)$.

Besides, we use the following notation. Given a probability space π and $p \in [0, 1]$, $p \cdot \pi$ is the measurable space obtained by multiplying p to the probability measure of π. Given a denumerable set of probability spaces π_i, $i \in I$, $\sum_{i \in I} \pi_i$ is the measurable space obtained by appropriately summing the measures of the different probability spaces. Given a probability space π and a function f defined on the domain of π, we take $f(\pi)$ has being the probability space on the range of f induced from f [4].

Definition 3. *Let* $\mathsf{PSA} = (St, Ck, Distr, Act, \longrightarrow, s_0, C_0)$. *Its* residual lifetime semantics *is defined by the PTTS* $[\![\mathsf{PSA}]\!]_r = (\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \pi_0)$ *where:*

- $\Sigma \stackrel{\text{def}}{=} St \times TVal \times TVal$
- $\longrightarrow$ *is defined by the following rules:*

$$\frac{\begin{array}{c} enabled(s \xrightarrow{C,a} \rho, v, e) \\ a \in Act_u \implies \nexists b \in Act_u.\ a \prec b \wedge enabled(s \xrightarrow{C'',b} \rho'', v, e) \end{array}}{(s, v, e) \xrightarrow{a} \sum_{\substack{s' \in St \\ C' \subseteq Ck}} \rho(C', s') \cdot sample^{s',C'}_{v,e}(\mathcal{R}(Distr(\vec{C'})))} \qquad (1)$$

$$\frac{\forall t'.\ 0 \leq t' < t \implies \forall a \in Act_u.\ \neg enabled(s \xrightarrow{C,a} \rho, v + t', e)}{(s, v, e) \xrightarrow{t} (s, v + t, e)} \qquad (2)$$

- $\pi_0 \stackrel{\text{def}}{=} sample^{s_0,C_0}_{v_0,v_0}(\mathcal{R}(Distr(\vec{C_0})))$

where

– $enabled(s \xrightarrow{C,a} \rho, v, e) \stackrel{\text{def}}{\iff} \forall c \in C.\ v(c) \geq e(c)$;
– $sample^{s,C}_{v,e}(\vec{t}) \stackrel{\text{def}}{=} (s, v[\vec{C}/0], e[\vec{C}/\vec{t}])$; *and*
– *for all* $c \in Ck$, $v_0(c) \stackrel{\text{def}}{=} 0$, $(v+t)(c) \stackrel{\text{def}}{=} v(c) + t$, *and* $v[\vec{C}/\vec{t}](c) \stackrel{\text{def}}{=} \vec{t}(i)$ *whenever there is an index* i *such that* $c = \vec{C}(i)$, *otherwise* $v[\vec{C}/\vec{t}](c) \stackrel{\text{def}}{=} v(c)$.

Rule (1) defines the execution of a control transition $s \xrightarrow{C,a} \rho$ requiring, therefore, that it is enabled. Notice that if a is an urgent action, it is also necessary to check that no control transition with higher priority is also enabled. In addition, the postcondition of the concrete transition is a random selection of a control state together with the set of clocks to be started and a sample terminating value for this clocks. Function $sample^{s',C'}_{v,e}$ takes care of appropriately constructing the next state from a tuple of sampled time values. Note that $sample^{s',C'}_{v,e}$ is applied to a probability space on tuples of time values, thus yielding an induced probability space on states.

Rule (2) controls the passage of time. It states that the system is allowed to stay in the control state s as long as no urgent action becomes enabled. As a consequence, maximal progress on urgent action is ensured.

Besides, the initial state is determined by the initial control state together with a sample of terminating values for C_0 (clocks not in C_0 are considered to be terminated).

4.3 Spent Lifetime Semantics

In the *spent lifetime semantics* it is only important to keep track of the time value of the clock since its termination time is continuously resampled conditioned to the time that has passed since it was started. As before, a state is a triple (s, v, e) but now valuations are partial functions. A clock c is *active* whenever v is defined in c and, in this case, $v(c)$ is its spent lifetime. Otherwise we say it is *terminated*. $e(c)$ is only defined if c is sampled with the smallest time value among the active clocks and $e(c)$ is that value (note that this may hold true for several active clocks c). If time passes or an event occurs, this enabling value is resampled among the clocks which are still active, and function e changes according to this. Therefore, in this context, a control transition $s \xrightarrow{C,a} \rho$ becomes enabled at state (s, v, e), denoted $enabled(s \xrightarrow{C,a} \rho, v, e)$, if every clock in C is either terminated in the sense above or it has been sampled with value 0 (i.e., $e(c) = 0$).

Definition 4. *Let* $\mathsf{PSA} = (St, Ck, Distr, Act, \longrightarrow, s_0, C_0)$. *Its* spent lifetime semantics *is defined by the PTTS* $[\![\mathsf{PSA}]\!]_s = (\Sigma, Act \cup \mathbb{R}_{\geq 0}, \longrightarrow, \pi_0)$ *where:*

– $\Sigma \stackrel{\text{def}}{=} St \times Val \times Val$

– $\longrightarrow$ *is defined by the following rules:*

$$\frac{\begin{array}{c}\mathit{enabled}(s \xrightarrow{C,a} \rho, v, e)\\ a \in \mathit{Act}_u \implies \nexists b \in \mathit{Act}_u.\ a \prec b \wedge \mathit{enabled}(s \xrightarrow{C'',b} \rho'', v, e)\end{array}}{(s, v, e) \xrightarrow{a} \sum_{\substack{s' \in St \\ C' \subseteq Ck}} \rho(C', s') \cdot \mathit{PSpace}(s', (v - C)[\vec{C'}/0])} \tag{3}$$

$$\frac{\forall c \in \mathrm{dom}(e).\ 0 \leq t < e(c) \qquad \forall a \in \mathit{Act}_u.\ \neg \mathit{enabled}(s \xrightarrow{C,a} \rho, v, e)}{(s, v, e) \xrightarrow{t} \mathit{PSpace}(s, v + t)} \tag{4}$$

$$\frac{\forall c \in \mathrm{dom}(e).\ 0 \leq t = e(c) \qquad \forall a \in \mathit{Act}_u.\ \neg \mathit{enabled}(s \xrightarrow{C,a} \rho, v, e)}{(s, v, e) \xrightarrow{t} \mathit{PSpace}(s, (v + t) - \mathrm{dom}(e))} \tag{5}$$

Notice that in rules (4) and (5), if $e = \emptyset$ *(and* $v = \emptyset$ *as a consequence) then the precondition holds by emptiness and then* $(s, v, \emptyset) \xrightarrow{t} \mathit{PSpace}(s, v + t)$ *for any* $t \in \mathbb{R}_{\geq 0}$.

– $\pi_0 \stackrel{\text{def}}{=} \mathit{PSpace}(s_0, \emptyset[\vec{C_0}/0])$

where

– $\mathit{enabled}(s \xrightarrow{C,a} \rho, v, e) \stackrel{\text{def}}{\iff} \forall c \in C \cap \mathrm{dom}(v).\ c \in \mathrm{dom}(e) \wedge e(c) = 0$;
– $\mathit{PSpace}(s, v) \stackrel{\text{def}}{=} \mathit{sample}^s_v(\mathcal{R}([\mathit{Distr}(c_1) \mid v(c_1)], \ldots, [\mathit{Distr}(c_k) \mid v(c_k)]))$ *provided* $\mathrm{dom}(v) = \{c_1, \ldots, c_k\}$ *with* $\mathit{sample}^s_v(t_1, \ldots, t_k) \stackrel{\text{def}}{=} (s, v, \{c_j \mapsto t_j \mid t_j = \min_i t_i\})$[2]*; and*
– $v - C$ *is undefined in* C *and whenever* v *is not defined, otherwise it takes the same values as* v.

Rule (3) defines the execution of a control transition $s \xrightarrow{C,a} \rho$ and the hypotheses are as before: it must be enabled and if a is urgent, no control transition with higher priority is also enabled. The postcondition of the concrete transition is a random selection of a control state together with the set of clocks to be started and a sample terminating value for these clocks. However, in this case, the sample is taken for all active clocks and conditioned to the time that has already passed. Function sample^s_v takes care of appropriately constructing the next state ensuring that the enabling valuation e is only defined for the clocks whose sampled value is minimal among all the sampled values.

Rule (4) and (5) control the passage of time. Rule (4) defines the case in which no clock has reached its termination instant, while rule (5) considers the other case. Notice that for both cases it is required that no urgent action becomes enabled before letting time pass. Observe also that after letting time pass clock termination times are resampled. Besides, rule (5) ensures that terminated clock are removed from the domain of the time valuation v.

[2] $[f \mid t]$ is defined by $[f \mid t](t') \stackrel{\text{def}}{=} P(T \leq t + t' | T \geq t)$ where T is a random variable with distribution f.

4.4 A note on the difference between the two semantics

As it was already noted, residual lifetime semantics samples the clock termination time only once, namely, when the clock is started, while the spent lifetime semantics keeps resampling the termination value of the active clocks. This induces two main technical differences between the interpretations $[\![\mathsf{PSA}]\!]_r$ and $[\![\mathsf{PSA}]\!]_s$.

A first notorious difference is that timed transitions in $[\![\mathsf{PSA}]\!]_r$, i.e. transitions labelled with a real number, are trivial (namely, of the form $\sigma \xrightarrow{t} \sigma'$), while this is not the case in $[\![\mathsf{PSA}]\!]_s$. Here, transitions have the form $\sigma \xrightarrow{t} \rho$ where ρ is a probability space representing the resampling of all active clocks conditioned that t units of time have passed. To notice the difference compare rule (2) in Def. 3 with rules (4) and (5) in Def. 4.

The other difference lies in the form of the state, more precisely, in the valuations. Given a state (s, v, e) of $[\![\mathsf{PSA}]\!]_s$, $\mathrm{dom}(v)$ contains exactly all active clocks at this state, while $\mathrm{dom}(e) \subseteq \mathrm{dom}(v)$ contains those clocks that have been sampled as the clocks next to terminate (which may vary if time passes). In $[\![\mathsf{PSA}]\!]_r$, valuations are total functions. In this case, given (s, v, e), c is active if $e(c) - v(c) \geq 0$, i.e. if the elapsed time since c was activated did not reach the enabling value. As a consequence, predicate *enabled* has to be accommodated to this difference. Also rule (5) has to accommodate the fact that clocks in $\mathrm{dom}(e)$ are terminated.

5 Bisimulations

In this section, bisimulation relations are defined both on the symbolic model and the concrete model. We use bisimulation as our correctness criteria. At the end of this section, the relation between the bisimulation on PSA and the bisimulation on both its semantics is stated.

The definition of the symbolic bisimulation is a straightforward modification of probabilistic bisimulation [26] in order to fit clocks.

Definition 5. *Given a PSA, a relation $R \subseteq St \times St$ is a* (symbolic) bisimulation *on PSA if the following statements hold:*

1. *R is an equivalence relation, and*
2. *whenever $\langle s_1, s_2 \rangle \in R$ and $s_1 \xrightarrow{C',a} \rho_1$, there is a probability space ρ_2 such that*
   (a) $s_2 \xrightarrow{C',a} \rho_2$ and
   (b) $\rho_1(S) = \rho_2(S)$ for every equivalence class $S \in (Ck \times St)/R_{Ck}$ induced by the relation $R_{Ck} \stackrel{\text{def}}{=} \{\langle (C, s_1), (C, s_2) \rangle \mid C \subseteq Ck, \langle s_1, s_2 \rangle \in R\}$.

Two control states s_1 and s_2 are bisimilar*, notation $s_1 \sim s_2$, if there is a bisimulation R such that $\langle s_1, s_2 \rangle \in R$. Two PSA, PSA_1 and PSA_2 are* bisimilar*, notation $\mathsf{PSA}_1 \sim \mathsf{PSA}_2$, if $C_0^1 = C_0^2$ and $s_0^1 \sim s_0^2$ in the disjoint union of PSA_1 and PSA_2.*

The definition of the concrete bisimulation is a little more involved since it must deal with continuous probability spaces (see e.g. [36, 11, 4]). In particular, we follow the definition given in [4]. We first give some necessary definitions.

Let $(\Omega, \mathcal{F}, \mu)$ and $(\Omega', \mathcal{F}', \mu')$ be two probability spaces. We say that they are *equivalent* (notation $(\Omega, \mathcal{F}, \mu) \approx (\Omega', \mathcal{F}', \mu')$) if (a) for all $A \in \mathcal{F}$, $A \cap \Omega' \in \mathcal{F}'$ and $\mu(A) = \mu'(A \cap \Omega')$, and (b) for all $A' \in \mathcal{F}'$, $A' \cap \Omega \in \mathcal{F}$ and $\mu'(A') = \mu(A' \cap \Omega)$.

Given an equivalence relation R on a set Σ and a set $I \subseteq \Sigma$, we define the function $EC_{I,R} : I \to \Sigma/R$ which maps each state $\sigma \in I$ into the corresponding equivalence class $[\sigma]_R$ in Σ.

Definition 6. *Given a PTTS, a relation $R \subseteq \Sigma \times \Sigma$ is a* bisimulation *on PTTS if the following statements hold:*

1. *R is an equivalence relation, and*
2. *whenever $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\sigma_1 \xrightarrow{a} \pi_1$, there is a probability space π_2 such that*
   (a) $\sigma_2 \xrightarrow{a} \pi_2$ and
   (b) $EC_{\Sigma_1,R}(\pi_1) \approx EC_{\Sigma_2,R}(\pi_2)$ where Σ_i is the sample space of π_i.

Two states σ_1 and σ_2 are bisimilar*, notation $\sigma_1 \sim \sigma_2$, if there is a bisimulation R such that $\langle \sigma_1, \sigma_2 \rangle \in R$. Two PTTS, PTTS_1 and PTTS_2 are* bisimilar*, notation $\mathsf{PTTS}_1 \sim \mathsf{PTTS}_2$, if $EC_{\Sigma_1,\sim}(\pi_0^1) \approx EC_{\Sigma_2,\sim}(\pi_0^2)$ in the disjoint union of PTTS_1 and PTTS_2.*

Symbolic bisimulation on PSAs is preserved by both residual lifetime semantics and spent lifetime semantics in the following sense.

Theorem 1. *Given two PSA, PSA_1 and PSA_2 such that $\mathsf{PSA}_1 \sim \mathsf{PSA}_2$, then $[\![\mathsf{PSA}_1]\!]_r \sim [\![\mathsf{PSA}_2]\!]_r$ and $[\![\mathsf{PSA}_1]\!]_s \sim [\![\mathsf{PSA}_2]\!]_s$.*

Proof (Sketch). It is routine to prove that if R is a symbolic bisimulation, then $\{\langle (s_1, v, e), (s_2, v, e) \rangle \mid \langle s_1, s_2 \rangle \in R, v \in \mathit{TVal}, e \in \mathit{TVal}\}$ is a bisimulation in the residual lifetime semantics, and similarly, changing *TVal* by *Val*, for the spent lifetime semantics. $\square$

6 Model Embeddings

In this section, three different stochastic formal models are encoded into PSA. Therefore, PSA is a reasonable framework that captures the characteristics of each one of them, and hence, this hints that those models are not so distant in expressiveness. Adequacy of these encodings are also given.

6.1 IGSMP and PSA

In [8, 4], an extension of generalized semi-Markov process [16] is presented in order to allow for compositional description of concurrent systems.

It considers three different types of transitions: standard action transitions, representing action execution, clock start transitions, representing the event in which a clock becomes active, and clock termination transition, representing the event of clock termination. This is different from PSA in which all these ingredients are integrated in only one control transition

Clock start transitions allow for preselection policy, therefore they are also labelled with a weight. They have the form $s_1 \xrightarrow{<c,w>} s_2$ where c is a clock name and $w \in \mathbb{R}_{>0}$ is the weight of the transition. Therefore, if there is another clock start transition $s_1 \xrightarrow{<c',w'>} s_3$, the probability of taking the first one is $\frac{w}{w+w'}$. When $s_1 \xrightarrow{<c,w>} s_2$ is performed, the clock c starts and continues its execution in every state traversed by the IGSMP. Whenever the clock c terminates, the IGSMP executes a termination transition of the form $s'_1 \xrightarrow{c^-} s'_2$. In particular, since each active clock c must continue its execution in each state traversed by the IGSMP, all such states must have an outgoing c^- transition. Like in PSA, simultaneously active clocks in an IGSMP must have different names so that the event of termination of a clock c^- is always related to a start event $<c,w>$ of the same clock (for some w). IGSMP treats this in a particular manner: clock names have the form $\langle f, i\rangle$ where f is its distribution and $i \in \mathbb{N}$ is a number to differentiate from other clocks $\langle f, j\rangle$ with the same distribution function. Besides, this natural number is used to dynamically name clocks in an ordered fashion.

An IGSMP has four different kind of states:

- *silent states*, enabling invisible action transitions τ and (possibly) visible action transitions only. In such states, the IGSMP performs a non-deterministic choice among the τ transitions and may interact with the environment through one of the visible actions but it is not allowed to idle.
- *probabilistic states*, enabling $<c,w>$ transitions and (possibly) visible action transitions only. In such states, the IGSMP performs a probabilistic choice among the clock start transitions and may interact with the environment through one of the visible actions but it is not allowed to idle.
- *timed states*, enabling c^- transitions and (possibly) visible action transitions only. In such states, the IGSMP is allowed to idle as long as no active clock terminates. The clock that terminates first determines the transition to be performed. Note that IGSMP is defined under the assumption that clocks cannot terminate at the same instant, therefore, only one clock terminates before the other ones. While the IGSMP sojourns in these states, it may interact with the environment through one of the outgoing visible action transitions.
- *waiting states* enabling standard visible action transitions only or no transition at all. In these states, the IGSMP sojourns indefinitely or, at any time, it may interact with the environment through one of the outgoing visible action transitions.

Formally, an IGSMP is defined as follows.

Definition 7. *An Interactive Generalized Semi-Markov Process is a structure* $\mathsf{IGSMP} = (St, Ck, Distr, Act_d \cup \{\tau\}, \longrightarrow, s_0)$ *where* St, s_0, Act_d, *and* τ *are as in Def. 1, and*

- $Ck = PDF \times I\!N$ *is the set of* clock names*;*
- $Distr : Ck \rightarrow PDF$ *assigns a probability distribution function to each clock such that for all* $\langle f, i\rangle \in Ck$, $Distr(\langle f, i\rangle) = f$*; and*
- $\longrightarrow \subseteq St \times (Ck^+ \cup Ck^- \cup Act_d \cup \{\tau\}) \times St$ *is the* control transition *relation, where*
  - $Ck^+ = Ck \times I\!R_{>0}$ *is the set of events indicating the start of a clock; for* $<c, w> \in Ck^+$, w *gives the* weight *that determines the probability of starting clock* c*; and*
  - $Ck^- = \{c^- \mid c \in Ck\}$ *is the set of events denoting the termination of a clock;*

  and satisfies

  1. $\forall s \in St.\ s \xrightarrow{\tau} \implies \forall \theta \in Ck^+ \cup Ck^- : s \stackrel{\theta}{\not\longrightarrow}$
  2. $\forall s \in St.\ (\exists \theta \in Ck^+.\ s \xrightarrow{\theta}) \implies \forall c^- \in Ck^-.\ s \stackrel{c^-}{\not\longrightarrow}$
  3. *exists* $\mathcal{S} : St \rightarrow 2^{Ck}$, *the* active clock function , *such that for all* $s \in St$,
     (a) $\mathcal{S}(s_0) = \emptyset$
     (b) $\cdot\ \forall a \in Act_d \cup \{\tau\}.\ s \xrightarrow{a} s' \implies \mathcal{S}(s') = \mathcal{S}(s)$
     $\cdot\ \forall <c, w> \in Ck^+.\ s \xrightarrow{<c,w>} s' \implies \mathcal{S}(s') = \mathcal{S}(s) \cup \{c\}$
     $\cdot\ \forall c^- \in Ck^-.\ s \xrightarrow{c^-} s' \implies c^- \in \mathcal{S}(s) \wedge \mathcal{S}(s') = \mathcal{S}(s) - \{c\}$
     (c) $\forall <\langle f, i\rangle, w> \in Ck^+.$
     $$s \xrightarrow{<\langle f,i\rangle,w>} \implies i = \min\{j \mid j \in I\!N, \langle f, j\rangle \in \mathcal{S}(s)\}$$
     (d) $c \in \mathcal{S}(s) \wedge s \stackrel{\tau}{\not\longrightarrow} \wedge (\forall \theta \in Ck^+.\ s \stackrel{\theta}{\not\longrightarrow}) \implies s \xrightarrow{c^-}$
  4. $\forall s \in St.\ (\exists \theta \in Ck^+.\ s \xrightarrow{\theta} s') \implies \mathrm{act}(s') \subseteq \mathrm{act}(s)$ *(with* $\mathrm{act}(s) = \{a \in Act_d \cup \{\tau\} \mid s \xrightarrow{a}\}$*)*

The constraints over the transitions guarantee that each state in IGSMP belongs to one of the four kind of states mentioned above. In particular, the first requirement says that if a state can perform τ actions, it cannot perform clock starts or clock terminations. Such a property derives from the assumption of *maximal progress*. The second requirement, says that if a state can perform clock start events then it cannot perform clock termination events. Such a property derives from the assumption of *urgency of delays*: clock start events cannot be delayed but must be performed immediately, hence they prevent the execution of clock termination transitions. The third requirement checks that clock starting and termination transitions are consistent with the set of clocks that should be active in each state. This is done by defining a function $\mathcal{S}$ that maps each state onto the expected set of active clocks. In particular, such a set is empty in the initial state. The fourth requirement implements the following constraint: The unique role of clock start transitions in an IGSMP must be to lead to a time state where the started clocks are actually executed; therefore, the execution of such transitions cannot cause new behaviours to be performable by the IGSMP.

The semantics of IGSMPs has been defined in [4] in terms of the so called *interactive stochastic timed transition systems (ISTTS* following the spent lifetime model. Basically, ISTTSs are a particular form of PTTSs. We redefine IGSMPs semantics in terms of PTTS without altering the original definition[3].

Recall that IGSMP is defined under the assumption that clocks cannot terminate at the same time. That is, the probability that two different clocks take the same value is 0. Let $\mathit{Term}_k = \{(t_1, \ldots, t_k) \mid \exists i, j.1 \leq i < j \leq k \wedge t_i = t_j\}$ and $\mathcal{R}(f_1, \ldots, f_k) = (\mathbb{R}^k, \mathcal{F}, P)$. Define $\breve{\mathcal{R}}(f_1, \ldots, f_k)$ to be the probability space $(\breve{\mathbb{R}}^k, \breve{\mathcal{F}}, \breve{P})$ where,

1. $\breve{\mathbb{R}}^k = \mathbb{R}^k - \mathit{Term}_k$,
2. $\breve{\mathcal{F}} = \{E \subseteq \breve{\mathbb{R}} \mid E \subseteq \mathcal{F}\}$, and
3. $\breve{P}(E) = P(E)$ for all $E \in \breve{\mathcal{F}}$.

Definition 8. *The semantics of* IGSMP *is defined by the PTTS* $[\![\mathsf{IGSMP}]\!] = (\Sigma, \mathit{Act} \cup \mathbb{R}_{\geq 0}, \longrightarrow, \pi_0)$ *where:*

- $\Sigma \stackrel{\text{def}}{=} \mathit{St} \times \mathit{Val} \times (\{-\} \cup (\mathit{Ck} \times \mathbb{R}_{>0}))$,
- $\mathit{Act} = \mathit{Act}_d \cup \mathit{Act}_u$ *with* $\mathit{Act}_u = \{\tau, \star\}$,
- $\pi_0(s_0, \emptyset, -) = 1$, *and*
- $\longrightarrow$ *is defined according to the rules in Table 1.*

In addition to the control state s, a state $(s, v, (c, t))$ contains (i) the set of active clocks together with its spent lifetimes (represented by the partial valuation v), and (ii) a pair (c, t) containing the time value sampled by the winning clock and the name of this clock. The latter field is set to "$-$" whenever active clocks of the IGSMP still have to be sampled. The sampling $(s, v, -) \xrightarrow{\star} P(s, v, -)$ leads to states where starting clocks are associated to a spent lifetime 0, active clocks in $\mathrm{dom}(v)$ (except those that were re-started) preserve their value, and the winning clock and its sampled value are indicated. All the work lies on function P which aggregates probabilities of the preselection policy on clock start transitions together with the sampling of the values of all active clocks. To define P two auxiliary functions are required. Function $TW : \mathit{St} \to \mathbb{R}_{>0}$ in Table 1 computes the overall weight of the clock start transitions leaving a state of an IGSMP. Function sample_v^s maps a tuple $(t_1, \ldots, t_k)$ of time values sampled by active clocks in $\mathrm{dom}(v)$ into the corresponding state $(s, v, (c_j, t_j))$ where j is the index of the clock which sampled with the least value.

Notice that, for all states (s, v, e) of any interpretation $[\![\mathsf{IGSMP}]\!]$ the following statements hold,

1. $(s, v, e) \xrightarrow{\tau}$ implies $(s, v, e) \stackrel{\star}{\not\longrightarrow}$ and $(s, v, e) \stackrel{t}{\not\longrightarrow}$ for any $t \in \mathbb{R}_{\geq 0}$;
2. $(s, v, e) \xrightarrow{\star}$ implies $(s, v, e) \stackrel{t}{\not\longrightarrow}$ for any $t \in \mathbb{R}_{\geq 0}$;
3. Either $(s, v, e) \xrightarrow{\tau}$, $(s, v, e) \xrightarrow{\star}$, or there is a $t \in \mathbb{R}_{>0}$ such that $(s, v, e) \xrightarrow{t}$

[3] It is straightforward to see that the new semantics preserves bisimulation with respect to the original one in terms of ISTTS.

Table 1. Semantics of IGSMPs

$$\frac{\exists \theta \in Ck^- \cup Ck^+ .\ s \xrightarrow{\theta}}{(s, v, -) \xrightarrow{\star} P(s, v, -)}$$

where P is defined by the following rules

$$\frac{(\exists c^- \in Ck^- .\ s \xrightarrow{c^-}) \qquad \mathrm{dom}(v) = \{c_1, \ldots, c_k\}}{P(s, v, -) \stackrel{\mathrm{def}}{=} \mathit{sample}^s_v(\breve{\mathcal{R}}([\mathit{Distr}(c_1) \mid v(c_1)], \ldots, [\mathit{Distr}(c_k) \mid v(c_k)]))}$$

$$\frac{(\exists <c, w> \in Ck^+ .\ s \xrightarrow{<c,w>}) \qquad Pr = \{(\langle c, s' \rangle, w/TW(s)) \mid s \xrightarrow{c,w} s'\}}{P(s, v, -) \stackrel{\mathrm{def}}{=} \sum \{w/TW(s) \cdot P(s', v \cup \{(c, 0)\}, -) \mid s \xrightarrow{<c,w>} s'\}}$$

with

- $TW(s) = \sum_{s \xrightarrow{<c,w>} s'} w$
- $\mathit{sample}^s_v(t_1, \ldots, t_k) = (s, v, (c_j, t_j))$
  provided $t_j = \min\{t_1, \ldots, t_k\}$ and $\mathrm{dom}(v) = \{c_1, \ldots, c_k\}$

$$\frac{0 \le t' < t}{(s, v, (c, t)) \xrightarrow{t'} (s, v + t', -)} \qquad \frac{s \xrightarrow{c^-} s' \quad c^- \in Ck^-}{(s, v, (c, t)) \xrightarrow{t} (s, (v - \{c\}) + t, -)}$$

$$\frac{t \ge 0 \quad \forall \theta \in Ck^- \cup Ck^+ .\ s \stackrel{\theta}{\not\longrightarrow} \quad s \stackrel{\tau}{\not\longrightarrow}}{(s, \emptyset, -) \xrightarrow{t} (s, \emptyset, -)}$$

$$\frac{s \xrightarrow{a} s' \quad a \in Act_d \cup \{\tau\}}{(s, v, -) \xrightarrow{a} (s', v, -)} \qquad \frac{s \xrightarrow{a} s' \quad a \in Act_d}{(s, v, (c, t)) \xrightarrow{a} (s', v, -)}$$

ISTTS considers separately probabilistic transitions and non-deterministic transitions (the latter are used both for action transitions and time transitions). We tried to preserve this characteristic. Observe in Table 1 that only transitions $(s, v, -) \xrightarrow{\star} P(s, v, -)$ are not trivial which would correspond to the probabilistic transition of ISTTS.

An IGSMP can be encoded in terms of a PSA as follows.

Definition 9. *Let* $\mathsf{IGSMP} = (St, Ck, Distr, Act_d \cup \{\tau\}, \longrightarrow, s_0)$. *Its interpretation in terms of PSA is given by* $\mathsf{PSA}(\mathsf{IGSMP}) \stackrel{\mathrm{def}}{=} (St, Ck, Distr, Act, \longrightarrow, s_0, \emptyset)$ *where* $Act = Act_d \cup Act_u$; $Act_u = \{\tau, -\} \cup \{\bar{w} | w \in I\!R_{>0}\}$ *with* $\prec$ *the least priority relation satisfying* $\alpha \prec \tau$ *for every* $\alpha \in Act_u - \{\tau\}$ *and* $- \prec \bar{w}$ *for every* $w \in I\!R_{>0}$;

and $\longrightarrow$ is defined by the following rules:

$$\frac{s \xrightarrow{a} s' \quad a \in Act_d \cup \{\tau\}}{s \xrightarrow{\emptyset, a, \emptyset} s'} \qquad\qquad \frac{s \xrightarrow{c^-} s'}{s \xrightarrow{\{c\}, -, \emptyset} s'}$$

$$\frac{\exists <c,w> \in Ck^+.\ s \xrightarrow{<c,w>} \qquad \rho(C, s') = \textbf{if}\ (C = \{c\})\ \textbf{then}\ \sum\{w/TW(s) \mid s \xrightarrow{<c,w>} s'\}\ \textbf{else}\ 0}{s \xrightarrow{\emptyset, \overline{TW(s)}} \rho}$$

where TW is as in Table 1.

The encoding is quite simple: each of the three type of transitions in IGSMP is encoded in a PSA control transition containing the only ingredient it represents. However, clock start transitions deserve particular attention. Since weights determine a probabilistic jumps, all clock start transitions emanating from one state should be encoded in a unique probabilistic transition. Thus, for instance, if $s_0 \xrightarrow{<c_1,1>} s_1$ and $s_0 \xrightarrow{<c_2,2>} s_2$ are the only two clock start transitions leaving s_0, then $s_0 \xrightarrow{\emptyset, \overline{3}} \{\langle c_1, s_1\rangle \mapsto \frac{1}{3}, \langle c_2, s_2\rangle \mapsto \frac{2}{3}\}$. Label $\overline{3}$ is kept for compositional matters. (Weights do not behave in the same way as probability values.) This will become more apparent in Section 7.1.

The following theorem states the adequacy of the translation of IGSMPs into PSAs. The notion of bisimulation on IGSMPs has been defined in [4].

Theorem 2. *Given* IGSMP_1 *and* IGSMP_2 *the following statements hold:*

1. $\mathsf{IGSMP}_1 \sim \mathsf{IGSMP}_2$ *if and only if* $\mathsf{PSA}(\mathsf{IGSMP}_1) \sim \mathsf{PSA}(\mathsf{IGSMP}_2)$.
2. $[\![\mathsf{PSA}(\mathsf{IGSMP}_1)]\!]_s \sim [\![\mathsf{PSA}(\mathsf{IGSMP}_2)]\!]_s$ *implies* $[\![\mathsf{IGSMP}_1]\!] \sim [\![\mathsf{IGSMP}_2]\!]$.

Proof (Sketch).

1. It is routine to prove that the same relation R is a bisimulation on IGSMPs if and only if it is a bisimulation on their translation.
2. Let R be a bisimulation relation such that $[\![\mathsf{PSA}(\mathsf{IGSMP}_1)]\!]_s \sim [\![\mathsf{PSA}(\mathsf{IGSMP}_2)]\!]_s$ Define R' by

$$\begin{aligned} R' \stackrel{\text{def}}{=} & \{\langle (s_1, v_1, -), (s_2, v_2, -)\rangle \mid \langle (s_1, v_1, e), (s_2, v_2, e')\rangle \in R\} \\ & \cup \{\langle (s_1, v_1, (c_1, t_1)), (s_2, v_2, (c_2, t_2))\rangle \mid \\ & \qquad \langle (s_1, v_1, \{c_1 \mapsto t_1\}), (s_2, v_2, \{c_2 \mapsto t_2\})\rangle \in R \\ & \qquad \wedge \forall i \in \{1,2\}.\ \forall c, c' \in \mathrm{dom}(v_i).\ v_i(c) = v_i(c') \implies c = c'\} \end{aligned}$$

It can be proved that R' is a bisimulation. □

Notice that, although bisimulation is preserved forth and back at symbolic level, these is not the case at concrete level. The direct interpretation of IGSMP is weaker in the sense that more states are equated. In fact, the semantics of IGSMP makes an aggregation of the sampling transitions (those labelled with $\star$) while this is not present in the semantics of PSA.

6.2 Stochastic Automata and PSA

Another extension to generalized semi-Markov process which allows for compositional description of concurrent systems was introduced in [14, 11]. Rather than splitting actions, starting clocks, and terminating clocks in three different transitions, this model includes this three ingredients in only one symbolic transition just like PSA.

A *stochastic automata* (SA) [14, 11] is a PSA $(St, Ck, Distr, Act, \longrightarrow, s_0, C_0)$ such that for all $s \xrightarrow{C,a} \rho$, ρ is a trivial distribution function. Following nomenclature in [14, 11], if $Act_u = \emptyset$, we say that the SA is *open*, that is, it represents a system that cooperates with the environment or is intended to be part of a larger system. If $Act_d = \emptyset$, the SA is *closed*, i.e., it represents a system that is complete by itself and no external interaction is required. In this last case there is a flat order in Act_u (i.e., actions in Act_u cannot be compared with each other).

As a matter of fact, in its original definition, a control transition in a SA had the form $s \xrightarrow{C,a} s'$ and there was a clock resetting function κ that took a control state and returned the clocks that should be started at the moment of reaching the state. The translation to this setting is straightforwardly given by $s \xrightarrow{C,a,\kappa(s')} s'$. We stick to the definition of SA given above since the difference does not give any sensible insight.

6.3 Stochastic Bundle Event Structures and PSA

Stochastic bundle event structures (SBES) were introduced in [9, 23]. They present a true concurrency framework rather than an interleaving one like PSA, IGSMP, or SA. We also present an encoding of this model in terms of PSA.

We briefly recall the definition of SBESs.

Definition 10. *A* bundle event structure (BES) *is a structure* $(E, \leadsto, \rightarrowtail, l, Act)$ *where* E *is a set of* events, $\leadsto \subseteq E \times E$ *is the* asymmetric conflict *relation,* $\rightarrowtail \subseteq 2^E \times E$ *is the* bundle *relation, and* $l : E \to Act$ *is the* action-labelling *function, such that* $\leadsto$ *is irreflexive, and for all* $X \subseteq E$, $e \in E$, *if* $X \rightarrowtail e$ *then for all* $e', e'' \in X$, $e' \neq e''$ *implies* $e' \leadsto e''$.

A stochastic bundle event structure (SBES) *is a triple* $\langle \mathcal{E}, \mathcal{F}, \mathcal{G} \rangle$ *where* $\mathcal{E}$ *is a bundle event structure* $(E, \leadsto, \rightarrowtail, l, Act)$, *and* $\mathcal{F} : E \to PDF$ *and* $\mathcal{G} : \rightarrowtail \to PDF$ *are two functions that associate distribution functions to events and bundles, respectively.*

Bundle event structures are variation of event structures [37], a well known causal based model and inherently different to the kind of model we have seen so far. Bundles indicate cause: if $X \rightarrowtail e$, one event of X *must* occur before e. Moreover, e can only occur if exactly one event of each of its bundles have already occurred. Besides, there are events whose occurrence prevent the occurrence of other events. This is indicated by the asymmetric conflict relation: if $e \leadsto e'$, then e cannot occur after e'.

A SBES is a BES decorated with stochastic information indicating the time in which actions are allowed to occur. Function $\mathcal{F}$ indicates that the occurrence time of an event e since the beginning of the execution is distributed according to $\mathcal{F}(e)$. $\mathcal{G}(X \rightarrowtail e)$ is the distribution of the time elapsed between the occurrence of X's only executed event and e.

In a causal based model such as event structures, executions are not represented by a total (linear) order such as a trace or a sequence of transitions (as it is the case in automata based models). In order to represent the independence of occurrence of concurrent events, the execution is defined in terms of partial orders. A partial execution is called a configuration and is defined as follows.

Definition 11. *Given a BES, a* configuration *of it is a set* $\mathrm{Cf} \subseteq E$ *such that there is a strict total order* $\prec$ *in* Cf *where for all* $e \in \mathrm{Cf}$,

1. $X \rightarrowtail e$ *implies* $X \cap \{e' \in \mathrm{Cf} \mid e' \prec e\} \neq \emptyset$ *(intuitively: every bundle that causes e was already visited), and*
2. *for all* $e' \prec e$, $e \rightsquigarrow e'$ *does not hold, i.e. e is not in conflict with its predecessors.*

$\prec$ *is an order that* determines that Cf is a configuration*.*

We say that a configuration Cf is right before *a configuration* $\mathrm{Cf} \cup \{e\}$ *(*$e \notin \mathrm{Cf}$*) if whenever* $\prec$ *is an order that determines that* Cf *is a configuration,* $\prec \cup (\mathrm{Cf} \times \{e\})$ *determines that* $\mathrm{Cf} \cup \{e\}$ *is a configuration.*

The notion of "right before" makes it possible to obtain a notion of transition: $\mathrm{Cf} \xrightarrow{l(e)} \mathrm{Cf} \cup \{e\}$ if Cf is right before $\mathrm{Cf} \cup \{e\}$. This notion is central to the translation of SBES into PSA.

Definition 12. *Given* $\mathsf{SBES} = \langle \mathcal{E}, \mathcal{F}, \mathcal{G} \rangle$ *with* $\mathcal{E} = (E, \rightsquigarrow, \rightarrowtail, l, Act)$*, its interpretation in terms of PSA is given by* $\mathsf{PSA}(\mathsf{SBES}) \stackrel{\text{def}}{=} (St, Ck, Distr, Act, \longrightarrow, s_0, C_0)$ *where*

- *St is the set of all configurations in* $\mathcal{E}$*, with* $s_0 = \emptyset$*, the empty configuration;*
- $Ck = E \cup \rightarrowtail$ *with* $C_0 = E$*;*
- $Distr = \mathcal{F} \cup \mathcal{G}$*;*
- $\longrightarrow$ *is the least relation satisfying*

$$\frac{\mathrm{Cf} \textit{ is right before } \mathrm{Cf} \cup \{e\} \qquad C_g = \{e\} \cup \{(X, e') \in \rightarrowtail \mid e' = e\} \qquad C_s = \{(X, e') \in \rightarrowtail \mid e \in X\}}{\mathrm{Cf} \xrightarrow{C_g, l(e), C_s} \mathrm{Cf} \cup \{e\}}$$

If $Act_d = Act$ *and* $Act_u = \emptyset$ *we say that the interpretation* $\mathsf{PSA}(\mathsf{SBES})$ *is* open*; if* $Act_d = \emptyset$ *and* $Act_u = Act$*, we say it is* closed*.*

The translation is rather simple. Since in a SBES timing is associated to bundles (with $\mathcal{G}$) and to events (with $\mathcal{F}$), we take $Ck = E \cup \rightarrowtail$. $\mathcal{F}(e)$ associates the distribution of the time elapsed since the system starts. Therefore all "clocks

$e \in E$" are started at initialization, that is, $C_0 = E$. "Clocks in $\rightarrowtail$" indicates time between two control transitions, then they are set on some control transition. The control transition is obtained from the same transition of the BES Cf $\xrightarrow{l(e)}$ Cf $\cup \{e\}$ which is decorated with the stochastic information obtained from the SBES. Therefore C_g contains the clocks that enable event e, that is all bundles that cause it. In addition it contains "clock e" which was set at initialization. On executing this transition, all bundles caused by e must start counting time. Then $(X, e') \in\rightarrowtail$ will start if and only if $e \in X$. Notice that $\mathsf{PSA}(\mathsf{SBES})$ is infinite if SBES is infinite.

Unfortunately, no equivalence relation is actually provided for SBES. We could imagine that equivalences on event structure can be lifted up to SBES. In this case, the translation $\mathsf{PSA}(\mathsf{SBES})$ would only be adequate for interleaving based equivalences such as bisimulation relations, since any causal information is lost in the translation.

7 Semantics of Stochastic Process Algebras

Stochastic process algebras are extensions of traditional process algebras [21, 31, 2] with some mean of representing stochastic time delay and occasionally probabilistic jumps. The syntax of a classic (non-stochastic) process algebra is defined by the following grammar:

$$P ::= \ \mathbf{0} \ | \ X \ | \ a.P \ | \ P+P \ | \ P/L \ | \ P[\phi] \ | \ P \,||_S\, P \ | \ rec\,X.P \qquad (6)$$

where $a \in Act$ is an *action name*, X is a process variable, $L, S \subseteq Act - \{\tau\}$, $\phi : (Act - \{\tau\}) \to (Act - \{\tau\})$.

Intuitively, their interpretation is as follows. $\mathbf{0}$ is a process that does not do anything. The *prefix* $a.P$ first performs the action a and then behaves as P. The *choice* $P + Q$ provides the possibility of executing one out of two possible behaviours P and Q. Though usually the choice is resolved non-deterministically, it can also be resolved depending on the stochastic information, which very much depends on the language choices of every stochastic process algebra. P/L behaves like P except that actions in L are hidden to the environment. $P[\phi]$ behaves like process P but actions are renamed according to ϕ. $P \,||_S\, Q$ defines the *parallel composition*. It describes a process that executes P and Q in parallel forcing synchronization of actions in the set S. Other actions can be executed independently of the partner process. $rec\,X.P$ defines the recursion on the variable X in the usual way.

In the remaining of this section we give semantics to several stochastic process algebras in terms of PSA and show that their semantics are equivalent to the originally given in terms of their original models.

7.1 The calculus of IGSMP and PSA

The calculus for IGSMP [8, 4] extends traditional process algebra with a prefix operation that makes it possible to represent stochastic time delay and probabilistic jump. Its full syntax is given by adding the *delay prefix* $<f, w>.P$ to that of the classic calculus (6), where $w \in \mathbb{R}_{>0}$ is a weight, and $f \in \mathit{PDF}$ is a distribution function. Given $<\!f, w\!>.P$, w determines the probability of actually executing this process (this probability depends on the context), and f determines the probability of the waiting time before executing P in case this process has been selected to be executed. Therefore, in a process like $<\!f, 1\!>.P + <\!g, 2\!>.Q$, one third of the times the system waits a random time depending on f and then behaves like P, and the other 2/3, it waits for a random time according to g and then behaves like Q.

IGSMP semantics has to make possible the distinction of, e.g., the time event on the left-hand side of $||$ from the one in the right-hand side in process $<\!f, 1\!>.\mathbf{0} \,||\, <\!f, 1\!>.\mathbf{0}$. The most problematic part is to keep the relation between start and termination events (i.e. the problem of expressing ST semantics, see Section 2.5). To do so IGSMP semantics uses a dynamic technique to name clocks (well-naming rule). When a new f-distributed time event appears, a fresh name $\langle f, i\rangle$ is generated. $i \in \mathbb{N}$ is the least index not yet used by other active delays with distribution f. Since start events and termination events are represented in different control transitions, IGSMP requires an additional operator $f_i^-.P$, which is associated to the termination of a clock meaning that clock $\langle f, i\rangle$ should terminate before executing P.

Since the problem of clock naming occurs because of parallel composition, to define IGSMP semantics, it needs an additional parameter: $P \,||_{S,M}\, Q$ extends the parallel composition with a set $M \subseteq \mathit{Ck} \times (\{r_i \mid i \in \mathbb{N}\} \cup \{l_i \mid i \in \mathbb{N}\})$. M records the association between the name $\langle f, i\rangle$, generated according to the well naming rule for identifying f at the level of $P \,||_{S,M}\, Q$, and the name $\langle f, j\rangle$, generated according to the well naming rule for identifying f at the level of P (or Q). In this way, when afterwards such a delay f terminates in P (or Q), the name $\langle f, j\rangle$ can be remapped to the correct name $\langle f, i\rangle$ at the level of $P \,||_{S,M}\, Q$ by using the information recorded in M. More precisely, in a tuple $(\langle f, i\rangle, l_j)$, M records that the event $\langle f, i\rangle$ in $P \,||_{S,M}\, Q$ is actually named $\langle f, j\rangle$ in P ("l" stands for *left*). In a tuple $(\langle f, i\rangle, r_j)$, M records that the event $\langle f, i\rangle$ comes from an event named $\langle f, j\rangle$ in Q ("r" is for *right*). In this context, $P \,||_S\, Q$ is defined to be $P \,||_{S,\emptyset}\, Q$.

Let IGSMP_{sg} be the set of all strongly guarded processes defined with this new operations $f_1^-.P$ and $P \,||_{S,M}\, Q$.

The semantics of this calculus in terms of IGSMP is given in [8, 4]. Its semantics in terms of PSA is as follows.

Definition 13. *Let $\longrightarrow$ be the least relation satisfying rules in Tables 2, 3, and 4. The* interpretation *of an IGSMP process P is given by* $\mathsf{PSA}(P) \stackrel{\text{def}}{=} (\mathit{St}_P, \mathit{Ck}, \mathit{Distr}, \mathit{Act}, \longrightarrow, P, \emptyset)$ *where*

- *Ck, Distr, Act_d and Act_u are as in Definitions 7 and 9, and*
- *St_P is the subset of IGSMP_{sg} such that (a) $P \in \mathit{St}_P$, and (b) if $Q \in \mathit{St}_P$, $Q \xrightarrow{C,a} \rho$ and $\rho(C', Q') > 0$, then $Q' \in \mathit{St}_P$.*

Rules in Table 2 are standard in process algebra. Rules in Table 3 define the clock start transitions. Notice that τ transitions are taken into account in

$a.P \xrightarrow{\emptyset,a,\emptyset} P$	$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P' \quad a \in L}{P/L \xrightarrow{\emptyset,\tau,\emptyset} P'/L}$	$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P'}{P[\phi] \xrightarrow{\emptyset,\phi(a),\emptyset} P'[\phi]}$
$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P'}{\begin{array}{c} P+Q \xrightarrow{\emptyset,a,\emptyset} P' \\ Q+P \xrightarrow{\emptyset,a,\emptyset} P' \end{array}}$	$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P' \quad a \notin L}{P/L \xrightarrow{\emptyset,a,\emptyset} P'/L}$	$\dfrac{P\{rec\,X.P/X\} \xrightarrow{\emptyset,a,\emptyset} P'}{rec\,X.P \xrightarrow{\emptyset,a,\emptyset} P'}$
$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P' \quad a \notin S}{\begin{array}{c} P \,\|_{S,M}\, Q \xrightarrow{\emptyset,a,\emptyset} P' \,\|_{S,M}\, Q \\ Q \,\|_{S,M}\, P \xrightarrow{\emptyset,a,\emptyset} Q \,\|_{S,M}\, P' \end{array}}$	$\dfrac{P \xrightarrow{\emptyset,a,\emptyset} P' \quad Q \xrightarrow{\emptyset,a,\emptyset} Q' \quad a \in S}{P \,\|_{S,M}\, Q \xrightarrow{\emptyset,a,\emptyset} P' \,\|_{S,M}\, Q'}$	

Table 2. Standard rules for the IGSMP calculus ($a \in Act \cup \{\tau\}$)

rules for summation and parallel composition in order to ensure their priority over delays. A similar consideration is taken in the rule of hiding for actions in set L. Since weights define a probabilistic choice, clock start transitions need to be combined appropriately in a summation or a parallel composition. Suppose the left operand is willing to perform a clock start transition with weight w and the right operand is willing to perform a clock start transition with weight w'. Then, the left-hand side processes will be performed with probability $\frac{w}{w+w'}$ and the right ones, with probability $\frac{w'}{w+w'}$. These factors are henceforth used to construct the new distribution. Besides, the new transition carries the weight of both operands, that is $w + w'$. Finally, we address the attention to auxiliary function $(\rho \,\|^{P,Q}_{S,M}\, \rho')$. Apart from appropriately distributing measures, it has the duty to extend set M. Notice that M is extended with $(\langle f, n(M_f)\rangle, l_i)$ (or $(\langle f, n(M_f)\rangle, r_i)$) if the left (or right) process performs $\langle f, i\rangle$; the term $n(M_f)$ is in charge of choosing the least $j \in \mathbb{N}$ such that $\langle f, j\rangle$ is not yet used in M.

Rules in Table 4 define the clock termination transitions. They also take into account the priority of τ transitions over delays but, in addition, they take into account the priority of clock start over clock terminations. In particular, notice the rules for $P \,\|_{S,M}\, Q$. When P terminates clock $\langle f, i\rangle$, clock $\langle f, j\rangle$ associated to l_i in M terminates at the level of the parallel composition (and hence eliminated from M). A similar mechanism takes place if Q terminates clock $\langle f, i\rangle$.

The following theorem states that the semantic of the IGSMP calculus in terms of PSA is equivalent to the original semantics in terms of IGSMPs.

Theorem 3. *For any IGSMP process P,* $\mathsf{PSA}(P) \sim \mathsf{PSA}(\mathsf{IGSMP}(P))$*, where* $\mathsf{IGSMP}(P)$ *is the IGSMP semantics of P as defined in [4].*

Proof (Sketch). More precisely, $\mathsf{PSA}(P)$ and $\mathsf{PSA}(\mathsf{IGSMP}(P))$ are identical. It can be proved using structural induction that the identity function is an isomorphism. □

$$<\!f,w\!>.P \xrightarrow{\emptyset,\bar{w},\{\langle f,1\rangle\}} f_1^-.P$$

$$\frac{P \xrightarrow{\emptyset,\bar{w}} \rho \quad Q \xrightarrow{\emptyset,\bar{w}'} \rho'}{P+Q \xrightarrow{\emptyset,\overline{w+w'}} (\frac{w}{w+w'}.\rho + \frac{w'}{w+w'}.\rho')} \qquad \frac{P \xrightarrow{\emptyset,\bar{w}} \rho \quad Q \not\xrightarrow{\emptyset,\bar{w}'} \quad Q \not\xrightarrow{\emptyset,\tau}}{\begin{array}{c} P+Q \xrightarrow{\emptyset,\bar{w}} \rho \\ Q+P \xrightarrow{\emptyset,\bar{w}} \rho \end{array}}$$

$$\frac{P \xrightarrow{\emptyset,\bar{w}} \rho \quad Q \xrightarrow{\emptyset,\bar{w}'} \rho'}{P \,||_{S,M}\, Q \xrightarrow{\emptyset,\overline{w+w'}} (\frac{w}{w+w'}.\rho \,||_{S,M}^{P,Q}\, \frac{w'}{w+w'}.\rho')} \qquad \frac{P \xrightarrow{\emptyset,\bar{w}} \rho \quad Q \not\xrightarrow{\emptyset,\bar{w}'} \quad Q \not\xrightarrow{\emptyset,\tau}}{\begin{array}{c} P \,||_{S,M}\, Q \xrightarrow{\emptyset,\overline{w}} (\rho \,||_{S,M}^{P,Q}\, null) \\ Q \,||_{S,M}\, P \xrightarrow{\emptyset,\overline{w}} (null \,||_{S,M}^{Q,P} \rho) \end{array}}$$

$$\frac{P \xrightarrow{\emptyset,\bar{w}} \rho \quad \forall a \in L.\ P \not\xrightarrow{\emptyset,a}}{P/L \xrightarrow{\emptyset,\bar{w}} \rho/L} \qquad \frac{P \xrightarrow{\emptyset,\bar{w}} \rho}{P[\phi] \xrightarrow{\emptyset,\bar{w}} \rho[\phi]} \qquad \frac{P\{rec\,X.P/X\} \xrightarrow{\emptyset,\bar{w}} \rho}{rec\,X.P \xrightarrow{\emptyset,\bar{w}} \rho}$$

where

$$(\rho \,||_{S,M}^{P,Q}\, \rho')(R) \stackrel{\mathrm{def}}{=} \begin{cases} \rho(\{\langle f,i\rangle\},P') & \text{if } R \equiv (\{\langle f,n(M_f)\rangle\}, P' \,||_{S,M\cup\{(\langle f,n(M_f)\rangle,l_i)\}}\, Q) \\ \rho'(\{\langle f,i\rangle\},Q') & \text{if } R \equiv (\{\langle f,n(M_f)\rangle\}, P \,||_{S,M\cup\{(\langle f,n(M_f)\rangle,r_i)\}}\, Q') \\ 0 & \text{otherwise} \end{cases}$$

$$null(P) \stackrel{\mathrm{def}}{=} 0$$

$$(\rho/L)(C,Q) \stackrel{\mathrm{def}}{=} \begin{cases} \rho(C,P) & \text{if } Q \equiv P/L \\ 0 & \text{otherwise} \end{cases} \qquad (\rho[\phi])(C,Q) \stackrel{\mathrm{def}}{=} \begin{cases} \rho(C,P) & \text{if } Q \equiv P[\phi] \\ 0 & \text{otherwise} \end{cases}$$

and

$$M_f = \{i \in \mathbb{N} \mid \exists j \in \mathbb{N}.\ \exists d \in \{r_j, l_j\}.\ (\langle f,i\rangle, d) \in M\}$$

$$n(M_f) = \min\{j \in \mathbb{N} \mid j \notin M_f\}$$

Table 3. Rules for start moves in the IGSMP calculus

$$f_i^-.P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P \qquad \frac{P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P' \quad Q \not\xrightarrow{\emptyset,\bar{w}'} \quad Q \not\xrightarrow{\emptyset,\tau}}{P+Q \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P' \quad Q+P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'}$$

$$\frac{P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P' \quad \forall a \in L. P \not\xrightarrow{\emptyset,a}}{P/L \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'/L}$$

$$\frac{P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'}{P[\phi] \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'[\phi]} \qquad \frac{P\{rec\,X.P/X\} \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'}{rec\,X.P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P'}$$

$$\frac{P \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} P' \quad (\langle f,j\rangle, l_i) \in M \quad Q \not\xrightarrow{\emptyset,\bar{w}'} \quad Q \not\xrightarrow{\emptyset,\tau}}{P \,||_{S,M}\, Q \xrightarrow{\{\langle f,j\rangle\},-,\emptyset} P' \,||_{S,M-\{(\langle f,j\rangle,l_i)\}}\, Q}$$

$$\frac{Q \xrightarrow{\{\langle f,i\rangle\},-,\emptyset} Q' \quad (\langle f,j\rangle, r_i) \in M \quad P \not\xrightarrow{\emptyset,\bar{w}'} \quad P \not\xrightarrow{\emptyset,\tau}}{P \,||_{S,M}\, Q \xrightarrow{\{\langle f,j\rangle\},-,\emptyset} P \,||_{S,M-\{(\langle f,j\rangle,r_i)\}}\, Q'}$$

Table 4. Rules for termination moves in the IGSMP calculus

7.2 $\spadesuit$ and PSA

$\spadesuit$ (read "spades") extends traditional process algebras with two new operations: one that makes it possible to start a clock and the other that waits for its termination. $\spadesuit$'s full syntax is given by adding the *clock setting* operation $\{\!|C|\!\}\, P$ and the *clock triggering* operation $C \mapsto P$ to that of the classic calculus (6), where $C \in Ck$. Process $\{\!|C|\!\}\, P$ behaves just like P except that initially it starts clocks in C and sample their termination value from their respective distribution given by function *Distr*. Process $C \mapsto P$ waits for all clocks in C to terminate and then executes P.

Unlike IGSMP, clock naming in $\spadesuit$ is a syntactic issue. Semantic rules assume terms are already well named. Processes $P \,||_A\, Q$ and $P+Q$ are well named if bounded clocks in P (i.e. those clocks in C of a subterm $\{\!|C|\!\}\, P'$ of P) do not occur in Q and those bounded in Q do not occur in P. Then $\{\!|x|\!\}\,\{x,z\} \mapsto a;\mathbf{0} \,||_a\, \{\!|y|\!\}\,\{y,z\} \mapsto a;\mathbf{0}$ is well named, while $\{\!|x|\!\}\,\{x\} \mapsto a;\mathbf{0} \,||_a\, \{\!|x|\!\}\,\{x\} \mapsto a;\mathbf{0}$ is not. All guarded terms are α-congruent to some well named processes [11].

The semantics of $\spadesuit$ in terms of SA has been defined in [14, 11] and, up to some minor notational changes, it is the same as the one we give here.

Definition 14. *Let κ be the least function satisfying equations in Table 5. Let $\longrightarrow$ be the least relation satisfying rules in Table 6. The* semantics *of $P \in \spadesuit$ is given by the SA $\mathsf{PSA}(P) \stackrel{\text{def}}{=} (St_P, Ck, Distr, Act, \longrightarrow, P, \kappa(P))$ where:*

- *St_P is the subset of $\spadesuit$ such that (a) $P \in St_P$, and (b) if $Q \in St_P$, $Q \xrightarrow{C_g,a,C_r} Q'$, then $Q' \in St_P$;*
- *Ck and Distr are just like in $\spadesuit$; and*

– *if* Act_d *is the same set as the set of* $\spadesuit$ *action names, and* $Act_u = \emptyset$ *we say that the semantics is* open, *if instead* $Act_d = \emptyset$ *and* Act_u *is the set of* $\spadesuit$ *action names, we say that the semantics is* closed

$$
\begin{array}{ll}
\kappa(a.P) = \kappa(\mathbf{0}) = \emptyset & \kappa(P + Q) = \kappa(P \,||_S\, Q) = \kappa(P) \cup \kappa(Q) \\
\kappa(\{\!|C|\!\}\, P) = \kappa(P) \cup C & \kappa(C \mapsto P) = \kappa(P/L) = \kappa(P[\phi]) = \kappa(rec\, X.P) = \kappa(P)
\end{array}
$$

Table 5. Clock resetting function in $\spadesuit$

$$
a.P \xrightarrow{\emptyset, a, \kappa(P)} P
\qquad
\frac{P \xrightarrow{C_g, a, C_r} P' \quad a \in L}{P/L \xrightarrow{C_g, \tilde{\tau}, C_r} P'/L}
$$

$$
\frac{P \xrightarrow{C_g, a, C_r} P'}{\{\!|C|\!\}\, P \xrightarrow{C_g, a, C_r} P'}
\qquad
\frac{P \xrightarrow{C_g, a, C_r} P'}{C \mapsto P \xrightarrow{C \cup C_g, a, C_r} P'}
\qquad
\frac{P \xrightarrow{C_g, a, C_r} P' \quad a \notin L}{P/L \xrightarrow{C_g, a, C_r} P'/L}
$$

$$
\frac{P \xrightarrow{C_g, a, C_r} P'}{\begin{array}{c} P + Q \xrightarrow{C_g, a, C_r} P' \\ Q + P \xrightarrow{C_g, a, C_r} P' \end{array}}
\qquad
\frac{P \xrightarrow{C_g, a, C_r} P' \quad a \notin S}{\begin{array}{c} P \,||_S\, Q \xrightarrow{C_g, a, C_r} P' \,||_S\, Q \\ Q \,||_S\, P \xrightarrow{C_g, a, C_r} Q \,||_S\, P' \end{array}}
\qquad
\frac{P \xrightarrow{C_g, a, C_r} P'}{P[\phi] \xrightarrow{C_g, \phi(a), C_r} P'[\phi]}
$$

$$
\frac{P \xrightarrow{C_g, a, C_r} P' \quad Q \xrightarrow{C'_g, a, C'_r} Q' \quad a \in S}{P \,||_S\, Q \xrightarrow{C_g \cup C'_g, a, C_r \cup C'_r} P' \,||_S\, Q'}
\qquad
\frac{P\{rec\, X.P/X\} \xrightarrow{C_g, a, C_r} P'}{rec\, X.P \xrightarrow{C_g, a, C_r} P'}
$$

Table 6. Rules for $\spadesuit$

Function $\kappa(P)$, given in Table 5, defines the set of clocks that needs to be started before executing P. For instance, in process $\{\!|x|\!\}\, \{x\} \mapsto a; Q$, clock x has to be started in order to wait for it to terminate and then enables the execution of a. Then, $\kappa(\{\!|C|\!\}\, P)$ has to include C and those clocks that are started in P (it could be that $P \equiv \{\!|C'|\!\}\, Q$ for some C' and Q). $P + Q$ needs to start all clocks started by both P and Q and similarly for $P \,||_A\, Q$. Function κ is used to define the clocks to be set in a control transition and to define the clocks to be started at initialization.

Rules in Table 6 define the control transition. $a.P$ can perform action a at any moment; therefore it does not wait for any clock to terminate. When this transition is executed all clocks in $\kappa(P)$ are started. $C \mapsto P$ performs any activity P does but after all clocks in C are terminated. Notice that $\{\!|C|\!\}\, P$ proceeds exactly like P. The behavioural difference lies in the clocks to start at initialization. Rules for $P + Q$ and $P \,||_A\, Q$ are quite standard except the rule for synchronization. In a synchronizing action $a \in A$ in $P \,||_A\, Q$, both P and

Q should be ready to perform it, so all clocks controlling a in P and all clocks controlling a in Q have to terminate. Moreover, when the transition is executed it starts all clocks that P and Q would have started independently.

Rules for the other operators follow the usual definitions. Notice, in particular, that the hiding operation is hiding to a silent action $\tilde{\tau} \in \mathit{Act}$ which is *not* the same action τ considered as the maximum of Act_u in Def. 1. $\spadesuit$ does not impose maximal progress on the silent step. In this sense, it is not different from any other action. However, the silent step cannot synchronize and cannot be renamed.

7.3 GSPA and PSA

Katoen et al. [9, 23] introduced a generalized stochastic process algebra (GSPA) and gave semantics to it in terms of SBES. GSPA introduces a stochastic timed action prefix $(f)a.P$ which replaces the action prefix in the classic calculus (6). $(f)a.P$, where $f \in \mathit{PDF}$, executes action a after waiting an amount of time sampled according to f, and then it behaves like P.

Notice that SBESs do not contain clocks. Instead, distributions are associated to each causal link (i.e. either bundles or the execution starting time). So clock naming was not a problem in GSPA's original semantics. However, since each execution of an action is considered a different event in this setting, event naming had to be considered with care [23].

In order to give semantics to GSPA in terms of PSA, we use a static clock naming technique explained in the following. First, define an auxiliary term $\langle f, i\rangle a.P$, with $\langle f, i\rangle \in \mathit{PDF} \times \mathbb{N}$. This term is a particular clock naming of the distribution f governing the delay of $(f)a.P$ and it is assigned according to function π defined in Table 7. More precisely, π is a function that looks for the set of clocks to be started on arriving to P, activates them and assigns them a name: given a GSPA process P and a set $\mathcal{C}$ of already active clock names, it returns an appropriately named process P' together with the new set of active clock names (which extends $\mathcal{C}$) and the set of clocks that it has activated. Then, if GSPA$'$ is the set of all terms in this extended syntax, $\pi : (\text{GSPA}' \times 2^{\mathit{Ck}}) \to (2^{\mathit{Ck}} \times (\text{GSPA}' \times 2^{\mathit{Ck}}))$. In particular, $\pi((f)a.P, \mathcal{C})$ returns process $\langle f, i\rangle a.P$ provided $\langle f, i\rangle \notin \mathcal{C}$; then $\mathcal{C} \cup \{\langle f, i\rangle\}$ is the new set of active clock names and $\langle f, i\rangle$ is the only clock that needs to be set on arriving to $\langle f, i\rangle a.P$. Compare this with $\pi(\langle f, i\rangle a.P, \mathcal{C}) = (\emptyset, (\langle f, i\rangle a.P, \mathcal{C}))$. Since $\langle f, i\rangle$ is a clock name, no new clock has to be created. Moreover, no clock has to be set on arriving to $\langle f, i\rangle a.P$ in this case, since it was already set when $\langle f, i\rangle$ was created. Notice that π defines indeed a static naming: to name clocks for Q in $P \,||_A\, Q$ or $P + Q$, π "looks" how clocks were named on P (first line in Table 7). In fact, $P \,||_A\, Q$ and $Q \,||_A\, P$ have associated different semantics objects (and similarly for $P + Q$ and $Q + P$). This does not happen under a dynamic technique (compare to IGSMP).

The interpretation of a GSPA process in terms of PSA is given in the following.

Definition 15. *Let π be the least function satisfying equations in Table 7. Let $\longrightarrow$ be the least relation satisfying rules in Table 8. The* semantics *of the GSPA term P in terms of PSA is given by* $\mathsf{PSA}(P) \stackrel{\text{def}}{=} (St_P, Ck, Distr, Act, \longrightarrow, (P', \mathcal{C}), C)$ *where:*

- *$St_P \subseteq GSPA' \times Ck$ such that (a) $(P', \mathcal{C}) \in St_P$, and (b) if $(Q, \mathcal{C}) \in St_P$, $(Q, \mathcal{C}) \xrightarrow{C_g, a, C_r} (Q', \mathcal{C}')$, then $(Q', \mathcal{C}') \in St_P$;*
- *$Ck = PDF \times I\!N$ is a set of clocks, and $Distr$ is defined by $Distr(\langle f, i\rangle) = f$;*
- *P', $\mathcal{C}$, and C are such that $\pi(P, \emptyset) = (C, (P', \mathcal{C}))$; and*
- *if Act_d is the same set as the set of GSPA action names, and $Act_u = \emptyset$ we say that the semantics is* open*, if instead $Act_d = \emptyset$ and Act_u is the set of GSPA action names, we say that the semantics is* closed

States in $\mathsf{PSA}(P)$ are pairs $(Q, \mathcal{C})$ where Q is an extended GSPA process and $\mathcal{C}$ is the set of active clock names. GSPA does not have preselection policy; therefore it does not include probabilistic jump and transitions are trivial. Hence $\mathsf{PSA}(P)$ is a stochastic automata.

Rules in Table 8 define the transition relation. There is no rule for $(f)a.P$ since clocks are not named. The rule for the auxiliary prefix states that whenever the system is in control state $(\langle f, i\rangle a.P, \mathcal{C})$ (for any $\mathcal{C} \in Ck$), it can perform action a provided clock $\langle f, i\rangle$ has terminated; afterwards, it sets clocks in C_P and moves to $(P', \mathcal{C}_P)$ where P' is the clock named version of P, $\mathcal{C}_P$ is the new set of active clocks (notice that $\langle f, i\rangle$ was removed from the set of active clocks), and C_P is the set of clocks that becomes active in P'. For the other operators, rules are very similar to $\spadesuit$, except synchronization that needs special care on clock naming: notice that the set of active clocks for the source of Q transition is the set of active clocks for the target of P transition (namely, $\mathcal{C}_P$). Besides, like in $\spadesuit$ the silent step $\tilde{\tau} \in Act$ is not action $\tau \in Act_u$. It is only special in the sense that it cannot synchronize neither be renamed.

In the following we state that the translation to PSA of the process, or the translation to PSA of its semantics in terms of SBES is the same up to bisimulation of its interpretations.

Theorem 4. *Let P be a GSPA process. Then $[\![\mathsf{PSA}(P)]\!] \sim [\![\mathsf{PSA}(\mathcal{E}_S[\![P]\!])]\!]$ both in the open and closed interpretation.*

The proof of this theorem makes use of the definition of symbolic bisimulation given in [11]. We omit it here as it requires some technical knowledge of both [23] and [11]. Nevertheless we observe that bisimulation (as defined in this paper) of $\mathsf{PSA}(P)$ and $\mathsf{PSA}(\mathsf{SBES}(P))$ is not possible as it can be seen in the next example. Consider the GSPA process $P = (f)a.\mathbf{0} \mid\mid_a (g)a.\mathbf{0}$. Then

1. $\mathsf{PSA}(P)$ initially sets clocks $\langle f, 1\rangle$ and $\langle g, 1\rangle$ and contains the only transition:

$$(\langle f, 1\rangle a.\mathbf{0} \mid\mid_a \langle g, 1\rangle a.\mathbf{0}, \{\langle f, 1\rangle, \langle g, 1\rangle\}) \xrightarrow{\{\langle f,1\rangle, \langle g,1\rangle\}, a, \emptyset} (\mathbf{0} \mid\mid_a \mathbf{0}, \emptyset)$$

Provided

$$\pi(P,\mathcal{C}) = (C_P,(P',\mathcal{C}_P)) \quad \text{and} \quad \pi(Q,\mathcal{C}_P) = (C_Q,(Q',\mathcal{C}_Q))$$

we define

$$\pi(\mathbf{0},\mathcal{C}) = (\emptyset,(\mathbf{0},\mathcal{C}))$$
$$\pi((f)a.P,\mathcal{C}) = (\{\langle f,i\rangle\},(\langle f,i\rangle a.P,\mathcal{C}\cup\{\langle f,i\rangle\}))$$
$$\text{with } i = \min\{j \in \mathbb{N} \mid \langle f,j\rangle \notin \mathcal{C}\}$$
$$\pi(\langle f,i\rangle a.P,\mathcal{C}) = (\emptyset,(\langle f,i\rangle a.P,\mathcal{C}))$$
$$\pi(P+Q,\mathcal{C}) = (C_P \cup C_Q,(P'+Q',\mathcal{C}_Q))$$
$$\pi(P \,||_A\, Q,\mathcal{C}) = (C_P \cup C_Q,(P' \,||_A\, Q',\mathcal{C}_Q))$$
$$\pi(P/L,\mathcal{C}) = (C_P,(P'/L,\mathcal{C}_P))$$
$$\pi(P[\phi],\mathcal{C}) = (C_P,(P'[\phi],\mathcal{C}_P))$$

Table 7. Defining the clocks to set and the next state in GSPA

$$\frac{\pi(P,\mathcal{C}-\{\langle f,i\rangle\}) = (C_P,(P',\mathcal{C}_P))}{(\langle f,i\rangle a.P,\mathcal{C}) \xrightarrow{\{\langle f,i\rangle\},a,C_P} (P',\mathcal{C}_P)}$$

$$\frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}')}{\begin{array}{c}(P+Q,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}') \\ (Q+P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}')\end{array}} \qquad \frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}') \quad a \notin S}{\begin{array}{c}(P \,||_S\, Q,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P' \,||_S\, Q,\mathcal{C}') \\ (Q \,||_S\, P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (Q \,||_S\, P',\mathcal{C}')\end{array}}$$

$$\frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}_P) \quad (Q,\mathcal{C}_P) \xrightarrow{C'_g,a,C'_r} (Q',\mathcal{C}_Q) \quad a \in S}{(P \,||_S\, Q,\mathcal{C}) \xrightarrow{C_g \cup C'_g,a,C_r \cup C'_r} (P' \,||_S\, Q',\mathcal{C}_Q)}$$

$$\frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}') \quad a \in L}{(P/L,\mathcal{C}) \xrightarrow{C_g,\tilde{\tau},C_r} (P'/L,\mathcal{C}')} \qquad \frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}') \quad a \notin L}{(P/L,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P'/L,\mathcal{C}')}$$

$$\frac{(P,\mathcal{C}) \xrightarrow{C_g,a,C_r} (P',\mathcal{C}')}{(P[\phi],\mathcal{C}) \xrightarrow{C_g,\phi(a),C_r} (P'[\phi],\mathcal{C}')}$$

Table 8. Rules for GSPA

2. According to [23], the interpretation of P in terms of SBES is given by $\mathsf{SBES}(P)$ where $E = \{(e_l, e_r)\}$, $\rightsquigarrow = \emptyset$, $\rightarrowtail = \emptyset$, $l(e_l, e_r) = a$, $\mathcal{F}(e_l, e_r) = \max(f, g)$, and $\mathcal{G} = \emptyset$. That is, the synchronization of two events e_l and e_r is given by a new event (e_l, e_r) that couples them, and whose time delay is given by a random variable which is the maximum of the random variables associated to each synchronizing event. Hence, according to Def. 12, $\mathsf{PSA}(\mathsf{SBES}(P))$ initially sets the *only* clock (e_l, e_r) and has the only transition $\emptyset \xrightarrow{\{(e_l,e_r)\},a,\emptyset} \{(e_l, e_r)\}$.

The structures of $\mathsf{PSA}(P)$ and $\mathsf{PSA}(\mathsf{SBES}(P))$ are clearly different and not bisimilar. Still, though so different in structure, it is not difficult to notice that their stochastic behavior is the same: both processes will have to wait some time which will be the maximum of two values, one sampled according to f, and the other sampled according to g.

8 Notes and discussions

Recall that a control transition $s \xrightarrow{C,a} \rho$ in PSA performs four activities:

1. *waits for termination of timers*, that is, it waits for clock in C to terminate, hence enabling the transition,
2. *executes an action*, in this case action a, as part of the transition to a new state,
3. *start timers* as part of a probabilistic jump (recall that ρ is a distribution on $2^{Ck} \times St$), and
4. *probabilistically selects the next state* according to the distribution ρ.

Let us analyze how each of this issues are treated by different frameworks including those above discussed.

IGSMP and its calculus. The four ingredients are present in this framework. IGSMP provides three different kind of transitions: one to start a timer, another to wait for its termination, and the last one to show the execution of an action. The probabilistic jump is provided together with the start transition, and only at this level a probabilistic selection is possible. Action transitions are selected non-deterministically and termination transitions are selected in the moment their clocks terminate. In this sense, we could say that start transitions provide preselection policy, while termination transition provide a race selection policy.

Though the calculus provides the action prefix operation $a.P$ which relates to action execution, the other three ingredients provided by PSA are gathered in only one operation: the delay prefix $<f, w>.P$ which is given an ST-like semantics. Still the calculus provides auxiliary separated operations for clock start and clock termination. This auxiliary operations are used in order to achieve a sound and complete axiomatization of bisimulation, and to provide an expansion law.

SA and $\spadesuit$*.* As transitions in SA are PSA transitions with a trivial probabilistic jump, the fourth ingredient is missing in this framework. The rest is treated as in PSA. Therefore SA and $\spadesuit$ do not provide preselection policy.

$\spadesuit$ provides the same three ingredients in three different basic operations: $\{\!|C|\!\}\, P$, $C \mapsto P$, and $a.P$. Again, the separations of concern makes it possible to achieve a sound and complete axiomatization of the algebra, and to obtain an expansion law.

SBES and GSPA. Rather than an interleaving model, Katoen et al. decided to pursue a true concurrency approach dropping, therefore, the idea of having an expansion law and any sensible axiomatization. Hence, parallel composition is considered a basic operation. From this viewpoint, there is no need to separate the different issues in different ingredients. Notice that in SBES clock start and termination are hidden and associated to bundles that enable events. Similarly, GSPA provides the clock start and termination, and action execution in only one prefix operation $(f)a.P$. Like SA and $\spadesuit$, this framework does not provide preselection policy since there is no probabilistic selection.

Strulo's SPADES*.* SPADES [36, 18] is one of the earliest stochastic process algebras with generally distributed delays. Its design is such that the process algebra is directly mapped in the concrete model (similar to our PTTS) and no intermediate symbolic model is defined. Contrarily to the process algebras discussed above, SPADES provides a considerably larger variety of basic operations including four different prefix operations, a non-deterministic choice and a probabilistic choice (which allows for preselection policy).

Among the prefix operations, we encounter the action prefix $a.P$, a sampling prefix $\mathcal{R}[t \leftarrow f].P$, which samples a value according to distribution f, saves it in variable t and executes P, and a delay operation $(t).P$ which delays the amount of time indicated by variable t. This separation of concerns is different for the one proposed by $\spadesuit$ and IGSMP. Notice that, $(t).P$ involves both start and termination of a timer, but its sampling is explicitly done in a separate operation $\mathcal{R}[t \leftarrow f].P$. As a consequence it cannot be encoded in PSA. Besides, the fact that start and termination of a timer is not separated in different operations prevents to decompose the parallel composition by means of an expansion law. So, strictly speaking, parallel composition should also be considered a basic operation as a general expansion law is not possible in this framework. As a consequence, though a large set of sound laws have been given for SPADES, finding a complete axiomatization is not easy.

NMSPA. NMSPA [28] has semantics in a particular kind of model that combines both symbolic and concrete views. Probability measures are not explicit. Rather, it labels the transition with a random variable. Besides, transitions are also labelled with a non-negative real value indicating passage of time. As a consequence, the semantic object associated to a process is uncountably large.

Some further remarks. For most of the algebras, a decision was taken to separate action execution, timer start, and timer termination in different operations. Such a design choice was made specially to achieve a solid algebraic framework, and therefore to facilitate the syntactic manipulation of the language. However, at the moment of modelling a system, the stochastic timed action prefix of GSPA seems to be enough.

A remark that holds in general is that race policy is present in all the previously discussed frameworks. In most of them, it can be simply present in a choice operation. The calculus for IGSMP requires a little of encoding but it is still present as the underlying model (the IGSMP) allows it.

Other process algebras that we have not explicitly discussed but is worth to mention are TIPP [17], stochastic π calculus [32], and GSMPA [6], among others. The first two were given semantics on infinite symbolic transition systems (i.e. based on distributions and not on real valued transitions) that required to carried conditional distribution function in a similar manner that we do in the spent lifetime semantics. Besides, expansion laws were not possible in these contexts. GSMPA can be regarded as a variant of IGSMPs which is based strictly on ST semantics as IGSMPs, but where names are generated statically (instead of dynamically) according to syntactic locations of delays w.r.t. to parallel composition operators in the algebraic term. Notably in [6] a definition of symbolic bisimulation equivalence based on clock associations is defined which accounts for probabilistic choices.

9 Conclusion

We thoroughly discussed the characteristics of stochastic process algebra with general distributions and presented a unifying setting, namely PSA. We could map several well known stochastic formal frameworks on PSA and see that their differences are not that significant. This says that, as far as the methodology for representing general distributions is concerned, the expressiveness of all the existing stochastic process algebras and their respective semantic models is basically the same. That is, stochastic action prefix (like in GSPA) non-deterministic choice, parallel composition, renaming (and hiding), and recursion constitute a "core" algebra that will suffice for representing general distributions in concurrent systems at, basically, the same expressive level as in all such algebras. The only one exception is probabilistic choice that turns to be a useful operator and is not present in all algebras. It appears in SPADES and in IGSMP when two weighted delayed prefixes are involved, e.g., in a choice.

Though residual and spent lifetime semantics were studied in different contexts, they are actually orthogonal to the original symbolic model. This is shown by the fact that PSA can be interpreted under both viewpoints.

Finally, it is worth to mention that the different frameworks were designed with some different objectives in mind. For instance, the true concurrent approach of SBES was originally pursued to propose an alternative semantic view of the cumbersome interleaving models for general distributions of its time. The design of GSMPA/IGSMP, instead, was strongly based on a conceptual study of the kind of semantics needed for a process algebra with general distributions, ending up with ST semantics (see [4]). On the other hand, the design of SA is related to that of timed automata with the idea that SA could be model checked at least in a stochastically abstracted setting (see [1, 12, 25]).

We finish with a brief note on open problems. An important development of the work on stochastic systems would be to understand how to develop an equivalence notion which equates different patterns of generally distributed delays as follows. In the definition of [31] the internal computations of processes are standard "τ" actions. In an algebra with generally distributed delays we can see also a delay as an internal computation (a *timed "τ"*). Therefore the idea is to extend the notion of bisimulation in such a way that it can equate, e.g., a sequence of timed τ with a single timed τ provided that distribution of durations are in the correct relationship. For example a sequence (or a more complex pattern) of exponential timed τ could be equated by a phase-type distributed timed τ. It is worth noting that the possibility of extending the notion of bisimulation in this way strictly depends on the fact that we can express delays with any duration distribution (in languages expressing exponential distributions only there is not such a possibility). This is desirable because it would lead to a significant state space reduction of semantic models. See [30] for a solution of this problem in the context of semi-Markov processes, i.e. in the absence of a parallel composition operator.

Acknowledgements. We thank the anonymous referees for their useful remarks. Moreover, we thank the organisers of the VOSS event to bring us two together on writing this article, specially Holger Hermanns who has been very helpful on suggestions and mediations.

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, editors, *Proceedings of the* 18^{th} *International Colloquium Automata, Languages and Programming (ICALP'91),* Madrid, volume 510 of *Lecture Notes in Computer Science*, pages 113–126. Springer-Verlag, 1991.
2. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
3. M. Bernardo. *Theory and Application of Extended Markovian Process Algebras.* PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 1999.
4. M. Bravetti. *Specification and Analysis of Stochastic Real-Time Systems.* PhD thesis, Dottorato di Ricerca in Informatica. Università di Bologna, Padova, Venezia, February 2002. Available at `http://www.cs.unibo.it/~bravetti/`.
5. M. Bravetti and M. Bernardo. Compositional asymmetric cooperations for process algebras with probabilities, priorities, and time. In *Proc. of the 1st Int. Workshop on Models for Time-Critical Systems, MTCS 2000,* State College (PA), volume 39(3) of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2000.
6. M. Bravetti, M. Bernardo, and R. Gorrieri. Towards performance evaluation with general distributions in process algebra. In D. Sangiorgi and R. de Simone, editors, *Proceedings CONCUR 98,* Nice, France, volume 1466 of *Lecture Notes in Computer Science*, pages 405–422. Springer-Verlag, 1998.

7. M. Bravetti and R. Gorrieri. Deciding and axiomatizing weak st bisimulation for a process algebra with recursion and action refinement. *ACM Transactions on Computational Logic*, 3(4):465–520, 2002.
8. M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science*, 282:5–32, 2002.
9. E. Brinksma, J.-P. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, 38(6):552–565, 1995.
10. D.R. Cox. The analysis of non-markovian stochastic processes by the inclusion of supplementary variables. In *Proc. of the Cambridge Philosophical Society*, volume 51, pages 433–440, 1955.
11. P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.
12. P.R. D'Argenio. A compositional translation of stochastic automata into timed automata. Technical Report CTIT 00-08, Department of Computer Science, University of Twente, 2000.
13. P.R. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. MoDeST – a modelling and description language for stochastic timed systems. In L. de Alfaro and S. Gilmore, editors, *Proc. of PAPM/PROBMIV 2001,* Aachen, Germany, volume 2165 of *Lecture Notes in Computer Science*, pages 87–104. Springer-Verlag, September 2001.
14. P.R. D'Argenio, J.-P. Katoen, and E. Brinksma. An algebraic approach to the specification of stochastic systems (extended abstract). In D. Gries and W.-P. de Roever, editors, *Proceedings of the IFIP Working Conference on Programming Concepts and Methods, PROCOMET'98,* Shelter Island, New York, USA, IFIP Series, pages 126–147. Chapman & Hall, 1998.
15. R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Proceedings PARLE conference,* Eindhoven*, Vol. II (Parallel Languages)*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242. Springer-Verlag, 1987.
16. P.W. Glynn. A GSMP formalism for discrete event simulation. *Proceedings of the IEEE*, 77(1):14–23, 1989.
17. N. Götz, U. Herzog, and M. Rettelbach. TIPP - Introduction and application to protocol performance analysis. In H. König, editor, *Formale Beschreibungstechniken für verteilte Systeme*, FOKUS series. Saur Publishers, 1993.
18. P.G. Harrison and B. Strulo. Spades: Stochastic process algebra for discrete event simulation. *Journal of Logic and Computation*, 10(1):3–42, 2000.
19. H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
20. J. Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertation in Computer Science. Cambridge University Press, 1996.
21. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.
22. K. Kanani. *A Unified Framework for Systematic Quantitative and Qualitative Analysis of Communicating Systems*. PhD thesis, Imperial College (UK), 1998.
23. J.-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures*. PhD thesis, Department of Computer Science, University of Twente, April 1996.
24. J.-P. Katoen, E. Brinksma, D. Latella, and R. Langerak. Stochastic simulation of event structures. In Ribaudo [33], pages 21–40.
25. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In C. Palamidessi, editor,

Proceedings CONCUR 2000, State College, Pennsylvania, USA, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137. Springer-Verlag, 2000.
26. K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.
27. A.M. Law and W.D. Kelton. *Simulation Modelling and Analysis*. McGraw-Hill Inc., second edition, 1991.
28. N. López and M. Núñez. NMSPA: A non-Markovian model for stochastic processes. In *Proceedings of the International Workshop on Distributed System Validation and Verification (DSVV'2000),* Taipei, Taiwan, ROC, 2000. Available at `http://www.math.ntu.edu.tw/~eric/dsvv_proc/`.
29. N. López and M. Núñez. A testing theory for generally distributed stochastic processes (extended abstract). In K. Larsen and M. Nielsen, editors, *Proceedings CONCUR 2001,* Aalborg, Denmark, volume 2154 of *Lecture Notes in Computer Science*, pages 321–335. Springer-Verlag, 2001.
30. N. López and M. Núñez. Weak stochastic bisimulation for non-markovian processes, 2003. Submitted for publication.
31. R. Milner. *Communication and Concurrency*. Prentice-Hall International, 1989.
32. C. Priami. Stochastic π-calculus with general distributions. In Ribaudo [33], pages 41–57.
33. M. Ribaudo, editor. *Proc. of the 4th Workshop on Process Algebras and Performance Modelling, PAPM'96,* Torino, Italy. Università di Torino, 1996.
34. Theo C. Ruys, Rom Langerak, Joost-Pieter Katoen, Diego Latella, and Mieke Massink. First passage time analysis of stochastic process algebra using partial orders. In Tiziana Margaria and Wang Yi, editors, *Proc. of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001,* Genova, Italy, volume 2031 of *Lecture Notes in Computer Science*, pages 220–235. Springer-Verlag, 2001.
35. R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1995.
36. B. Strulo. *Process Algebra for Discrete Event Simulation*. PhD thesis, Department of Computing, Imperial College, University of London, 1993.
37. G. Winskel. *Events in Computation*. PhD thesis, Department of Computer Science, University of Edinburgh, 1980.